Techniques for Improving Web Search by Understanding Queries

by

Daniel Wayne Crabtree

Submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2011

Abstract

This thesis investigates the refinement of web search results with a special focus on the use of clustering and the role of queries. It presents a collection of new methods for evaluating clustering methods, performing clustering effectively, and for performing query refinement.

The thesis identifies different types of query, the situations where refinement is necessary, and the factors affecting search difficulty. It then analyses hard searches and argues that many of them fail because users and search engines have different query models.

The thesis identifies best practice for evaluating web search results and search refinement methods. It finds that none of the commonly used evaluation measures for clustering meet all of the properties of good evaluation measures. It then presents new quality and coverage measures that satisfy all the desired properties and that rank clusterings correctly in all web page clustering situations.

The thesis argues that current web page clustering methods work well when different interpretations of the query have distinct vocabulary, but still have several limitations and often produce incomprehensible clusters. It then presents a new clustering method that uses the query to guide the construction of semantically meaningful clusters. The new clustering method significantly improves performance.

Finally, the thesis explores how searches and queries are composed of different aspects and shows how to use aspects to reduce the distance between the query models of search engines and users. It then presents fully automatic methods that identify query aspects, identify underrepresented aspects, and predict query difficulty. Used in combination, these methods have many applications — the thesis describes methods for two of them. The first method improves the search results for hard queries with underrepresented aspects by automatically expanding the query using semantically orthogonal keywords related to the underrepresented aspects. The second method helps users refine hard ambiguous queries by identifying the different query interpretations using a clustering of a diverse set of refinements. Both methods significantly outperform existing methods.

Contents

Chapter 1

Introduction

In the context of information retrieval [179, 107], search is the problem of distilling the few relevant items from the many irrelevant items in a corpus. The corpus can be any collection of items, physical or virtual, homogeneous or heterogeneous. For example, the corpus might be a library of books, a collection of telephone numbers, or a collection of digital audio, documents, and videos. The relevant items are those that satisfy the user's information need or "search goal", for example, to find Java programming books or Bob's telephone number.

Users can search an unorganized corpus by examining every item and determining one-by-one if they are relevant. While exhaustive search can be practical for very small corpora such as the sentences in a single short article, it becomes time consuming for even very modest corpora such as the telephone numbers of a small town. For more interesting corpora such as the trillions of web pages on the internet [4], exhaustive search is completely impractical. To help users search more efficiently, search tools organize or "index" the contents of a corpus. For example, telephone directories order surnames with associated telephone numbers and book indexes order keywords with associated page numbers.

In the context of the web, the corpus is a collection of documents (web

pages), and the search tools are called search engines.[1] The typical search engine [20] organizes the web by crawling it to find all accessible documents and building indexes that map terms (words or phrases) to the documents in which they occur. Upon receiving a "query" from a user, the typical search engine uses its index to find documents containing all the query's terms, orders those documents by a measure of relevance, and then presents this linear list of documents — the search results or "result set" — to the user as shown in figure 1.1.

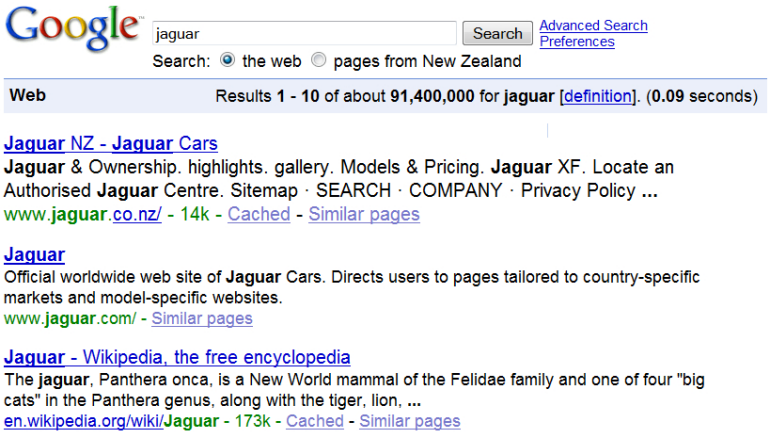

Figure 1.1: Search results for the query "jaguar" from Google

Users approach a search engine when they have a search goal and they communicate that goal to the search engine by expressing it as a query consisting of a set of keywords. For example, a user with the goal of finding information about the jaguar animal may approach Google, a popular web search engine, and enter the query "jaguar" as shown in figure 1.1. The user then evaluates each document in turn assessing its relevance against their search goal. For example, they would find documents 1 and 2 from figure 1.1 irrelevant as they relate to the Jaguar car and find document 3

[1] There are also web based search engines for other corpora such as maps and videos, and general search engines often search multiple corpora at once. Unless specifically stated, this thesis discusses search engines that search web pages.

relevant as it relates to the animal. If the user were unsatisfied with the result set because it contained too few relevant documents or too many irrelevant documents, they would refine their query (by adding, removing, or altering keywords [119]) and submit the new query to the search engine, restarting the search process. This process [55, 56] repeats until they solve their search goal or abandon it.

A search is successful if the user finds enough relevant documents to satisfy their search goal within a reasonable timeframe. Current search engines are successful for many searches, but for some searches, it is very difficult and very time-consuming [94] for users to construct effective queries that communicate their search goal. Users waste significant time refining queries for these hard searches because current search engines do not help users refine queries and because users and search engines understand queries differently, essentially they speak different languages.

With more than 130 billion searches in December 2009 [109] from 1.8 billion internet users [163], up from 61 billion searches from 750 million users in August 2007 [115], there are many users who would benefit from improved search engines and the potential to improve productivity is immense.

This thesis examines problems with the current methods of searching the web and examines why efforts to solve those problems have not yet succeeded. Based on that analysis, the thesis develops several new methods that help users refine queries and that help search engines understand the queries users construct better. The thesis also analyzes how to evaluate these methods and where necessary, develops new evaluation measurements.

1.1 Hard Searches

The typical web user believes that search engines work exceptionally well and they say things such as "I *always* find what I want" and "it can read

my mind". The reality is that current search engines work very well for some searches (the easy searches), but not for others (the hard searches).

Hard searches are those where it takes many queries and a lot of time for users to find relevant documents and those where users never find any relevant documents. While accurate, this definition of hard searches is too abstract to be useful for analysis. Therefore, for analysis this thesis uses the following definition: easy searches take less than 5 minutes, hard searches take more than 5 minutes, and very hard searches take more than 30 minutes.[2]

Improving any search is desirable, but the typical user is generally satisfied by finding just a few relevant documents [139][3] and is very sensitive to the time and effort devoted to search [131]. Consequently, it is more valuable to users to reduce the time they spend refining queries and to increase the fraction of relevant documents among the top ranked documents (precision), than it is to find more documents that are relevant (recall) or to improve the ordering of low ranked documents.

Table 1.1: A log of an easy search (15 seconds) for a high school from the AOL query logs [139]

Timeline (mm:ss)	Query
00:00	de witt clinton public high school
00:13	dewitt clinton public high school

There is marginal value in improving easy searches that search engines already solve satisfactorily because users spend very little time refining the queries for these searches and the first few documents are usually rel-

[2]The literature [2, 3] sometimes uses a third definition based on Precision — easy queries have high Precision and hard queries have low Precision. The evaluations in chapter 6 predominantly use this third definition.

[3]Few users look beyond the first few results.

Table 1.2: A log of a hard search (almost 30 minutes) for nurse licensing information from the AOL query logs [139]

Timeline (mm:ss)	Query
00:00	nursing registry
04:18	certified nursing assistant 1
08:48	nursing assistant registry
09:48	license look up for nursing assistants
10:06	nursing assistant 1 certification
11:42	nursing assistant 1 license look ups
12:18	nursing assistant 1 expiration look up
12:30	nursing registry in Raleigh
13:24	nursing aide registry of Raleigh
15:00	nursing aide registry of Raleigh website
16:06	nursing aide registry of Raleigh
19:48	north carolina board of nursing information for nursing assistant 1
22:24	license look up for nursing assistant 1
24:36	license information for nursing assistant 1 expiration
28:30	north carolina nursing assistant 1 license information

evant. In contrast, there is significant value in improving hard searches, because users must often make a time consuming series of refinements, examine many irrelevant documents, and examine many result pages to find a few relevant documents. For example, table 1.1 shows a log of a user finding relevant documents in less than 15 seconds for an easy search, while table 1.2 shows a log of another user spending almost 30 minutes to find relevant documents for a hard search [139]. Furthermore, users currently fail to solve many hard queries, even when relevant documents exist; increasing the number of successful searches provides additional value

to users.

Users spend most of their time on hard searches. Studies of search engine usage [94, 139][4] show that 48% of searches need just one query and 80% need at most three (the easy searches). However, the remaining 20% (the hard searches) take 80% of the user's time and effort to refine as shown in figure 1.2. The excellent performance on easy searches is what leads the typical user to believe incorrectly that current search engines have already solved the search problem.

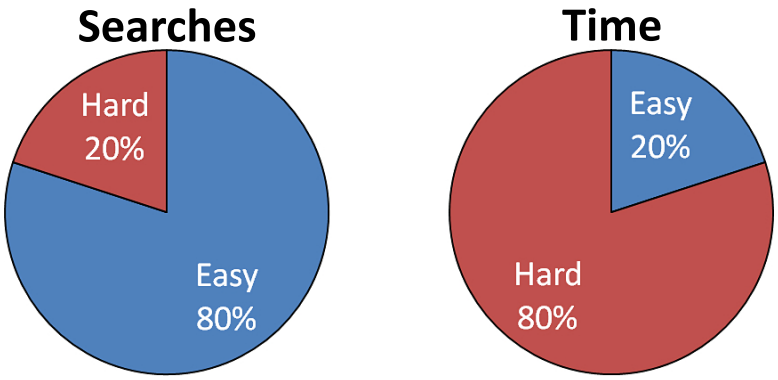

Figure 1.2: Relatively few hard searches consume most of the time users spend searching

Hard searches affect all users, including experts. Knowledge about how to construct effective queries can help users solve some hard searches. For example, when a result set has no results because the query contains too many terms, an experienced user will remove the least important terms. However, knowledge is insufficient to overcome the limitations of the typical search engine for harder searches such as those where the search goal has multiple components. For these harder searches, even experienced users can struggle to formulate effective queries. For example, I spent 7 hours searching for papers containing statistics on search engine usage before finding an effective query and several relevant research papers.

[4]Chapter 2 elaborates on these results.

For both easy and hard searches, there are two classes of search failure: those where the user's query is inaccurate or imprecise such as ambiguous queries like "Jaguar" and those where the search engine understands the user's query incorrectly such as queries with too many terms. For the first kind of failure, refinement is unavoidable, but for the second kind, if a search engine correctly interpreted the user's intention for the query, there would be no need for refinement. One way to improve a search engine's performance on queries is to improve the search engine's relevance measure; another way is to automatically extend or modify queries to express the goals of the queries better.

1.1.1 Relevance Measures

The intelligence of a typical search engine is stored in its relevance measure and search model. Ideally, search engines would access the user's definition of relevance (the search goal) and determine with certainty whether a document was relevant or irrelevant. Unfortunately, this is not yet possible and instead search engines must use surrogates or proxies for relevance. The typical search engine uses a soft conjunction of hundreds of factors [120], such as the importance of query terms and the frequency of query terms in a document, to determine the relevance of a document to a query. When these surrogate measures are adequate, the search is easy, but when they are inadequate, the search is hard.

Researchers have investigated many different relevance measures and search models [120] on relatively small datasets and these methods can make some hard searches easier. For example, using a probabilistic model brings a search engine's understanding of queries with too many terms closer to that of the user. The problem is that many of these improved models are intractable on web scale datasets (tens of billions of pages) [31] and so typical search engines must make judicious use of them in the instances where they are most helpful. For example, as of 2007 [71],

Google mostly treats queries as individual words, but also models a limited number of the most useful phrases such as "San Francisco" using a richer model.

Research on applying improved relevance measures and search models to web scale datasets is sparse, because most researchers (those outside large public search engines) do not have access to web scale datasets or the resources necessary for experiments on them. Although this thesis focuses on web search and is therefore evaluated using web scale datasets, the focus is on search tools that extend existing search engines (search extensions) such as by changing queries or by helping users refine queries and as such external access to existing search engines is sufficient, thereby avoiding the difficulties associated with web scale datasets.

1.1.2 Changing the Query

Changing the query is another method of improving a search engine's interpretation of a query and is the method used by users when refining queries. Researchers have investigated methods that attempt to improve search performance without user input by extending queries before they are processed by a search engine and have termed this type of method Automatic Query Expansion (AQE) [150].

AQE has been developed primarily in the field of information retrieval, but unfortunately the focus there has been on improving recall [181] (which has a low priority for web search). Additionally, AQE often decreases precision (which has a high priority for web search) because the terms used to expand a query often change the query's meaning — this effect is called query drift [50].

There are various methods of finding terms to extend queries such as Pseudo-Relevance Feedback [156], Thesauri Expansion [88], and Query Log Analysis [52]. While all these methods generally improve search engine performance for easy searches, the problem is that none of them gen-

erally improve performance for hard searches and sometimes the methods make hard searches even harder by decreasing performance.

Pseudo-Relevance Feedback works well on queries that already produce good results (high precision) [199]. However, for hard searches where queries have poor initial results (low precision), Pseudo-Relevance Feedback often reduces precision further due to query drift [50]. Thesauri Expansion behaves similarly to Pseudo-Relevance Feedback because the descriptively related terms such as synonyms that are used for expansion are often polysemous causing query drift. Query Log Analysis improves the most common searches (those performed by many users) as it relies on finding the commonalities between the refinements of different users. Unfortunately, hard searches are typically uncommon [19], and consequently, hard searches do not generally benefit from Query Log Analysis.

Recently, researchers observed the importance of distinguishing queries for easy searches from queries for hard searches and have posed this as an important research question [186]. One identified application is to limit query expansion methods to easy searches [199]. While this will improve performance, the benefits are marginal compared to those achieved by improving the hard searches.

This thesis investigates how to automatically distinguish queries for easy searches from queries for hard searches and develops a novel method that automatically improves search performance for hard searches by helping search engines understand the users' queries better.

1.2 Helping Users Refine Queries

Query refinement is challenging because it involves both understanding why the previous query failed and generating a new query to overcome those limitations. Typical search engines do not help users refine queries and their flat unstructured result sets hinder manual refinement, because users must sift through the irrelevant documents to determine the various

reasons their query failed.

Users cannot rely on improved search engines solving all their search problems; even with search engines that understand queries better, users will still need to refine inaccurate and imprecise queries such as ambiguous queries. However, search engines can help users refine queries and they can structure the result sets to help users quickly understand their contents.

Researchers have investigated various methods to help users refine queries, which can be classified into two categories: Interactive Query Expansion (IQE) [150] and Web Page Clustering [201]. IQE uses user input to guide the addition of extra query terms to queries, while clustering finds groups of similar documents in the result set.

1.2.1 Interactive Query Expansion

IQE methods include those used for AQE such as Pseudo-Relevance Feedback [156], Thesauri Expansion [157], and Query Log Analysis [52]. When used for IQE, they present the user with a set of different refinements, rather than automatically changing the query, but they suffer the same limitations as under AQE. A different IQE method is Relevance Feedback [150], which has the user mark which result set documents are relevant and which are irrelevant.

For easy searches that require refinement, experienced search users can usually find an effective query easily. For example, the query "jaguar" is easy to refine because the different intents are distinct — documents about animals have little in common with documents about cars — and therefore, any refinement term selected by a user is likely to be effective. However, novice users can have significant trouble as they may not understand the concept of query refinement and even experts would benefit from the time saved by a search engine automatically presenting a set of refinements.

The problem with the IQE methods is that they are quite limited in the types of search they can refine, even when only considering easy searches. Pseudo-Relevance Feedback works well when the top ranked documents are relevant (queries that usually do not require refinement), Thesauri Expansion works well when a single term is ambiguous, Query Log Analysis works if similar queries occur frequently, and Relevance Feedback works when the user can identify some relevant documents. However, even for easy searches, the top ranked documents may be irrelevant, the query may contain multiple terms, the query may not be popular, and the user might not find any relevant documents.

This thesis develops a novel IQE method that can suggest useful refinements when other methods fail and even when none of the documents retrieved by the original query are relevant.

1.2.2 Web Page Clustering

Web Page Clustering algorithms [201, 136, 122] group similar documents together into clusters and then present them to the user. Each cluster represents a different refinement and as a whole, the clustering provides the user with an overview of the different documents in the result set.

For easy searches that require refinement, the result set usually contains some relevant documents, although when buried deep in the ordered list, users may struggle to find them manually. Web page clustering helps by identifying them, grouping them, and bringing the cluster containing them to the user's attention. Even when the result set contains no relevant documents, the overview provided by the clustering helps the user quickly determine why their query failed, which forms the basis for the user to construct their own refinement.

Although web page clustering algorithms differ in implementation and performance, and many algorithms use additional factors to guide the clustering, the primary influence on how all algorithms cluster the doc-

uments is the document contents. Web page clustering is effective on easy searches because the different intents of the query align with the different document contents.

This thesis identifies the conditions under which web page clustering algorithms are effective, identifies the problems that cause them to fail, and develops new methods that improve web page clustering performance on easy searches by making clusters that are more semantically meaningful to users.

1.2.3 Hard Searches

For hard searches that require refinement, even experienced search users can struggle to find effective queries, because it can be hard to find queries that exclude the irrelevant documents without also excluding the relevant ones. For example, the query "wildlife extinction" seems innocuous, but it is hard to refine because the documents for different query intents use similar terms — documents about efforts to stop extinction share a lot in common with documents about the causes of extinction — and therefore, refinement terms selected by a user are unlikely to be effective.

The problem is that users and search engines speak different languages (treat queries differently). The typical user might expect the query "efforts to stop wildlife extinction" to find documents about efforts to stop extinction and it will, but the user may struggle to find them as the query also finds many irrelevant documents about the causes of extinction because the words "efforts" and "stop" occur on all these documents. In contrast, the typical user might not expect the query "wildlife extinction donation" to find documents about efforts to stop extinction because semantically it means something different, but it is an effective refinement because the term "donation" occurs on the relevant documents, and discriminates between the relevant and the irrelevant documents.

None of the existing IQE methods and none of the existing web page

clustering algorithms are particularly effective at finding good refinements for hard searches. The IQE methods are ineffective for the same reasons they are ineffective on easy searches and for the same reasons the related AQE methods were ineffective on hard searches. The better web page clustering algorithms assist with some hard searches, but in general, they produce refinements that are semantically meaningless to users, as the clusters do not align with the different query intents because the document contents for different intents are often very similar.

This thesis analyzes the differences between the way users understand queries and the way search engines understand them, identifies how to refine hard searches effectively, and develops novel methods that automatically construct effective query refinements for hard searches.

1.3 Major Contributions

This thesis contributes to evaluation, clustering, and query refinement in the web search domain.

While this thesis focuses on web search, no web specific features such as hyperlinks are used and therefore, its contributions are equally applicable to any text search problem, including searching email, text document repositories, and even sufficiently labelled images and videos. Some contributions, such as those that help users refine hard searches, may be even more valuable in other search domains: in the fields of law, finance, and medicine, users routinely spend significant amounts of time refining queries for hard searches because their searches are often important, valuable, or even a matter of life or death (for example, searching for the antidote to a poison swallowed by a child).

1.3.1 Evaluation

One contribution of this thesis to evaluation is to determine best practice (based on a survey of the research literature) for conducting an evaluation of web search results with respect to selecting a methodology, selecting a corpus, obtaining a query test set, choosing a baseline, and selecting performance measures.

Another contribution to evaluation is to determine the properties of good web page clustering evaluation methods and to evaluate the extent to which existing evaluation measurements meet these criteria.

The final contribution to evaluation is QC4, a new web page clustering evaluation method that improves on existing gold standard methods by using a richer ideal clustering and four new overall measurements. The insight is that the typical ideal clustering (a flat partition) is limiting, because web page clusterings are often hierarchical with overlap between clusters. QC4 is the only method that meets all the criteria of a good web page clustering evaluation method. In particular, the measurements allow the fair comparison of algorithms that construct clusterings with vastly different characteristics (cluster granularity: coarse or fine, clustering structure: hierarchical or flat, disjoint or overlapping, and cluster size: large or small).

1.3.2 Web Page Clustering

One contribution of this thesis to web page clustering is to determine the conditions under which web page clustering algorithms are effective and the problems that cause them to fail.

Another contribution to web page clustering is QDC (Query Directed Clustering), a new clustering algorithm with five key innovations that enable QDC to effectively summarize the result sets of ambiguous queries for easy searches and help users refine their queries onto one of the query's interpretations. The key insight is that the query itself provides a lot of information about how to cluster the documents in a way that is meaningful

to users.

QDC is efficient and generates better clusters by pruning ambiguous and low value words before clustering based on calculations of the semantic distance of words from the query. Following the pruning, QDC uses an intelligent merge-split clustering algorithm to construct clusters that are semantically meaningful by creating clusters that are similar in both intension (description) and extension (documents). The splitting phase also solves the problem of cluster chaining (cluster drift). QDC then chooses the correct number of clusters to show the user and finally, ranks the documents according to cluster relevance.

1.3.3 Query Refinement

One contribution of this thesis to query refinement is to determine what makes searches hard and to find a method that automatically distinguishes queries for easy searches from queries for hard searches. The Query Aspect Approach presented by this thesis distinguishes them by identifying queries containing multiple aspects[5] where some aspects are underrepresented in the result set (a very common failure for queries for hard searches).

The Query Aspect Approach identifies query aspects by analyzing the order of words in the query and determines if the result set adequately represents the identified aspects by modelling the vocabulary associated with each aspect. This approach is shown to be very powerful and with small variations, it addresses many problems, including underrepresented aspects, hard ambiguity, and keyword overload (too many terms in the query).

The principal insight behind the Query Aspect Approach is that documents that represent an aspect should contain a good proportion of that

[5]Aspects are the distinct terms from the query that correspond to the different parts of the user's search goal.

aspect's vocabulary. The approach is powerful because it better approximates the descriptive semantics users associate with query terms and it provides an improved measure of relevance that considers document focus. Using this relevance measure, methods can automatically determine which refinement provides the highest quality results.

Another contribution to query refinement is AbraQ, a novel automatic query expansion method that uses the Query Aspect Approach to improve search performance for many hard searches without user involvement by improving the way search engines understand the user's queries. When the result set underrepresents any query aspects, AbraQ finds additional terms related to the underrepresented aspect and tests how well each term resolves the problem by adding them (one at a time) to the query and reusing the Query Aspect Approach to evaluate their effectiveness; AbraQ then applies the best refinement automatically.

The final contribution of the thesis to query refinement is Qasp, an interactive query expansion method that builds on AbraQ to help users refine ambiguous queries for hard searches. Qasp expands the powerful Query Aspect Approach further by using a simple clustering algorithm to identify the different interpretations in queries with multiple aspects. Unlike other interactive query expansion approaches, Qasp groups similar refinements together and relates them to the aspects they affect. This helps users understand the effects of refinements before applying them and enables users to choose a helpful refinement more frequently.

1.4 Thesis Outline and Publications

While conducting the research presented in this thesis, I wrote several fully refereed publications, which have been extended to form the basis of this thesis. This section outlines the structure of the remainder of the thesis and shows how each chapter relates to prior publications.

Chapter 2 provides background information related to web search, search

engines, search refinement, and search extensions and sets the context for the rest of the thesis. It also analyzes how users search the web, the different kinds of problematic search they encounter, why these searches are problematic for typical search engines, and how the existing search engine extensions fail to address them. Related publications include

- **Daniel Crabtree**. "Enhancing Web Search through Query Expansion". *Encyclopedia of Data Warehousing and Mining, Second Edition*, Chapter 116, pages 752–757, Information Science Reference (2008).

Chapter 3 identifies the best practice for conducting an evaluation of web search results and explains the decisions behind the web search evaluation choices made in this thesis.

Chapter 4 determines the properties of good web page clustering evaluation methods and evaluates the extent to which existing evaluation measurements meet these criteria. Then it presents and evaluates QC4, a new web page clustering evaluation method. Related publications include

- **Daniel Crabtree**, Peter Andreae, and Xiaoying Gao. "QC4 - A Clustering Evaluation Method". In *Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference, PAKDD07*, pages 59–70, Springer-Verlag (2007).

- **Daniel Crabtree**, Xiaoying Gao, and Peter Andreae. "Standardized Evaluation Method for Web Clustering Results". In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI05*, pages 280–283, IEEE Computer Society (2005).

Chapter 5 determines the conditions under which web page clustering algorithms are effective and the problems that cause them to fail. Then it presents and evaluates QDC, Query Directed Clustering, a new web page clustering algorithm. Related publications include

- **Daniel Crabtree**, Peter Andreae, and Xiaoying Gao. "Query Directed Web Page Clustering". In *The 2006 IEEE/WIC/ACM Interna-*

tional Conference on Web Intelligence, WI06, pages 202–210, IEEE Computer Society (2006).

- **Daniel Crabtree**, Xiaoying Gao, and Peter Andreae. "Improving Web Clustering by Cluster Selection". In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI05*, pages 172–178, IEEE Computer Society (2005).[6]

Chapter 6 determines what makes searches hard and then presents the Query Aspect Approach, a method of distinguishing queries for easy searches from queries for hard searches. Then it presents and evaluates AbraQ, a method that automatically improves queries for many hard searches and Qasp, a method that helps users refine ambiguous queries for hard searches. Finally, it identifies other uses of the Query Aspect Approach including one that addresses keyword overload in queries. Related publications include

- **Daniel Crabtree**, Peter Andreae, and Xiaoying Gao. "Exploiting Underrepresented Query Aspects for Automatic Query Expansion". In *The Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD07*, pages 191–200, ACM (2007).

- **Daniel Crabtree**, Peter Andreae, and Xiaoying Gao. "Understanding Query Aspects with applications to Interactive Query Expansion". In *The 2007 IEEE/WIC/ACM International Conference on Web Intelligence, WI07*, pages 691–695, IEEE Computer Society (2007).

Chapter 7 presents the conclusions and potential directions for future research.

[6]This paper published research conducted before this thesis on ESTC. It is included here as one part of QDC extends and improves ESTC's cluster selection algorithm.

Chapter 2

Background

Web search originates from the well-established field of information retrieval [179, 107] and both share concepts, problems, and algorithms. The first generation (1995–1997) of web search engines mirrored methods from the information retrieval literature [21], but subsequently web search engines have evolved to consider web specific characteristics and the field has branched off to become a distinct and flourishing area of research. Because this thesis focuses on web search, it predominantly references the web search literature, but where appropriate it also refers to the information retrieval literature.

This chapter sets the context for the remainder of this thesis and provides background information on how users search the web, the problems users encounter, how typical search engines operate, the limitations of current search engines, and efforts to address those limitations.

2.1 Search Goals: Why Users Search

To help users search the web more effectively, it is important to know the reasons users search, because there is little value in improving searches of no interest to users and marginal value in improving searches that are already easy for users. Users search the web for various reasons and the

type of search goal can affect both the type of results expected from search engines and the search difficulty.

Broder [21] classified search goals into three classes: navigational, informational, and transactional. The intent of navigational goals is to find a particular site (for example, find the website of Victoria University). The intent of informational goals is to find information about some topic (for example, find out when the dinosaurs became extinct). The intent of transactional goals is to perform some action (for example, to purchase a book).

By manually analyzing query logs, Rose and Levinson [149] found that the most common searches are informational (60 – 65%), the least common navigational (10 – 15%), and the remainder transactional (20 – 25%). Jansen et al. [95] found similar results[1] on more recent query logs using an automatic approach.

Informational and transactional goals are similar in that there are usually many relevant documents that satisfy these goals. In contrast, there is usually only a single relevant document for navigational goals. Consequently, search engines must treat navigational goals differently from other goals and must find the right document to satisfy the user's search goal.

First generation web search engines were very similar to the methods used in classic IR, which assume searches are informational, and consequently, navigational goals were hard searches [21]. Research on navigational goals such as the personal home page finding task [47] helped identify that hyperlink analysis methods such as Google's PageRank [20] were effective for navigational goals and since the early 2000s navigational goals have been easy searches [159].

As current search engines solve most navigational goals, the experiments in this thesis did not include any navigational goals and this thesis does not consider them further. However, the methods in this thesis are still applicable to navigational goals. For example, Query Directed Clus-

[1] after accounting for misclassification errors

tering (chapter 5) helps users refine queries for navigational goals that are ambiguous.

2.2 Search Statistics: How Users Search

To help users search the web more effectively, it is important to know how users search, so that evaluations are realistic and methods do not make unrealistic assumptions about users.

Researchers have conducted user studies and analyzed query logs to identify how users search and their level of success [10, 94, 158]. The most recent publicly available query logs are the AOL query logs released in 2006 by AOL [139], which contain 20 million queries from 650,000 users using a Google powered search engine. Due to the massive backlash against AOL for violating privacy [69], major search engines have not publicly released any more recent data sets.

Typically, users use very short queries with fewer than three keywords. Over the first half decade of the web, there was an increase in session interactivity, query length, and range of information needs [94]: evidence that users were becoming more proficient at search. Initially, comprehensive query formulation was rare, query modification atypical, and queries were particularly short [10]; in 1998, the mean query length was 2.35 [158] and just 22.4% of queries were refined [94]. Since 2002, search engine usage has stabilized and query refinement has become more common; in 2002, the mean query length was 2.92 and 52.5% of queries were refined [94] and since 2002, the mean query length has stayed around 3.[2]

Users rarely look beyond the first result page. The fraction of queries where the user viewed just the first result page was 85.2% in 1998 [158], 72.8% in 2002 [94], 81% in 2003 [12], 79% in 2004/2005 [12], and 78.3% in 2006 [139].

[2] 2.7 in 2004/2005 [12], 2.75 in 2006 (from my analysis of the AOL query logs), and 2.98 in 2007 [143]

The use of advanced Boolean query operators such as +, -, *and*, *or*, and *not* is decreasing. The number of queries containing Boolean operators was 6% in 2002 [94], 2% in 2004/2005 [12], and 0.23% in 2006.[3] The phrase operator " " is more widely used with 14% of queries using it in 2002 [94]. None of the queries in the AOL query logs contain the phrase operator, indicating that it was removed when the queries were transformed.[4]

The experiments in this thesis use short queries without advanced Boolean operators and use evaluation measures that focus on the quality of the first result page.

2.3 Hard Searches

To improve relevancy for hard searches and to help users refine queries for hard searches it is useful to know which searches are hard. This section identifies the frequency of hard searches and their relative difficulty, the types of search that challenge users, the types of query that are difficult for users to refine, the factors that affect search difficulty, and finally, how experts refine hard searches.

2.3.1 Frequency and Relative Difficulty

On current search engines, many searches are easy for users, but the few hard ones are significantly harder and consume all their time. 72% of searches are solved in under 5 minutes with 48% requiring no refinement [94] (my analysis of the more recent AOL query logs found search engines have improved and it is now 79% and 56% respectively). However, users spend the majority of their time refining the relatively few hard queries: the hardest 20% of searches consume 80% of the user effort, as measured

[3] From my analysis of the AOL query logs.

[4] The queries are all in lower case, so they were probably pre-processed or transformed in some way.

by time spent searching [94] (my analysis of the more recent AOL query logs found this to be unchanged).

My analysis of the AOL query logs used the session detection algorithm outlined by He and Göker [85] with a session boundary of 15 minutes. There is a tradeoff between a short and a long session boundary: a short boundary may allocate related queries to different sessions, while a long boundary may allocate unrelated queries to the same session. A 15 minute boundary is reasonable because He and Göker [85] identified the optimal range as being 10 to 15 minutes and a more recent study [19] of the AOL query logs used 20 minutes.

One limitation of these studies and the session detection algorithm is that they do not indicate whether the user's search was successful or not. This limitation may lead to an underestimation of the difficulty: users might abandon searches quickly because they were unable or unwilling to refine their query or avoid queries entirely due to the perceived difficulty of a search. Alternatively, this limitation may lead to an overestimation of the difficulty: users might continue searching even if their search was successful because they want more documents that are relevant or they have multiple search goals that relate to a larger goal such as planning an overseas holiday.

In 2006 [91], while 82% (up from 68% in 2002) of users usually try refining failed queries, 3% (unchanged from 2002) simply give up when the first query fails. This suggests that users might abandon hard searches quickly (the studies underrepresent the problem of hard searches) and furthermore, it suggests that users may be abandoning hard searches even after devoting significant time to them (the user's hard search goals are not being solved).

To test the accuracy of the session detection algorithm for hard searches, I manually classified a sample of 200 very hard searches from the AOL query logs. The sample was randomly selected from all searches over 30 minutes (these represent the hardest 2% of searches and 17% of user ef-

fort), as opposed to all hard searches, because the longer sessions are the most susceptible to type 3 errors (unrelated searches are allocated into the same session) [85].

87%±5% (95% confidence interval) of the searches were correctly classified. Searches classified as correct include those related to a single search goal (A) and those related to multiple search goals where the majority of the time was consumed by individual search goals each consuming more than 30 minutes of user effort (B). Searches classified as incorrect include those related to multiple search goals where one hard search (usually 10 – 25 minutes) preceded or followed some easy searches (C) and those related to multiple easy search goals (D). The majority of the correct classifications were of type A rather than of type B and the majority of the incorrect classifications were of type C rather than of type D.

On balance, the studies give a reasonable indication of the frequency (20% of queries) and relative difficulty (80% of user effort) of hard searches and show that hard search is an important direction for research on improving search.

2.3.2 The User's Experience

To gain insight into the problems experienced by users, I conducted an informal study of friends, family, and colleagues (the users) during 2005 and asked them to submit[5] details about any hard searches they encountered. The users understood a hard search to be any for which they could not easily find relevant results. Most users were surprised to learn how often they encountered hard searches and how frequently their searches failed.[6]

There were three types of user: novice, typical, and experts. The group mostly consisted of typical users. The novice users rarely used computers or the internet and generally felt search was difficult and believed they

[5]anonymously via a web interface

[6]The exception was novice users who admitted upfront that they found most searches hard.

lacked understanding of how to use search engines properly — they generally stated upfront that most searches were hard and in practice had problems with many searches that would be easy for typical users. The typical users were confident and generally felt search was easy and falsely believed they knew how to make best use of search engines — they generally stated upfront that they could always find what they want, but in practice they had problems with a lot of hard searches and this surprised them. The expert users knew how search engines operate and had tricks to get the best from them — they generally admitted they encountered some hard searches, but that they were rare "1 out of 10 searches".

Users conduct searches for a variety of reasons and one important distinction is whether the reason for searching is important or unimportant. Most searches are unimportant (e.g. Who was the director of E.T.?) and when users do not immediately find relevant results (e.g. Who was the actress in the film from the 1980's that had a dog in it?) they quickly abandon the search and quickly forget the search failed. Other searches are important (e.g. What should I do if my child has swallowed poison?) and when users do not find relevant results they either keep searching or become dissatisfied.

Most users only reported the important searches they found hard — the searches to which they devoted a lot of time. On retrospect, most users (once becoming more aware of their own search habits) admitted they often encountered unimportant searches that were hard and that previously they overlooked them.

The important searches the users found hard included a wide range of topics such as medical information (e.g. What condition do I have?), information about an expensive or long-lasting purchase (e.g. What are the problems with this car?), technical information (e.g. What are this microchip's power requirements?), and research (e.g. Who was my great great great grandfather's wife?).

The hard searches from the AOL query logs (those over 30 minutes)

were predominantly important searches and featured a similar range of topics to those in the informal study — suggesting users generally have similar experiences to those in the study and that users probably were experiencing difficulties refining many of the hard searches identified in the AOL query logs. Among others, the topics of hard searches in the AOL query logs include medical information, information about significant purchases (e.g. What kind of wood is best for a new deck?), technical information (e.g. How to use a specialized ruler? — this user spent 90 minutes seeking this information), tax and legal information, specialized products (e.g. Where can I buy oversized shoes? — this user also spent 90 minutes seeking this information), financial or real estate information, and research (e.g. Where is my long lost friend now?).

The following sub-sections (2.3.3 and 2.3.4) generalize the types of problem and the hard searches encountered by the users in the informal study and the hard searches from the AOL query logs.

2.3.3 Problematic Queries

While the hard searches conducted by users span an enormous range of search goals, the queries users construct to communicate these goals exhibit a small range of problems. This section describes eight different situations where queries can have problems: ambiguous keywords, missing aspects, broad search goals, narrow search goals, multiple aspects, imprecise keywords, higher order constraints, and natural language.

2.3.3.1 Ambiguous Keywords

Queries with ambiguous keywords have multiple interpretations and find irrelevant documents related to other search goals. For example, "Jaguar" may refer to cars, animals, or a Macintosh operating system, among others.

Refinement generally involves adding keywords to specify the rele-

vant interpretation. Some ambiguous queries such as "Jaguar" are easy for most users to refine (only novice users experience any difficulty refining these), but could still be made even easier to refine. Other ambiguous queries such as "mobile phone usage cuba", which may refer to the rules surrounding their use or to the number of people using them, are hard for even experts to refine because obvious refinements such as "number of mobile phones in cuba" are also problematic.[7]

2.3.3.2 Missing Aspects

Queries with missing aspects do not express some component of the user's search goal and find irrelevant documents related to other search goals. For example, "Jaguar car" is problematic if the user wanted to find Jaguar car clubs, because the query does not express the club component and it may find irrelevant documents such as dealers selling cars.

Refinement generally involves adding keywords that correspond to the missing aspects. The problem is that simply adding keywords can sometimes create a query with different problems such as a narrow query.

2.3.3.3 Broad Search Goals

Queries for broad search goals often contain just a few keywords and find irrelevant documents related to a wide range of narrower search goals. For example, "car" should be a good query for a user wanting general information about cars, but this query mostly finds places to buy cars (a narrower version of the user's goal).

Refinement is often difficult because there are no obvious keywords to add and terms like "information" are not helpful as they are widely used — "car information" finds car safety information and different places to

[7]The keyword "number" is ambiguous (it may refer to phone numbers such as 911), the query has multiple aspects, and it uses natural language (to signify the relationship between number and mobile phones).

buy cars (those suggesting you ask them "if you need any more information"). Experts sometimes use the Boolean operator *not* to exclude irrelevant documents, but this can lead to the exclusion of relevant documents too.

2.3.3.4 Ambiguous Queries

Queries with ambiguous keywords, queries with missing aspects, and queries for broad search goals have different causes, but share a common refinement strategy — the query needs narrowing to exclude irrelevant documents that relate to other search goals. Due to this similarity, the remainder of this thesis uses the term ambiguous queries to encompass all three types of problematic query and uses ambiguous keywords, missing aspects, and broad queries to refer to the individual types where necessary. Chapters 5 and 6 introduce methods that help users refine ambiguous queries.

2.3.3.5 Narrow Search Goals

Queries for narrow search goals often contain too many keywords and may find no results because no documents contain all the keywords. For example, "reflection TargetInvocationException [protected method] ashx cassini"[8] finds no results, despite the existence of relevant documents and even though none of the query terms is redundant or irrelevant to the user's search goal (to find information about a particular programming error), because no documents contain all the selected keywords.

Refinement generally involves removing some keywords, but sometimes this approach is unsuccessful because it can lead to a query with missing aspects. Finding the right balance between aspect coverage and the number of keywords is troublesome and sometimes impossible. The solution is to create an entirely new query, but typical users often just cre-

[8]The square brackets around [protected method] indicate this is a phrase.

ate another problematic query with too many keywords, missing aspects, or a different issue entirely.

Longer queries contain more information about the user's search goal and should produce better results than short queries. Chapter 6 introduces the Query Aspect Approach and outlines how this approach could be used to take advantage of the additional information in queries with too many keywords to automatically construct simpler queries.

2.3.3.6 Multiple Aspects

Queries with multiple aspects sometimes have underrepresented aspects that fail to narrow the range of retrieved documents. For example, "tunnel disaster" finds documents related to all disasters in tunnels including both those during construction and those during subsequent use. "tunnel construction disaster" should be a good refinement for finding documents about disasters that occurred during construction, but it fails to narrow the results because both relevant and irrelevant documents are likely to contain the term "construction". For example, irrelevant documents may contain the phrase "Construction was completed in".

Typical users find it challenging to refine queries with underrepresented aspects, as they generally do not understand the problem. Refinement of these queries is time consuming, even for experts, because trial and error is normally necessary to discover effective refinements. The problem is that the difference between relevant and irrelevant documents is subtle, because all the retrieved documents are within the vicinity of the query's general topic area.

Queries contain information about the user's search goal and not merely information about what terms should occur in relevant documents. Chapter 6 introduces a method that automatically modifies queries with underrepresented aspects to improve performance.

2.3.3.7 Imprecise Keywords

Queries with imprecise keywords do not express the user's search goal using the most accurate terms. For example, if a user wanted information about heart attacks, the query "chest pain" will find less relevant information than "acute myocardial infarction".

Refinement can be easy or hard. Some initial queries such as "chest pain" put the user on a path to discovering more precise terms, but this is not true of all queries with imprecise keywords. For example, "query refinement" is a reasonable starting point when seeking research on improving queries, but it finds irrelevant documents related to using search engines, which do not help the user discover the more precise phrase "query expansion", which is used within the research literature.

While helping users learn keywords that are more precise is beyond the scope of this thesis, the methods in the thesis that help users refine ambiguous queries also help users understand the result set, which can help users refine queries with imprecise keywords more quickly.

2.3.3.8 Higher Order Constraints

Queries with higher order constraints try to specify meta-characteristics of relevant documents. For example, "maths for 9 year olds" seeks documents written for a particular audience. Users find relevant documents for these queries when authors intentionally satisfied the constraint and consequently included detail of the constraint in the document. However, some authors may not explicitly detail the constraint or even if they do, might use different terminology. For example, an author may use "3rd grade" instead of "9 year olds". Generally, documents satisfy higher order constraints unintentionally; consequently, many relevant documents will not detail the constraint and queries with higher order constraints will fail. An example would be finding articles on a specific topic with a particular political bias such as left-wing or right-wing.

2.3.3.9 Natural Language

Queries using natural language often contain questions or specify relationships between query terms. For example, "What is the defining characteristic of the dark ages?" specifies a question and "hotels between $100 and $150 in Sydney" contains a relationship between different query terms. As with higher order constraints, when document authors have intentionally sought to answer a question or to satisfy a relationship, their documents will explicitly detail the question or relationship and therefore, queries using natural language will succeed for common questions and relationships. However, most documents do not explicitly answer questions or satisfy relationships and natural language queries will fail.

2.3.3.10 Complex Queries

Queries with higher order constraints and queries using natural language are challenging to refine, because search engines do not support them. Refinement generally involves creating a new query that does not use higher order constraints or natural language, but sometimes that is impossible.

Search engine support for complex queries is an open research question, but is beyond the scope of this thesis. However, the methods in this thesis may inadvertently help users refine these complex queries or may help users indirectly — by enabling the user to find effective keyword based queries for their search goal and thereby avoid the need to use a complex query.

2.3.4 Factors affecting Search Difficulty

This section describes six factors that affect the difficulty of a particular search, including three extrinsic factors (search experience, document complexity, and popularity) and three intrinsic factors (domain knowledge, corpus knowledge, and search specificity). These factors are orthogonal to the problematic queries discussed in section 2.3.3 and their impact

on a particular search depends on the corpus, the user, and the search engine as shown in table 2.1.

Table 2.1: Six factors affect search difficulty and their influence depends on the user, the corpus, and the search engine.

		Corpus	User	Search Engine
Extrinsic Factors	Search Experience		■	■
	Document Complexity	■		■
	Popularity	■		■
Intrinsic Factors	Domain Knowledge		■	
	Corpus Knowledge	■	■	
	Search Specificity	■		

Extrinsic factors depend on the search engine, while intrinsic factors are independent of the search engine. With improvements, search engines can mitigate extrinsic factors. In contrast, search engines cannot avoid intrinsically hard searches, but they can help users refine their queries to overcome intrinsic factors. Unfortunately, current search engines neither mitigate extrinsic factors nor assist users with intrinsic factors.

2.3.4.1 Search Experience

Users who are inexperienced with search and the query model used by search engines will find it harder to construct effective queries than experienced users. Any user may construct queries that use too many keywords (for narrow search goals), use too few keywords (for broad search goals), directly specify higher order constraints, or use natural language. However, experienced users will quickly identify the problem and try refining their query, while inexperienced users may struggle, because they may not

understand why their query failed and novice users may not even know that refining their query might help.

2.3.4.2 Document Complexity

It is harder to construct effective queries for searches that seek long complex documents than it is for short simple documents. Long documents contain a wider range of terms and therefore, are more likely to contain ancillary terms that do not reflect the core focus of the document.

When seeking long complex documents, queries with multiple aspects are more likely to have underrepresented aspects, and queries that use imprecise keywords or have narrow search goals are more likely to retrieve irrelevant documents.

2.3.4.3 Popularity

It is harder to construct effective queries for rare information (few documents) than popular information (many documents). For example, "install windows" finds information about the operating system, but nothing about the windows that contain glass. Many different authors write popular information in many different ways, and consequently, there is a higher probability that relevant documents contain the query's keywords. As information becomes rarer, fewer authors write about it and consequently there are fewer effective queries.

When seeking rare information, ambiguous queries and queries that use imprecise keywords are more likely to find irrelevant documents, narrow search goals are more likely to find no documents, and queries with multiple aspects are more likely to have underrepresented aspects.

2.3.4.4 Domain Knowledge

Users who are domain novices will find it harder to construct effective queries than domain experts. Domain experts write the most relevant in-

formation using domain specific vocabulary; novices lacking this vocabulary are more likely to create queries that use imprecise keywords.

2.3.4.5 Corpus Knowledge

Users who are inexperienced at searching for information on a topic will find it harder to construct effective queries than users that are experienced. As users search within a topic they learn which irrelevant topics in the corpus share vocabulary with the relevant topic and which keywords are most effective at discriminating between the relevant and the irrelevant topics. Experience at searching for a topic helps users avoid ambiguous queries and refine them more easily; experience also helps users refine queries for narrow search goals and queries with underrepresented aspects more easily.

2.3.4.6 Search Specificity

It is harder to construct effective queries for general information than specific information.[9] For example, if a user wanted to find the power of the engine in a radio controlled F16 fighter jet, the query "Evolution .52NX engine displacement" that specifies a specific engine will be more effective than the more general query "radio controlled F16 fighter jet engine displacement". Note the distinction between specific (used here) and narrow (used in section 2.3.3.5) — specific relates to a single aspect (search goal component) whereas narrow relates to multiple aspects.

Specific information is easier to find because the keywords that describe specific information are generally less ambiguous and consequently some aspects (as in the earlier F16 example) and some higher order constraints (as in the following example) are often implicit and therefore unnecessary. For example, if a user wanted to find information about teach-

[9] Assuming the user has the required domain knowledge to express their search goal using a precise query.

ing mathematics to children, the query "teaching fractions" and other similarly specific queries are more effective than the more general query "teaching mathematics", because the more general query might retrieve irrelevant documents about teaching mathematics to secondary and tertiary students.

2.3.5 Refinement Strategies

Query refinement is inevitably trial and error, but there are numerous strategies to improve the probability of finding an effective query. These strategies typically involve examining retrieved documents, identifying and categorizing irrelevant documents, determining why queries failed, and determining keywords that discriminate between relevant and irrelevant documents. Novice users are generally unaware of these strategies and predominantly use trial and error, whereas typical users often make use of more systematic approaches when trial and error fails.

For complex queries, users often employ a multi-step approach. Instead of trying to find relevant documents directly, users try to find documents that will link to relevant documents. For example, someone looking for hotels within a certain price range may look for travel sites or destination guides that they expect would provide this information. From those sites, the user can find links to specific hotels (the relevant documents) that meet their criteria.

Unfortunately, for many hard searches, none of these approaches is successful. My study found that search experts have another refinement strategy (termed refinement by semantically orthogonal keywords in this thesis) for hard searches. Instead of constructing queries that describe what they want to find, experts construct queries that contain the terms they expect to find on relevant documents. For example, the term "donation" in the "wildlife extinction donation" query from section 1.2.3 is irrelevant to the user's search goal of finding efforts to stop extinction, but it is

effective since many efforts to stop extinction seek donations. Norvig [134] also identified this strategy and suggests refining the "mobile phone usage cuba" query from section 2.3.3.1 using the terms "worldwide" and "table" because there probably exist relevant documents that contain a big table listing usage around the world.

To humans, queries with semantically orthogonal keywords can look semantically meaningless and at least semantically deceptive, because they often have a completely different meaning from both the descriptive query terms and the user's search goal. However, since the semantically orthogonal keywords usually co-occur with descriptive terms on relevant documents, they are often very effective and help search engines discriminate between relevant and irrelevant documents. The disconnect between the queries that are meaningful to users and those that work well with search engines is due to the differences between the search models used by humans and search engines (section 2.5.1 discusses these differences).

The problem with refinement by semantically orthogonal keywords is that it requires a lot of user effort and it requires substantial domain knowledge that users may not possess. Chapter 6 presents a method that automatically identifies semantically orthogonal keywords and then uses them to refine queries with underrepresented aspects and to suggest refinements to users for ambiguous queries.

2.4 Search Engines

2.4.1 Types

There are several different approaches to web search including unassisted keyword search (e.g. Google), assisted keyword search (e.g. clustering), query by example (description, snippet, or page), and directory based search (e.g. Open Directory Project and Yahoo[10]) [211, 56]. Unassisted

[10] until 2005

keyword search involves entering keywords and reviewing a list of documents that contain the query's keywords. Assisted keyword searches are similar to unassisted keyword searches, but the search system helps the user refine their query by suggesting additional keywords, query refinements, or by organizing the results. Query by example searches are similar to unassisted keyword searches, but instead of taking keywords as input, they take a description, snippet, or relevant document, and find similar documents. Directory based search involves navigating a human constructed hierarchical categorization of documents organized by topic.

Query by example is most suited to finding additional documents once the user has already discovered some relevant documents. Bruza et al. [22] showed that directory based search is ineffective: it does not offer increased relevance and takes longer than keyword search. This thesis focuses on improving unassisted and assisted keyword search systems.

2.4.2 Components

A search engine can be broken down into three components: the crawler, the indexer, and the retriever [20]. The crawler recursively navigates the web, downloading documents, and identifying new URLs to visit from those previously visited. Next, the indexer parses and cleans the documents, turning them into an index that provides an efficient lookup for the chosen document model. Finally, given a query, the retrieval system will use the index to find documents that contain the query terms and rank them according to the chosen relevancy-ranking algorithm.

This thesis presents algorithms that sit atop existing search engines (search extensions), taking the query and the output of the underlying search engine as input. Nonetheless, some details of the underlying search engine are still relevant. For example, many search extensions clean and index the retrieved documents using approaches similar to those used by typical search engines. The following sub-sections provide relevant back-

ground information on the three search engine components.

2.4.3 Document Cleaning and Pre-processing

Preparing documents for indexing typically involves three pre-processing steps: tokenization, stop word removal, and stemming [150]. The extent to which search extensions implement these steps varies widely with some skipping entire steps.

Web pages are messy, filled with HTML tags, punctuation, and other superfluous information such as comments. Tokenization is the process or removing them to get the raw text and most approaches use tokenization, although some search extensions [8] do make use of the additional information and structure.

Another superfluous element, when natural language analysis is not required, is stop words, which are very common and uninformative words such as "the", "it", and "on". Stop word removal is the process of removing stop words or replacing them with a placeholder, the latter preserves word order and enables phrase queries; stop word removal reduces the index size and improves performance, although it can cause problems with some queries, for example, "to be or not to be" [31].

For many retrieval algorithms and search extensions, performance improves when words with the same semantic intent are treated identically. The most common approach to achieve this is stemming, typically using the Porter stemming algorithm [141], which reduces words to their root form, for example, "dogs" becomes "dog". Generally, methods use only light stemming to reduce the likelihood of changing the semantic intent of terms. However, when words are polysemous even light stemming can be problematic. For example, sock and SOCKS share the same root, but refer to footwear and a network protocol respectively. While this problem somewhat dissuades the use of stemming in web search [31], stemming often works well in search extensions that operate over a result set because

the restricted scope filters many polysemous terms.

2.4.4 Document Models

There are many document representations; the three most common models differ by their mathematical basis: the Boolean model, the vector space model, and the probabilistic model. Richer models generally improve result quality at the cost of performance by resolving incorrect assumptions of the simpler document models. For instance, the Extended Boolean model provides adaptive strictness for the query operators (AND and OR) [153] and Latent Semantic approaches account for term inter-relatedness [54].

Processing web scale datasets (tens of billions of pages, with many additions, updates, and deletions everyday) requires efficient indexing algorithms with complexity proportional to the size of the corpus. The richer models, particularly those incorporating term inter-relatedness are computationally intensive: latent semantic indexing uses the optimal rank-r approximation of the full Singular Value Decomposition matrix and computing this has complexity $O(dtr)$, where d is the number of documents, t is the number of distinct terms, and r is the rank of the resulting model [18]. While the probabilistic model and other more advanced models are frequently used in information retrieval on smaller corpora (such as those used in TREC), applying these to web scale datasets is still intractable [31]. Therefore, current search engines still use the simpler models: as of 2007 [71], Google still uses a vector space model[11] based on individual words and the inverted index [31, 53], much as it was originally 10 years ago [20].

The Boolean model is the simplest: it models documents as sets of terms (words or phrases). The vector space model [212, 166, 155, 211, 57] is more sophisticated: it models documents as vectors of term weights. Var-

[11] Albeit with many non-general enhancements, such as modelling a limited number of phrases such as "San Francisco", perhaps using one of the richer models.

ious weighting algorithms exist [120]; in the classical vector space model, the weights are given by tf-idf (Term Frequency • Inverse Document Frequency) [145, 86], which considers local and global document properties [152]. The probabilistic model also represents documents as vectors, but differs by using a probability based retrieval mechanism. The methods presented in this thesis use the Boolean and vector space models.

Term Frequency is the number of times the term occurs in a document, normalized to account for the length of the document. Document Frequency is the number of documents containing the term. Inverse Document Frequency is the logarithm of the result of dividing the number of documents by the Document Frequency of the term in the documents. Typically, Term Frequency is a local document property that relates to an individual document, while Document Frequency and Inverse Document Frequency are global document properties that relate to the corpus. However, they find uses in other contexts where the reverse is true. For example, the Document Frequency of a term in the first N pages of a result set is a local document property.

2.4.5 Query Interpretation and Search Indexes

Each document model has its own retrieval mechanism. The Boolean model [179, 150] performs an exact match, finding the set of documents that contain all the terms in the query. The vector space [120] and probabilistic models [120, 179, 150] perform a best match, finding a ranked list of documents that are most similar to the query. In best match systems, any document with at least one query term has a non-zero score; however, for efficiency, current search engines only return documents containing all query terms.

The efficiency of retrieval depends on the available index. For the vector space model, indexing constructs an inverted index that maps terms to documents, ideal for efficient retrieval. Search engine indexes also store

the position of each term; this enables the reconstruction of documents from the index and facilitates phrase queries and proximity queries [31]. Phrase queries find documents containing a particular sequence of words such as "neural network" and proximity queries find documents that contain words within a certain distance of each other.

Some search extensions enumerate many phrases and require an efficient index that maps all phrases less than a certain length to documents. The inverted index is too slow for that, but fortunately, for the small result set segments analyzed by search extensions, the suffix tree model provides an effective index [212].

2.4.6 Relevancy Ranking

Relevancy ranking or scoring is the most important part of a search engine, because it determines the output seen by users. At its simplest, relevancy ranking uses a similarity function with two inputs (the corpus of documents and the query) and the output is the documents sorted in descending order of similarity (or score), where the similarity is measured between the document term vector and the query term vector.

Researchers have tried a wide range of similarity functions and in fact, most clustering similarity functions are suitable: at its core, relevancy ranking is separating documents into two clusters, called relevant and irrelevant, with the query providing an exemplar of a relevant document. Perhaps the most widely used similarity function for the vector space model is Cosine similarity ($cos(d, q)$) [120, 150, 14].

$$cos(d, q) = \frac{d \bullet q}{|d||q|}$$

where d is the document term vector and q is the query term vector. Cosine similarity is superior to using the magnitude of the vector difference because it normalizes document length [120].

For web search, using only Cosine similarity to rank the documents will produce a poor ranking, because it is typical for millions of documents

of widely varying quality to contain all the query terms and additionally, spammers and others such as those with commercial interests are trying to game the systems to get the top rankings.

Current search engines go well beyond a single ranking measure and most likely use a soft conjunction of hundreds of features[120]. Finding the right combination of features is near impossible for humans; one solution is to provide labelled examples of relevant and irrelevant documents for various queries, thus creating a 2-class classification task that standard machine-learning algorithms can solve [120].

There are two kinds of features: static features and dynamic features. Static features are fixed[12] and independent of either the query (most common) or the corpus (less common). For example, PageRank [20, 58] and the age of the domain hosting a document are independent of the query, while a rule that queries for addresses should return links to Google maps is independent of the corpus. Dynamic features change with every query and generally depend on both the query and the corpus, for example, similarity scores such as Cosine similarity are dynamic features. Other dynamic features include proximity or position of query terms, font, capitalization, and location (title, URL, heading, etc.) [20].

2.5 Limitations of Typical Search Engines

Currently, typical search engines have a number of limitations that make some searches harder than necessary. Queries with narrow search goals, broad search goals, multiple aspects, higher order constraints, or natural language are only problematic because of search engine limitations, while queries with ambiguous keywords, missing aspects, or imprecise keywords are more problematic than necessary because of search engine limitations.

This section describes three limitations of typical search engines. The

[12] although they will be recomputed and may change periodically

first limitation is that users and search engines have different query models — they interpret queries differently. The second limitation is that search engines do not help users to refine queries. The final limitation is that search engines use popularity as a proxy for authority.

2.5.1 Different Query Models

Users and search engines typically have different models of how to interpret queries; consequently, the documents retrieved by search engines in response to a query sometimes do not match the user's expectations. Users naturally consider the semantics, presence, absence, and order of keywords to be significant, and expect understanding of constraints and natural language. On the other hand, typical search engines merely retrieve all documents that contain all query terms.

To users, the semantics of the query terms are important. Users expect the retrieved documents to contain information related semantically to the query terms, but not necessarily the query terms themselves. For example, documents that do not include the term "attacks", but include semantically related terms such as "maul", "injury", and "fatal" may be relevant to the query "black bear attacks". As a result, users expect that adding terms might improve the accuracy of a query. In contrast, search engines ignore semantics and miss relevant documents users expect such as those without "attacks", because search engines require documents to contain all query terms. Consequently, queries with narrow search goals are problematic, as at best, longer queries miss relevant documents, and at worst, longer queries find no results at all.

To users, the presence of each individual query term is purposeful. Users expect the retrieved documents to focus substantially on all the query terms and not just parts in isolation. In particular, documents may contain a query term without focusing substantially on it. For example, a document that principally discusses the construction of transportation

tunnels may be irrelevant, even if it contains all the terms in the query "transportation tunnel disasters", because merely mentioning that disasters occur in tunnels does not put a substantial focus on disasters. In contrast, search engines ignore the purpose of query terms; consequently, queries with multiple aspects sometimes have underrepresented aspects and retrieve irrelevant documents that focus on parts of the query in isolation.

To users, the absence of a query term is purposeful. Users expect the retrieved documents to match the broadest interpretation of the query and not sub-topics of the query. In particular, if a user sought a particular subtopic, they would add related query terms (missing aspects not withstanding). For example, a user wanting general information about cars might use "car", while a user wanting to buy a car might use "buy car". In contrast, search engines ignore the absence of query terms and sometimes find documents related to sub-topics such as documents about buying cars for the query "car"; consequently, queries with broad search goals are problematic.

To users, the order of the query terms is significant and different orderings communicate different goals. Users expect the retrieved documents to reflect the phrases in the query and not merely the individual words, similarly they do not expect the retrieved documents to reflect phrases not present in the query. For example, the query "air new zealand" should find the airline, but "new zealand air" should not (it should find documents about breathable air such as documents about the quality of air in New Zealand). In contrast, search engines generally[13] ignore the order of the query terms. Given this, it is unsurprising that Huang and Efthimiadis [89] found that term reordering is not a beneficial refinement strategy and that any ordering of the terms "air", "new", and "zealand" finds the airline. Consequently, queries with multiple aspects ("new zealand" and "air") sometimes have underrepresented aspects ("air" in "new zealand air") and

[13]Search engines do model a limited number of phrases such as "San Francisco" [71].

retrieve irrelevant documents that focus on something completely different (an airline).

To users, higher order constraints and natural language are meaningful. In contrast, search engines ignore them, and consequently, queries that use higher order constraints or natural language are problematic.

2.5.2 No Refinement Help

Typical search engines do not provide refinement suggestions, ignore the iterative nature of search, and use flat unstructured result sets that hinder the user's ability to decipher why queries failed.

When a query finds few or no relevant results, such as any problematic query, users must refine their query to find relevant documents. Query refinement is challenging because it involves both understanding why the previous query failed and generating a new query to overcome those limitations. By improving search engine query models to match user query models, there will be fewer problematic queries, but there will still be problematic queries to refine, including queries with ambiguous keywords, queries with missing aspects, and queries with imprecise keywords. Unfortunately, typical search engines do not provide refinement suggestions and users must refine all problematic queries by themselves.

Query refinement is an iterative process in which users try a series of queries, looking for one that finds relevant documents. Each successive query communicates more about the user's search goal. The later queries communicate additional information both explicitly by adding new keywords and implicitly by inferring that previous result sets did not contain relevant results. Unfortunately, typical search engines treat each query independently, ignoring the iterative nature of search, and therefore, do not make full use of the information provided by the user about their search goal.

Refining queries requires understanding why the previous query failed,

which requires knowing what irrelevant results the query retrieved. There are usually many different classes of irrelevant results, each with many documents in the result set. Typical search engines provide only flat unstructured result sets, which provide no organization to the result set and no guidance on the range of retrieved documents. Consequently, users must either look through many pages of results and many irrelevant documents to identify the distinct classes before refining the query or refine the query many times, each time identifying and removing one distinct class. Unfortunately, both approaches are tedious and time consuming.

2.5.3 Hide Less Popular Information

Typical search engines use popularity as a proxy for authority, but this makes less popular information even harder to find.

For most queries, there are millions of matching documents, but the quality of these documents is highly variable. Search engine relevance measures use a number of factors to judge the quality of a document to help rank well-researched, trustworthy, high quality documents above poorly-researched, untrustworthy, low quality documents. One factor that has proven success as a proxy for quality is popularity. Popularity is determined by link analysis methods such as Google's PageRank [20, 58] by counting the number of incoming links to each document, weighted by the authority of the sites on which those links originate. The idea is to leverage the wisdom of crowds [173] — documents with many links from high quality documents are themselves likely to be of high quality, because authors of high quality documents would not link to low quality documents.

The problem with ranking by popularity is that it exacerbates the difficulties caused by the popularity factor discussed in section 2.3.4.3. Popularity measures are very successful at ordering relevant documents by quality, but search engines often retrieve both relevant and irrelevant documents. When the irrelevant documents relate to topics that are more pop-

ular than the search goal, the irrelevant documents often dominate the search results and even if the user looks through many result pages, they may find no relevant documents. Consequently, when ranking by popularity, ambiguous queries are more likely to find exclusively documents related to a popular interpretation of the query, rather than a range of documents for different interpretations. Queries with broad search goals are more likely to find popular, but irrelevant sub-topics. Queries with multiple aspects are more likely to find popular related topics that are irrelevant. Rare information that would normally be hard to find may become almost impossible to find.

Ranking by popularity also makes search engines susceptible to attempts to manipulate search rankings. Websites have incentives to be at the top of search results[14], regardless of their relevance. Therefore, websites try to manipulate search rankings for their own gain and popularity provides a mechanism for them to do this.

2.6 Addressing Search Engine Limitations

While users can compensate for the limitations of search engines, ultimately, improvements to search engines will address the limitations. The primary goal of this thesis is to develop some of those improvements. This section describes how users compensate for the limitations, describes the mechanisms provided by search engines to help users compensate, and identifies the limitations addressed in this thesis.

2.6.1 Compensating for Limitations

Users can compensate for the limitations of search engines by adapting to the search engine's model as they gain search experience. In practice,

[14]as this means more traffic to their sites, and ultimately greater revenue through sales or advertising

most users have probably adapted to the search engine query model — Strohmaier et al. [170] found that in the AOL dataset [139] between 96.99% and 98.31% of queries use an unintentional bag-of-words approach (the type of queries expected by search engines). The problem is that even once users gain search experience and understand the true search model, there remains a cognitive gap — users must translate queries from their natural query model to the search engine model to formulate effective queries. While users may prefer this approach for some easy searches as it enables them to express their search goal in fewer words, for many hard searches the cognitive gap, for which even experts must compensate, is large and as shown earlier in this chapter, sometimes even insurmountable. Improvements to search engines will reduce the cognitive gap (making search easier) and reduce the requisite search experience needed for users to construct effective queries (making search even more accessible).

Although search engines typically only find documents containing every query term, users can use the advanced Boolean query operators such as *and*, *or*, and *not* to specify that some terms are optional. However, these queries are tedious to construct[15] and when used inappropriately, they often reduce relevancy. Additionally, Eastman and Jansen [61] found that even when used appropriately, query operators do not improve relevancy — search engines do not take advantage of the additional information conveyed by query operators.

Although search engines typically ignore the order of the query terms, users can use the phrase operator to specify that some terms must occur together. While useful in some instances, in general, this mechanism fails to correct the interpretation of queries with different term orders. For example, even when "new zealand" is entered as a phrase separate from "air", the query still finds the airline. An alternate refinement strategy combines the phrase operator with the *not* operator to exclude the phrase "air new

[15]Specifying that documents must contain at least three out of five keywords requires 10 sets of 3 keywords to be correctly grouped using Boolean operators.

zealand" from the "new zealand air" query. This succeeds in removing documents related to that particular airline, but results in a query that finds other irrelevant documents that are more popular. To find the relevant documents it is necessary to exclude documents relating to air transport, the air force, NZ On Air, Apple MacBook Air, health spas, freight, air guitar, fighter planes, cinemas, travel, and navigation, among others. Furthermore, this refinement strategy only succeeded because this was an easy search; for hard searches, this refinement strategy typically fails.

2.6.2 Recent Developments at Google

Since beginning this thesis in earlier 2005, typical search engines such as Google have continuously improved. This section details three developments at Google since 2005 that combat the limitations of typical search engines.

Figure 2.1: Google suggests queries that are similar to the users.

Since late 2006 [172], Google[16] has been showing queries related to the current query at the bottom of the result page. For example, figure 2.1 shows the queries related to the ambiguous query "jaguar". While these do serve as refinement suggestions, they are not particularly useful. The

[16]Yahoo and Microsoft's Bing also have this feature.

problem is that for most queries, the suggestions are not closely related. For example, the suggestions for "asp.net c# reflection" include "asp.net c# foreach" and "asp.net c# string", which while broadly related to the query, bear no relation to a key part of the query (the term "reflection"). The suggestions are most useful for ambiguous queries, but even for the simplest ambiguous queries, they are not particularly useful. For example, while the suggestions for "jaguar" are useful for users seeking the animal interpretation, the refinements suggested for users seeking the car interpretation are poor and there are no suggestions at all for users seeking other interpretations such as the Macintosh operating system, or the Atari games console.

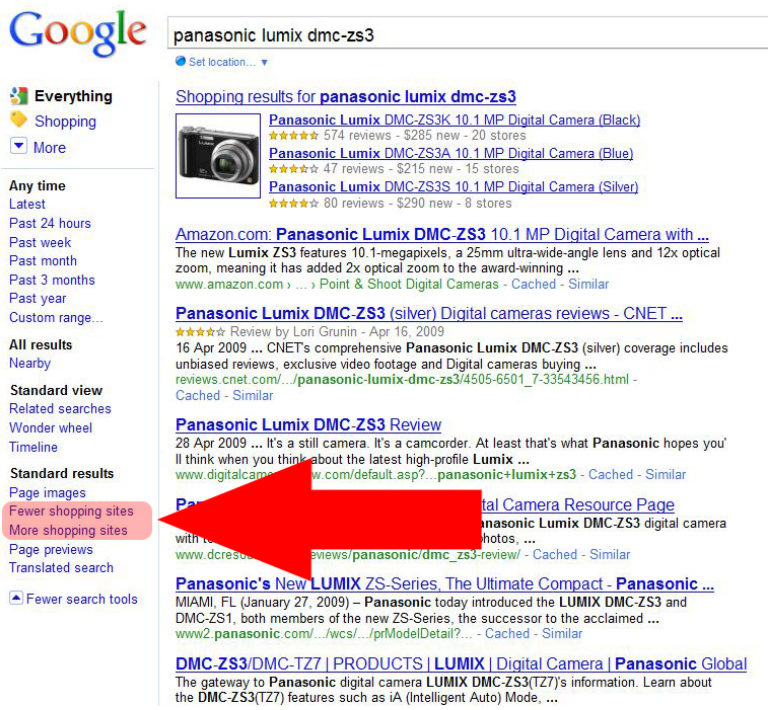

Figure 2.2: Google allows users to choose between more or less shopping sites.

Since late 2009 [35], Google has let users choose to view results with fewer shopping sites or more shopping sites as shown in figure 2.2. This

addresses a problem, caused by ranking documents using popularity, for a particularly common type of hard search — searches for product information from sources other than shops. Many users reported hard searches of this type in my study and for the most part, this recent addition to Google addresses those hard searches. However, while the method is useful, it does have two main limitations — the user has to choose when to apply it and it is highly specific. It only helps when shopping sites dominate the search results and not when other irrelevant documents dominate the results for the same reason (popularity).

In early 2010, I discovered that since completing the research and experiments for this thesis in early 2007, Google has changed from an exact match model to a best match model.[17] Instead of requiring the retrieved documents to contain all the query terms, the retrieved documents only need to contain a subset of them. My recent experiments (early 2010) suggest that although long queries now rarely find no results, adding additional terms to long queries tends not to increase accuracy or relevancy. Consequently, Google is not currently addressing long queries by considering the semantics of the query terms. Therefore, while the change will be beneficial for novice users that create queries with too many keywords, the change will generally not help users with hard searches that involve narrow search goals.

An additional negative consequence of the change from an exact match model to a best match model is that the refinement by semantically orthogonal keywords strategy (discussed in section 2.3.5) employed by experts is now less effective on Google. This refinement strategy relies on the addition of query terms to change the meaning of a query, but under a best match model, adding terms sometimes has no effect on the meaning of a query.

In the remainder of the thesis "Google" refers to the earlier versions of Google Search that used an exact match model.

[17] I could not find any public information on when this change occurred.

2.6.3 Limitations addressed by this Thesis

The user's queries provide an upper bound on the performance of search engines [170], so it is important that search engines make maximum use of that information. This thesis makes progress towards addressing the limitations of typical search engines primarily by making better use of the information contained in the user's queries.

The Query Directed Clustering approach in chapter 5 uses the semantics of the query terms to guide the construction of clusters. These clusters serve as both refinement suggestions for ambiguous queries and an overview of the result set that provides structure and aids users in understanding why their previous query failed.

The Query Aspect Approach in chapter 6 uses the order, presence, and semantics of the query terms to better understand queries with multiple aspects. AbraQ in chapter 6 uses the Query Aspect Approach to make searches with multiple aspects easier by lessening the impact of document complexity and popularity on those searches. Qasp in chapter 6 uses the Query Aspect Approach to suggest refinements. Chapter 6 also outlines how the Query Aspect Approach could be used to address queries for narrow search goals that use too many keywords and to leverage the iterative nature of search to gain additional information about the user's search goal.

Addressing the limitations related to the absence of query terms, higher order constraints, natural language, and popularity for single aspect queries is beyond the scope of this thesis. While this thesis does not explicitly learn from the absence of query terms, the corresponding problems arise mostly due to popularity, whose impact the Query Aspect Approach reduces for queries with multiple aspects. Although popularity for single aspect queries is still a problem, popularity is generally far less problematic for single aspect queries because they are generally far easier to refine than queries with multiple aspects. Future research could leverage the information learned from the absence of query terms to reduce the impact

of popularity on single aspect queries.

2.7 Search Engine Extensions

Search extensions build on existing search engines using the result set and query as input. Many types of search extensions exist: some add screenshots adjacent to results, others generate new snippets, and still others enhance certain results with additional information such as the ratings given by product reviews. This thesis focuses on search extensions that improve the quality of search results (automatic query expansion or AQE) and suggest refinements to users (interactive query expansion or IQE).

The research literature describes a variety of approaches for finding expansion terms, each with its own advantages and disadvantages. This section summarizes five different AQE and IQE methods: Relevance Feedback, Pseudo-Relevance Feedback, Thesauri Expansion, Web Page Clustering, and Query Log Analysis.

Query Directed Clustering in chapter 5 improves Web Page Clustering by using elements of Thesauri Expansion in a novel way. Chapter 6 explains a novel sixth approach termed the Query Aspect Approach.

2.7.1 Relevance Feedback (IQE)

Relevance Feedback methods [150] select expansion terms based on the user's relevance judgments for a subset of the retrieved documents. Relevance Feedback implementations vary according to the use of irrelevant documents and the method of ranking terms.

Normally, users explicitly identify only relevant documents; Ide [90] uses two approaches for determining the irrelevant ones. Ide-regular considers documents irrelevant when they are not marked as relevant and they occur earlier in the result set than the last relevant document. For example, when documents four and five are marked as relevant, documents

one, two, and three are irrelevant, and positions six and later are neither relevant nor irrelevant. Ide-dec-hi considers just the first non-relevant document irrelevant. While not outperforming Ide-regular, Ide-dec-hi is more consistent and improves more queries [150].

Researchers have considered many approaches for ranking terms, including all combinations of document frequency (df), term frequency (tf), and inverse document frequency (idf). Experiments consistently confirm that tf-idf performs the best [59].

Relevance Feedback generally improves Recall for queries that retrieve at least some relevant documents, but cannot do much for hard searches that retrieve only irrelevant documents. Another disadvantage is that Relevance Feedback places high demands on the user. Users must waste time checking the relevance of irrelevant documents, and they must make good judgments, because performance is highly dependent on judgment quality. In fact, Magennis and van Rijsbergen [114] found that while user selections improve performance, they were not optimal and often fell short of completely automatic expansion techniques such as Pseudo-Relevance Feedback.

2.7.2 Pseudo-Relevance Feedback (AQE & IQE)

Pseudo-Relevance Feedback [156], also called Blind or Ad-hoc Relevance Feedback, is nearly identical to Relevance Feedback. The difference is that instead of users making relevance judgments, the method assumes the first N documents are relevant[18], enabling Pseudo-Relevance Feedback to perform AQE.

Pseudo-Relevance Feedback generally improves Recall for queries that have high Precision [199], but often reduces performance for hard searches because they have low Precision and Pseudo-Relevance Feedback causes query drift for queries with low Precision [50].

[18] Using approximately N=10 provides the best performance [156].

Pseudo-Relevance Feedback can be adapted to IQE by suggesting the top ranked terms to users as different refinements. Like Relevance Feedback, this relies on user judgments, but the user's decision is simpler and performance is potentially greater because the feedback is more precise. However, as with AQE, the refinement suggestions are typically useful only when the top ranked documents are relevant.

2.7.3 Thesauri Expansion (AQE & IQE)

Thesauri provide information about the semantic relationships between terms. Thesauri Expansion [157] approaches use these relationships to identify expansion terms. As with Pseudo-Relevance Feedback, AQE approaches add these terms to the query, while IQE approaches show them to the user as possible refinements.

Human constructed thesauri like WordNet [38, 88] expose an assortment of term relationships. However, most approaches use only a limited number of these relationships such as synonyms (same or nearly the same meaning) and hyponyms (is-a) to avoid query drift. Despite that, these thesauri are not particularly useful for AQE or refinement of hard searches by IQE because even closely related terms such as synonyms can be polysemous and cause query drift. Thesauri are helpful for exploring related queries using IQE, because the term relationships are useful for constructing queries that are either more specialized or more generalized. Human constructed thesauri are also helpful when a single term is ambiguous.

The enormous number of documents on the web provides a great dataset for learning the semantics of terms. A useful alternative to human constructed thesauri is global document analysis, which can provide automatic measures of term relatedness. Cilibrasi and Vitanyi [40] introduced a particularly useful measure called Google distance that uses the co-occurrence frequency of terms in documents to gauge term relatedness. This thesis makes use of global document analysis and Google distance to learn the

semantics of terms.

Global document analysis is useful as it can identify co-occurring terms, which are potentially valuable for refinement, because semantically orthogonal keywords often co-occur with descriptive terms. However, adding co-occurring terms to a query often causes query drift. This thesis combines co-occurrence information with a deeper analysis of the query or result set to overcome this problem.

2.7.4 Web Page Clustering (IQE)

Typically, Web Page Clustering methods [201, 136, 122] differ from other IQE approaches in that selecting a refinement does not change the query. Web Page Clustering methods provide an overview of the result set by finding clusters of similar documents; selecting a cluster refines the result set onto a subset of it. Consequently, typical Web Page Clustering methods are directly helpful only when the result set already contains some relevant documents. However, they are indirectly helpful more frequently, because the clusters provide an overview of the retrieved documents and help the user efficiently understand why a query failed.

There are two methods of using Web Page Clustering to change the query in a way that is similar to other IQE approaches.[19] The first method is to use the selected cluster's description, which can simply be the most common terms in the cluster, to expand the query. The second method is to use Relevance Feedback, by assuming the documents in the selected cluster are relevant and that those in other clusters are irrelevant.

Query Directed Clustering in chapter 5 is a typical Web Page Clustering method that does not change the query.

[19] I am not aware of any researchers actually using these approaches.

2.7.5 Query Log Analysis (AQE & IQE)

Query Log Analysis examines the search patterns of users by considering the documents visited (user preferences and pairwise preferences) and the sequences of queries performed (query chains) [52, 96, 160]. By generalizing across many queries, search engines can identify reliable rules that describe patterns of behaviour that suggest refinements for AQE and IQE.

Search engines can learn which documents are most relevant and which are potentially irrelevant by analyzing the preference of users for different results. If most users who search for a given query choose to view the third result, then the first two results are more likely to be irrelevant. This information can guide the selection of terms for AQE or guide the relevancy ranking function to rank result three above results one and two.

Search engines can learn which queries are ambiguous from pairwise user preferences and learn refinements for them from the user's query chains. If one group of users who search for a given query views results one and five, but another group views results two, four, and nine, then it is possible that the query is ambiguous and these groups are seeking different interpretations of the query. Further evidence of ambiguity occurs when subsequent intra-group queries are similar and inter-group queries are different — the respective queries of each group may correspond to refinements for the different interpretations. Search engines could use this information as the basis for IQE and present the queries as refinement suggestions to future users who perform the same query.

Search engines can learn which queries are consistently refined in the same way from query chains and use AQE to change future queries in a similar way. This can also be an effective way for search engines to learn acronyms. For example, many users that search for "oed", may immediately search for "Oxford English Dictionary" [96].

Query Log Analysis works best for common searches, because it needs many similar queries for the generalizations to be reliable. Unfortunately, hard searches are typically uncommon [19]. This disadvantage is evident

from Google's related searches feature (section 2.6.2) that appears to use Query Log Analysis to show the most frequently occurring searches that are most similar to the current query, because it produces better results for common searches such as "jaguar" than for less common searches such as "asp.net c# reflection".

Chapter 3

Web Search Evaluation

Evaluating search involves comparing the relative effectiveness of different search methods. Different researchers have different goals, different constraints, and different interests. This diversity has led to numerous definitions of effectiveness and for each a multitude of approaches to measure or quantify that effectiveness. This variety of approaches, the tradeoffs between them, and other constraints make selecting an appropriate evaluation method challenging.

This chapter explores the multitude of approaches to evaluation, explores the considerations made by researchers in selecting an appropriate evaluation method, and in particular explains the selection of the evaluation method used by this thesis. The chapter starts very broad by considering the different criteria used for evaluation. Each subsequent section narrows the focus onto an element of the preceding section and the chapter culminates in an exploration of the different measures of performance.

3.1 Evaluation Criteria

The evaluation criteria define what it means for one search method to perform ”better” than another search method. There is a wide variety of evaluation criteria [106] and the criteria may be objective or subjective.

Objective measures include response time (the time taken to complete one query) and capacity (the number of queries a system can execute in one period). Subjective measures include relevancy (how closely the documents relate to the information needs pertaining to the query), quality (how authoritative, accurate, complete, and timely are the documents), and complexity (how much knowledge or learning is required before users can search effectively).

Objective measures are user independent, which makes evaluation simple using either theoretical methods like asymptotic complexity analysis or empirical methods like wall-clock timing. However, the usefulness of objective criteria is limited because search is user-centric and, relative to existing search methods, users get little benefit[1] from improved response time or capacity. The performance of current search methods is already more than adequate[2]: queries generally return results in less than 300 milliseconds and users encounter no practical limits on the number of queries they can execute. Therefore, from a user's perspective, objective measures alone are inadequate.

In contrast, subjective measures are user dependent, which makes evaluation more complicated. This thesis uses both objective measures and subjective measures. The objective measures ensure search methods are practical — it is possible and economically viable to achieve a response time and capacity comparable to existing search engines. The subjective measures evaluate the relative effectiveness of different search methods from the user's perspective. The remainder of this chapter explores the application of subjective measures to web search evaluation.

[1] Search engines may benefit due to reduced costs.

[2] The on-going growth in processing power provides scope for computationally expensive relevancy improvements without impeding on response time or capacity.

3.2 Methodologies

To evaluate subjective criteria, two methodologies have developed that differ in how they incorporate the user factor: the interactive or user oriented evaluation and the non-interactive or system oriented evaluation [144]. Interactive methods capture input by monitoring the experience of users exposed directly to the systems under evaluation; the users' actions can be supervised or unsupervised. Non-interactive methods capture input by gathering judgments in a controlled manner where users are unaware of the system under evaluation.

Each methodology solicits different information, has different strengths and weaknesses, and is therefore appropriate under different circumstances. This section explores the interactive methodologies, supervised and unsupervised, explores the non-interactive methodology, and then justifies the use of the non-interactive methodology in this thesis.

3.2.1 Supervised Interactive

A supervised interactive study [192, 97] involves giving users artificial information needs and then monitoring how successful users are at solving those information needs using the different search methods under evaluation. Performance is determined by comparing either quantitative or qualitative effectiveness measures. Quantitative measures include task completion time [192] and number of queries performed per session [97] by participants and qualitative measures include asking users for feedback about the systems such as their perceived effectiveness [192].

User studies are time consuming, costly, not reproducible, and difficult to perform correctly because experimenters must control numerous variables to ensure validity [80]. Another major drawback is that the user experience variable requires control, which entails all search methods being near production quality and able to run in real-time [103].

Supervised interactive studies are ideal for evaluating search complex-

ity: they have proven effective for comparing interfaces [192] and for determining the behaviour of users [97]. However, as many users must repeat the same experiments, they are inefficient for evaluating subjective criteria such as relevancy and quality.

3.2.2 Unsupervised Interactive

An unsupervised interactive study monitors users, who are unaware of the study, solving real information needs [94, 93]. Performance is determined by aggregating, across all participants, effectiveness measures such as task completion time and number of queries performed per session.

Like supervised user studies, unsupervised studies are time consuming, costly, and not reproducible. However, the unfocused nature of unsupervised studies amplifies these problems, as generally, analysis requires much data, and that requires many users. Another potential issue is that observations can have dual causes: task completion time may decrease because users solved their goals quicker or because users abandoned them quicker. Therefore, other studies that determine the cause and expected effect must precede unsupervised studies, so valid conclusions are drawn.

Unsupervised interactive user studies are ideal for determining the scope of a search method — the extent to which a method affects performance. Search methods can affect all queries, certain classes of queries, or affect some classes positively and others negatively. For example, one search method might improve all queries by 5%, another might improve the worst performing queries by 35%, and another might improve those same queries by 70%, but decrease the performance on all other queries by 5%. Scope defines the relative size of affected classes and the magnitude of a search method's effect on each class.

Scope is an excellent measure of the overall impact of a search method, but unfortunately, unsupervised interactive studies require many users, which restricts them to just the largest commercial search engines. The

result is relatively few studies, which were limited to studying individual search engines through query logs, thereby limiting results to general usage of current search engines [94, 93].

3.2.3 Non-Interactive

Non-interactive evaluations use a small number of queries and corresponding sets of relevance judgments[3] made by one or more expert users [11]. Performance is determined by comparing the relevance judgments with the search results.

Unlike interactive evaluations, non-interactive evaluations operate in a controlled environment that is completely independent of the search interface. This makes non-interactive evaluations unsuitable for evaluating search complexity and the effectiveness of different interfaces. However, it allows non-interactive evaluations to evaluate relevancy and quality for systems that are not yet production quality, currently lack an interface, or which cannot run in real-time because of resource constraints.

Non-interactive evaluations are ideal for evaluating the relative result relevancy and quality of many search methods and rank search methods consistently, even based on only one expert's relevancy judgments [83, 185]. Compared with supervised interactive studies, the same number of users can evaluate many more queries, dramatically improving efficiency and reducing the cost of performing evaluation.

3.2.4 Methodology Selection

Each methodology focuses on (or more correctly, is best at) evaluating a specific feature of search. Table 3.1 summarizes the focus, and cost[4] of each methodology. The colour indicates their relative applicability to evaluat-

[3]Quality judgments may supplement the relevance judgments.

[4]Cost represents both the resource requirements and complexity of performing that type of evaluation.

Table 3.1: Features of evaluation methodologies and their relative applicability to this thesis

	Focus	Cost
Non-Interactive	Relevancy / Quality	Low
Supervised Interactive	User Behaviour	Medium
Unsupervised Interactive	Scope	High

ing the search methods in this thesis: green, yellow, and red reflect high through low applicability respectively.

Ideally, one would evaluate search methods using all three methodologies. However, this is often impractical (due to resource constraints) and generally wasteful (due to duplication). For example, the search methods in this thesis use standard user interfaces that have already had their complexity and learnability studied by the literature. Re-evaluating these standard components using a supervised interactive user study would be wasteful, because it involves implementing an otherwise unnecessary user interface and ensuring it is of production quality.

For most search methods, including those in this thesis, evaluating scope would be beneficial. However, as discussed in section 3.2.2, this is impractical for most researchers, due to the enormous resource requirements. Consequently, like other researchers, this thesis is unable to evaluate scope, because it is infeasible to perform a large interactive unsupervised user study. However, where appropriate this thesis makes conjectures regarding scope based on large user studies of commercial search engines.

When investigating new user interfaces or user behaviour, a supervised interactive study offers significant benefits over other methodologies. However, they are less efficient than non-interactive studies at measuring relevancy and quality. Additionally, for some search methods, in-

cluding some in this thesis, it is impractical to create the production quality real-time experience necessary for a supervised interactive evaluation due to excessive resource requirements.

This thesis focuses on search methods that improve relevancy and principally use standard interfaces.[5] This means there are no user interface or user behaviour implications that require evaluation and no need to use the supervised interactive methodology. To evaluate relevancy, this thesis uses a non-interactive methodology and the remainder of this chapter explores the factors involved in performing non-interactive evaluation.

3.3 Corpus Selection

Evaluating a search method involves choosing the corpus on which to run tests. The corpus is the set of documents searched in response to a query and is the focus of this section.

Various corpora exist and each exhibits different characteristics such as size (number of elements), type (web pages, text documents, images, etc.), length (snippets, short articles, long articles, resolution, etc.), and other characteristics such as the degree of linkage between pages. Selecting a corpus that accurately represents the target domain is important because search methods perform differently on different corpora. For example, an effective algorithm for searching billions of web pages is probably ineffective at searching within a single text document and is certainly ineffective at searching through millions of images.

For an accurate and representative evaluation, the corpus must be sufficiently similar to the target domain. The search methods in this thesis target large collections of web pages such as those indexed by commercial

[5]By improving relevancy, some search methods in this thesis might additionally reduce complexity for standard interfaces; this thesis identifies these cases using qualitative analysis, but leaves further quantitative analysis for future work because that is outside the scope of this thesis.

search engines like Google and Yahoo and hence, their web page collections provide the most natural corpora to use for evaluation in this thesis. The remainder of this section explores the characteristics of available document corpora and explains why the smaller corpora are unsuitable substitutes for search engine corpora when evaluating the search methods developed in this thesis.

3.3.1 Document Corpora

The available corpora can be broken down into roughly four categories: classical corpora, news corpora, web corpora, and search engine corpora. An important contributor and the source of many recent document corpora is TREC, the Text REtrieval Conference [47]. Table 3.2 summarizes the available collections including those used by TREC, collating data from the collections themselves and several sources [11, 73, 99, 74, 81, 104, 184].

Table 3.3 summarizes the relative merits of each corpus category: green, yellow, and red reflect high through low applicability respectively. Each corpus category has problems and none is perfect for evaluating all search algorithms, but each has its merits and is well suited to evaluating certain kinds of search algorithm.

Classical corpora are useful for evaluating computationally expensive algorithms including Latent Semantic Indexing or Singular Value Decomposition based approaches[104] and those using richer natural language analysis such as question-answering systems [182]. News corpora are useful for classifier evaluation [108]. Web corpora are useful for evaluating baseline search algorithms like BM25 [23, 159]. Search Engine Corpora are useful for evaluating scale-dependent search improvements including linkage analysis methods like PageRank [20] and frequency analysis methods like Google Distance [40].

Table 3.2: Summary of Available Document Corpora

Corpus	Number of Documents	Creation Date	Avg Off-Site In-Degree
Classical Corpora			
LISA	5,872		
NPL	11,429		
CACM	3,204	58–79	
CISI	1,460		
Cranfield	1,400	22–63	
Time	523		
Medline	1,033		
ADI	82		
News Corpora			
OHSUMED	348,566		
Reuters-21578	21,578	87	
RCV1	810,000	96–97	
RCV2	487,000	96–97	
Aquaint	1,033,000	96–00	
Web Corpora			
GOV	1,247,753	2002	1.98
GOV2	25,205,179	2004	
NTCIR-3 Web	11,038,720	2001	
WT10g	1,692,096	1997	0.101
WT2g	247,491	1997	0.011
VLC	7,492,048	1997	
VLC2 (WT100g)	18,571,671	1997	
SPIRIT	94,552,870	2001	1.24
Blog06	3,215,171	2006	
Search Engine Corpora			
Google/Yahoo	20,000,000,000		4.916

Table 3.3: Merits of different corpora and their relative applicability to this thesis

	Classical	News	Web	Search Engine
Size	Very Small	Small	Medium	Large
Linkage Analysis	No Links	No Links	Too Small	Good
Frequency Analysis	Too Small	Too Small	Too Small	Good
Web Documents	No	No	Yes	Yes
Reusable	Yes	Yes	Yes	No
Modifiable Baseline	Yes	Yes	Yes	No

3.3.2 Search Engine Corpora

Search engine corpora are huge, tens of billions of pages, which, as this section will explain, makes them ideal for linkage analysis and frequency analysis. Still, search engine corpora have drawbacks. Researchers cannot modify nor even inspect the baseline search method used, because the search engine determines this parameter. This restriction makes search engine corpora useless for evaluating some search methods. However, all the search methods in this thesis build atop existing baseline methods and are therefore, unaffected by this restriction. Additionally, search engine corpora are neither stable nor static and as such, they are not reusable — a definite drawback for any evaluation, but insufficient to preclude their use when search methods use linkage analysis or frequency analysis.

Search methods that use linkage analysis and analyze the structure of the web through hyperlinks require very large collections. Ideally, samples of the web such as GOV2 would be sufficiently large to enable linkage analysis. However, TREC participants have routinely been unable to show any benefit of PageRank or link analysis methods on the relatively small TREC collections [73]. Even the largest web corpora, including SPIRIT [99] with almost 100 million pages is still too small to provide an adequate

representation of the web: the average off-site in-degree for the web is estimated to be four times larger than that in the SPIRIT collection [74]. Researchers have constructed small subsets of these collections that exhibit web like linkage characteristics and shown positive improvement from linkage analysis [11], but the improvements have been small [81] and document selection may be biased [74].

Search methods that use term frequencies perform significantly better on larger collections, because the term frequencies become more stable and more accurate. One problem with relatively small collections is that they often cover relatively few terms: while Google covered 99.96% of the nouns and noun phrases in WordNet, the TREC collections only covered 52.59% [98]. Even on the terms covered by the TREC collections, in a relatively basic hypernym chain test, Google's performance was about 1.5% better and the result was significant [98]. The performance difference grows dramatically on harder problems such as those analyzing longer phrases or co-occurrence of multiple terms. For the problem of question answering, a problem requiring analysis of longer phrases, Dumais et al. showed that Google (with 2 billion pages at the time) performed almost 300% better than the same algorithm on a TREC collection with three orders of magnitude fewer pages (1 million) [60]. They also showed Google performed 20% better at this task than MSN which was using Inktomi and 500 million pages at that time — evidence of the importance of collection size to the accuracy of frequency analysis.

The search methods in this thesis make extensive use of frequency analysis, which only works well on huge collections. Therefore, like others working on frequency analysis methods such as Google distance [40], this thesis evaluates search methods using search engine corpora. Additionally, this choice enables the baseline methods to benefit from linkage analysis, which increases the evaluation accuracy, as section 3.5 will explain.

3.4 Obtaining a Query Test Set

Having elected to use search engine corpora, the next challenge is obtaining the test set, a set of test queries and relevance judgments for the documents retrieved in response to each query. Constructing a new test set represents the largest cost in performing a non-interactive evaluation, because it involves experts inspecting and judging the relevance of thousands of documents per test query. Ideally, researchers avoid this huge cost by reusing test sets constructed by other researchers. However, while test sets exist for classical, news, and web corpora, no reusable test sets exist for search engine corpora, because search engine corpora are constantly changing. Therefore, choosing to use search engine corpora entails creating a new test set from scratch.

Search engine corpora provide the most accurate and representative evaluation of the search methods explored in this thesis. While an accurate and representative evaluation is of utmost importance, other factors such as cost, repeatability, and reusability deserve consideration. Of the available corpora, only web corpora are remotely similar to search engine corpora. While web corpora would provide a less accurate and less representative evaluation, they potentially lower the cost of evaluation significantly, because test sets already exist for web corpora. Therefore, web corpora warrant further consideration.

Unfortunately, web corpora are not a panacea for the problem of constructing a test set and in fact offer no advantages over search engine corpora. Firstly, it is possible to reuse the test queries from existing web corpora test sets with both web corpora and search engine corpora. Secondly, it is unsound to reuse the costly relevance judgments from web corpora test sets and therefore, as with search engine corpora, new relevance judgments are required when using web corpora. Finally, as section 3.5 will explain, using web corpora precludes using the best available baseline search method, which would taint the evaluation — lowering the eval-

uation's accuracy even further.

After explaining why it is beneficial to reuse TREC queries, this section explores the number of queries needed to evaluate search methods. Then it explains how to collect relevance judgments from experts efficiently and without bias toward any specific search method. Finally, it explains why the reuse of relevance judgments on large collections is unsound and hence, why web corpora offer no advantage over using search engine corpora with respect to obtaining a useable test set.

3.4.1 Reusing TREC Queries

Reusing queries from existing test sets reduces the cost of producing new test sets for different corpora. Additionally, objectivity can be increased and bias avoided by using test queries chosen by third parties. The challenge is finding suitable test queries that are reflective of real queries that search methods would encounter. Even more challenging is accurately specifying the search goal such that experts could unambiguously judge documents retrieved by a query as relevant or irrelevant to the given search goal.

There are two main sources of predefined test sets: the classical corpora and TREC, which through its various tracks has constructed many test sets, principally for web corpora. The test sets for classical corpora consist of long natural language queries, which are atypical of those posed to modern search engines and are therefore unsuitable for web search evaluation. Conversely, the TREC test sets consist of short keyword queries for a broad range of real world scenarios that cover all known search tasks including navigational, transactional, and informational [21, 149] and are therefore suitable for web search evaluation.

The clear test set definitions given by TREC, which consist of topic-narrative pairs, make it easy to reuse queries on different corpora and in new test sets, thereby reducing the cost of constructing new test sets. Each

"topic" provides a few keywords that can be treated as a query; each "narrative" gives specific details on what would be relevant and irrelevant documents, which can be treated as an unambiguous definition of the user's search goal and therefore as explicit guidelines for experts making relevancy judgments.

The diverse range of test sets created by TREC makes finding suitable test queries simple and objective. For example, it was simple to find hard queries for current search engines from the TREC HARD track, which features 50 queries determined to be difficult for current search engines.[6] If these third-party queries were unavailable, I would need to discover or construct my own hard queries, which could lead to a non-objective evaluation, because my assumptions about hard queries, which biased the construction of my algorithms, might bias my selection of the hard queries used for evaluation. Using a third-party source of test queries avoids this source of bias and makes the evaluation more objective.

3.4.2 Number of Queries

Ideally, all evaluations would consider many queries. Unfortunately, available resources impose hard limits on the number of queries evaluated. For typical non-interactive, relevance judgment based studies such as TREC, where relevance assessors are employed, this limit has typically been 50 queries. From my experience and that of other researchers [6], judging the relevance of 500 documents takes approximately 7 hours. On this basis, comparing 20 search refinement methods, which on average produce 10 refinements, across 50 queries on the top 10 results, would involve someone judging the relevancy of 100,000 documents and that would take one person about 40 weeks of very laborious full-time work.

Fortunately, a smaller sample size (number of queries) such as 15 queries is acceptable. When the results are statistically significant with high con-

[6] based on previous TREC results

fidence, there is a low probability of a Type I Error (false positive) and therefore, provided the test shows a significant difference, the sample size is irrelevant. The problem with using a small sample size is that the power of the statistical test will be low, which means there will be a large probability of a Type II Error (false negative), namely, a significant difference exists, but is not detected. This is not problematic here, since this thesis is interested in only large differences (>10%) between search methods, which are much less susceptible to Type II Errors. It is acceptable for the evaluation to have insufficient power to detect small differences, because such differences would have negligible impact on users anyway.

3.4.3 Collecting Relevance Judgments

It is costly and time consuming to collect relevance judgments from experts, so the process should be efficient; however, accuracy is critical to the integrity of the evaluation. To ensure accuracy, the collection process must minimize bias and should lead to consistent and valid conclusions.

An important factor in choosing the collection process is the size of the collection, because collection methods that work well on small collections are often impractical for large collections. With small corpora, those containing thousands of documents, it is feasible to build test sets that contain complete relevance judgments. A test set contains complete relevance judgments when experts have judged the relevancy of every document for every query in the test set. Test sets with complete relevance judgments are reusable because they can reliably rank any search method, including new search methods. Unfortunately, it is impractical to collect complete relevance judgments for large collections — for a collection containing millions or billions of documents that would mean judging the relevance of millions or billions of documents for every test query.

To enable the practical evaluation of search methods on large collections, Sparck Jones and Van Rijsbergen [100] proposed “pooling”, which

dramatically reduces the number of judgments required to compare search methods on large collections. To evaluate one query using pooling, each search method under evaluation runs the query, and then experts judge the relevancy of the top-N documents retrieved by each search method (the pool). Pooling is now widely used by researchers and within TREC. However, as the next section will discuss, because pooling assumes that un-judged documents are irrelevant, there are problems with reusing test sets to evaluate new search methods — methods whose retrieved documents were not included in the pool.

Bias can compromise the accuracy of a test set. Sources of bias include judgment order and search method awareness. The order in which experts judge documents is important, because an expert's opinion can change as they view documents. For example, if an expert judges all the documents retrieved by search method A before those retrieved by search method B, and the judge becomes stricter with their interpretation of relevance as they proceed, then the results will be biased towards search method A. It is also important that experts are unaware of the search method that retrieved the results they are judging, because experts may unconsciously favour certain methods. This thesis avoids these sources of bias by using the widely used approach [83] of merging the results retrieved from all search methods and then presenting them in a blind random order to the expert.

Another potential source of bias is the experts themselves, because different experts often make different judgments regarding the same documents. However, it turns out that even though these differences are common, it is sufficient to use just one expert's judgments per query. Voorhees [185] found that although three users only agreed on relevancy about 72% of the time, regardless of which single user's judgments were used, the overall ranking of methods is the same. Furthermore, while it is safe and typical to use a single expert per query, recent findings suggest that it may not even be necessary to use the same judge for an entire query [6] —

instead, the workload for a single query could be divided between multiple researchers, each judging the relevance of different documents. For efficiency and as is now standard, the relevance of each document was judged only once using a single expert; however, for safety, the same expert provided judgments for each entire query.

Relevance judgments may be binary or use a graded scale and both approaches have been used in the literature [83]. To compare these approaches, I used both binary and graded judgments while performing a subset of the web search evaluations in this thesis. The graded judgments required significantly more work, but although there was an absolute difference between binary and graded judgments, there was no difference to the ranking of methods, which agrees with the findings of more formal studies on binary and graded judgments [6]. Therefore, this thesis uses binary judgments.

3.4.4 Reusing Relevance Judgments

The most costly part of constructing a new test set is collecting the relevance judgments from experts, because this can involve experts judging the relevance of thousands of documents for each test query. Ideally, researchers would reuse existing relevance judgments to eliminate this cost. Unfortunately, no suitable relevance judgments exist for search engine corpora because the web is constantly changing and even though relevance judgments exist for web corpora, the reuse of the currently available judgments to evaluate new search methods is not sound. The result is that evaluations of new search methods on large corpora like web corpora and search engine corpora require the expensive and time-consuming collection of new relevance judgments.

While pooling makes it practical to evaluate search methods on large collections, it limits the reusability of those judgments on new search methods. The problem is that the performance of new search methods is un-

derestimated when they retrieve un-judged documents[7] that are relevant [161]. This occurs because the evaluation method incorrectly assumes the un-judged documents are irrelevant. Therefore, while it is valid to compare methods included in the pool, it is invalid to compare methods outside the pool with either methods inside the pool[8] or other methods outside the pool [26]. In general, this means that test sets constructed by pooling are not reusable.

When test sets contain "sufficiently complete" relevance judgments, it is safe to reuse them to evaluate new search methods [11, 26, 161]. A set of relevance judgments is sufficiently complete if the pools are large enough and diverse enough. The problem is determining whether pools are sufficiently large and sufficiently diverse. The standard test for this is known as the LOU test (leave out uniques) [26] and is based on comparing the difference in performance of search methods after removing the relevant documents that were uniquely contributed to the pool by that particular search method. Justin Zobel [210] performed the first LOU tests and found that the test sets for the small corpora of TREC-3, TREC-4, and TREC-5 contained sufficiently complete relevance judgments and are therefore reusable. Unfortunately, later studies have shown that the test sets for larger collections are insufficiently complete and cannot be reused [26].

The problem with larger collections is that experts judge few documents relative to the size of the collections. This means that large collections will generally have a higher percentage of un-judged relevant documents than small collections. Chris Buckley et al. [26] found using the LOU test that even the small Aquaint corpus has insufficiently complete relevance judgments, because the pool sizes were too small relative to the size of the collection. The problem is even worse on larger corpora like GOV2 where in one case, the LOU test showed a decrease in performance

[7] Documents that were not retrieved by methods included in the pool.

[8] However, it is valid to conclude that methods outside the pool exceed the performance of methods inside the pool.

of 45.5% [26]. Tragically, the best performing search methods are potentially the worst affected, because they find the greatest number of relevant documents that other algorithms completely missed and as a result are un-judged and assumed irrelevant.

The result is that even if this thesis used web corpora, new relevance judgments would be required to evaluate fairly the new search methods presented in the thesis. Therefore, there is no drawback, relative to web corpora, to the use of search engine corpora.

3.5 Selecting a Baseline Search Engine

There are two varieties of search method: search engines and search extensions. Search engines directly access the underlying documents: they index raw documents, then in response to queries they search those indexes and retrieve results that match the query's keywords. Search extensions indirectly access documents through search engines: they sit atop search engines and enhance functionality or performance by intercepting and modifying the input to search engines or the output from them. For example, automatic query expansion methods change the input by modifying queries to improve the relevancy, while clustering methods change the output by organizing the results to help users understand the different interpretations of their query. This thesis focuses on search extensions.

While search extensions fall into a range of categories such as clustering, interactive query expansion, automatic query expansion, each inherently requires an underlying search engine. To maximize the comparability of different search extensions, each search extension should extend a common baseline search engine. The common baseline serves two purposes, firstly it ensures a level playing field across search extensions, and secondly it provides an absolute baseline for comparison — search extensions that do not significantly improve the baseline search engine's performance are clearly ineffective. Note that evaluations will also involve

category specific baselines, for better comparison with prior research.

Two main categories of search engine are available for use as a baseline: commercial search engines such as Google and research algorithms such as those commonly used by TREC participants. Having already elected to use search engine corpora, the choice of baseline is limited to commercial search engines. However, as this section will show, using a commercial search engine for evaluation is currently optimal because they outperform the alternatives and would therefore be the preferred choice anyway.[9] Additionally, using commercial search engines is comparatively easy and cost effective because it is difficult and time consuming to maximize the performance of the best research algorithms.

3.5.1 State of the Art Baselines

Given a choice between baselines, the one with the best performance should be preferred. Firstly, an improved result ranking is preferred to a search extension that achieves the same effect, because search extensions usually involve additional user effort. Secondly, when evaluated on a poorly performing baseline, search extensions that improve limitations that only exist in the poor performing baseline may appear to outperform better search extensions that improve performance of any baseline. For example, two search extensions may improve a poorly performing baseline by 15% and 20%, while on a better baseline the same two extensions make improvements of 15% and 2% respectively. Hence, it is important to use the best performing baseline methods.

From its outset, Google has been a top ranking search engine [83]. Recently, Yahoo and Microsoft have dramatically shrunk the gap, but Google is still the top ranking search engine and outperforms Yahoo by about 4% and Microsoft's MSN/Live search engine by about 7% for informational

[9] This result reinforces my decision to use search engine corpora, because external parties can only use commercial search engines with search engine corpora.

queries [110, 111]. Therefore, with respect to commercial search engines, Google is the best baseline.

There is a wide variety of research algorithms for implementing search engines — the development of these algorithms is ongoing and TREC participants regularly evaluate their performance. These algorithms group together into classes known as search models or retrieval models. The three main models are vector space [152] (most known for methods using tf-idf), divergence from randomness [5] (most commonly implemented as Okapi BM25 [147, 63]), and language models [13] and all have been widely explored by TREC participants.

There is enormous variation in the particular implementations and algorithms often incorporate numerous additional features to improve performance such as PageRank [58] and Latent Semantic Analysis [54]. These variations lead to very substantial differences in performance, making it both hard to compare the performance of algorithm classes and difficult to implement a well performing state-of-the-art search engine. The TREC results of different algorithms on different data sets [48, 176] show that the most popular algorithms, probably due to their simplicity, are algorithms based on the vector space model and Okapi BM25, while the best performing algorithms, with comparable levels of performance, were those based on Okapi BM25 and language models . Therefore, with respect to research algorithms, good implementations of Okapi BM25 or language models provide the best baseline.

3.5.2 Comparing TREC and Commercial Search Engines

Several studies have compared the performance of state-of-the-art TREC systems (Okapi BM25) against commercial search engines. In many of these studies, search engines outperform TREC systems: search engines improved on TREC systems by 50 – 60% on a homepage finding task [159] and by 9% on a service location task [81]. In studies on queries that are

atypical of web search and for which commercial search engines are unoptimized such as long natural language queries [83], the optimized TREC algorithms outperformed commercial search engines by 10%.

Implementing Okapi BM25 or an algorithm based on language models requires implementing all the various tweaks and extensions that improve performance over the basic methods and then tuning the parameters on all these extensions to work well together. This in itself is a challenging endeavour that is time consuming, difficult to get right [48], and beyond the scope of this thesis. In contrast, Google's researchers have already put in years of work tuning their algorithms for the short keyword queries that are typical of web search and the focus of the research in this thesis.

While TREC systems may outperform commercial search engines on some queries, in their current form, the TREC systems fail to outperform commercial search engines on typical web queries like those investigated in this thesis. This performance difference, combined with the complexity of implementing a competitive TREC system, justifies the use of Google as the baseline search engine.

3.6 Performance Measure Selection

Researchers use many measures to evaluate web search performance. There are two main categories of measurement: set based measures and sequence based measures [106]. Set based measures work irrespective of order and typically measure the entire result set and document set, while sequence based measures consider the order of the result set.

This thesis primarily uses a sequence-based measure called Precision@N or more specifically uses P@5 and P@10, which are the probabilities of finding a relevant document within the first 5 and first 10 results respectively. P@10 is the critical measure of success for an informational search if the user is never willing to go beyond the first page of results, which for the most part is true with web search.

3.6.1 Set Based Performance Measures

The most common set based measures are Precision, which measures quality — the proportion of retrieved documents that are relevant; and Recall, which measures retrieval effectiveness — the total number of relevant documents retrieved. Other set based measures include Fallout and Error Rate [106].

$$\text{Precision} = \frac{r}{r+i}$$

$$\text{Recall} = \frac{r}{n_r}$$

$$\text{Fallout} = \frac{i}{n_i}$$

$$\text{Error Rate} = \frac{i + n_r - r}{n_r + n_i}$$

where r and i are the number of relevant and irrelevant documents retrieved and n_r and n_i are the total number of relevant and irrelevant documents.

Researchers rarely use set based measures for evaluating web search for two reasons. Firstly, they inadequately reflect how users actually use search engines. Users rarely look beyond the first few documents and hardly ever past the 30th, as evidenced by the study of two large AltaVista datasets [94] and my analysis of the AOL dataset [139]. Therefore, the relevancy of the 100th document is practically meaningless, let alone the 1000th document, however, the relevancy of the documents ranked 100th to 1000th have far more weight in set based measures than the documents ranked 1st to 100th. Secondly, they require complete relevancy judgments for the entire data set, which is impractical for all but the most trivial data sets [82].

3.6.2 Sequence Based Performance Measures

Sequence based measures all depend on the number of relevant documents in the first part of a sequence of results. The most common sequence

based measures are Precision@N ($P@N$), Mean Average Precision (MAP), and Mean Reciprocal Rank ($MRR1$). Other sequence based measure include Success@N ($S@N$), R-Precision, and Recall@N ($R@N$) [81, 23, 1].

$$P@N = \frac{r_N}{N}$$

$$MAP = \frac{\sum_{k=1}^{n_r} P@p_k}{n_r}$$

$$MRR1 = \frac{1}{p_1}$$

$$S@N = \frac{|\{q : q \in Q \wedge r_N > 0\}|}{|Q|}$$

$$\textit{R-Precision} = P@p_R$$

$$R@N = \frac{r_N}{n_r}$$

where r_N is the number of relevant documents in the first N documents, n_r is the number of relevant documents, p_n is the position of the n^{th} relevant document in the sequence, and Q is the set of queries.

Like set based measures, the computation of recall is impractical [82]. However, since recall primarily measures the performance of exhaustive search [81] and this thesis focuses on informational search, there is little value in measuring recall.

$S@N$ and $MRR1$ are appropriate measures when there is just one relevant document, as in navigation search. The remaining measures — $P@N$, MAP, and R-Precision — are appropriate for informational search and transactional search [81].

Like recall, MAP and R-Precision quickly become impractical for large corpora, because they can require relevance judgments for huge numbers of results, when documents known to be relevant occur far down the ordering. Furthermore, as shown in table 3.4, the simple $P@N$ measure produces comparable results to MAP with differences in order only occurring between algorithms whose performance is not significantly different;

MAP and R-Precision also produce comparable results as both approximate the area under a precision-recall curve and therefore their results are correlated [9].

This thesis uses $P@N$ for its simplicity and used N=5 and N=10 to reduce the number of judgments required. The choice of small N values reflects that users rarely look beyond the first page of 10 results. Additionally, $P@a$ quite accurately predicts $P@b$ where $a < b$, even for $a = 1$ [83], therefore $P@5$ and $P@10$ are quite informative.

Table 3.4: MAP and P@10 measures lead to comparable conclusions. The table shows results for both measures from the TREC-2004 Web Track [48] for a number of different algorithms. The results are sorted by MAP and the bracketed numbers 1 through 18 show the ranking of each algorithm with respect to MAP and P@10.

Algorithm	MAP	P@10
uogWebCAU150	0.179 (1)	0.249 (2)
MSRAmixed1	0.178 (2)	0.251 (1)
MSRC04C12	0.165 (3)	0.231 (3)
humW04rdpl	0.163 (4)	0.231 (4)
THUIRmix042	0.147 (5)	0.205 (6)
UAmsT04MWScb	0.146 (6)	0.209 (7)
ICT04CIIS1AT	0.141 (7)	0.208 (5)
SJTUINCMIX5	0.129 (8)	0.189 (11)
MU04web1	0.115 (9)	0.199 (9)
MeijiHILw3	0.115 (10)	0.153 (8)
csiroatnist	0.111 (11)	0.205 (12)
mpi04web01	0.106 (12)	0.177 (10)
VTOK5	0.101 (13)	0.135 (13)
fdwiedf0	0.090 (14)	0.117 (15)
wdf3oks0brr1	0.085 (15)	0.124 (14)
LamMcm1	0.049 (16)	0.087 (16)
irttil	0.018 (17)	0.029 (17)
XLDBTumba01	0.003 (18)	0.011 (18)

Chapter 4

Clustering Evaluation - QC4

Web Page Clustering methods find refinements in a different way from other search extensions. Instead of directly seeking query refinements, clustering methods organize the result set by grouping similar documents together into clusters; each cluster represents a different refinement of the search results and the set of refinements represent an overview or summary of the range of topics in the result set. This overview helps the user explore the result set and understand the variety of retrieved pages.

The search evaluation method described in chapter 3 is effective at comparing the refinement performance of different search extensions including clustering methods. However, it does not measure exploratory effectiveness and is therefore inadequate for properly evaluating clustering methods. A different evaluation method is required to compare clustering methods properly.

Many web page clustering evaluation methods exist [166, 203, 75, 167, 177, 121, 37]. Unfortunately, they are not effective at comparing web page clustering algorithms that produce structurally different clusterings, due to an intrinsic bias towards certain types of clustering. Additionally, existing evaluation methods produce spurious results on many boundary and extreme clusterings such as random clusterings.

This chapter discusses the desired characteristics of a method for web

page clustering evaluation and then identifies the problems with the existing methods of clustering evaluation. Then it presents QC4, a new web clustering evaluation method that addresses these problems by generalizing the "gold-standard" approach to use a new, richer structure for ideal clusterings and by developing new measures of quality and coverage.

4.1 Properties of a Good Evaluation Method

There are substantial differences between the characteristics desired from web page clustering and from clustering in other domains. In general, good clusterings are compact (members of each cluster should be homogeneous and as close to each other as possible) and separate (the clusters should be distinct and widely spaced) [75]. In contrast, good web clusterings are often loose (members of the same cluster can be very different using standard measures of similarity) and overlapping (some documents belong in multiple clusters). These differences make web page clustering more difficult and evaluation more challenging.

Web clusterings are often loose as many terms are irrelevant to the document's semantic interpretation and only the document's semantic interpretation is relevant to cluster membership. In particular, under standard distance measures, documents can be nearest neighbours by having many similar terms but have completely different semantic interpretations, and therefore belong in different clusters, due to subtle differences in relatively few terms [166]. This precludes the use of standard distance measures for web page clustering and evaluation.

Under the right measure of semantic distance, good web clusterings would be compact (though still overlapping) and evaluation easy. However, finding the right measure of semantic distance is a hard open research question and is the principal challenge of web page clustering. Despite this, it is necessary to evaluate web page clusterings, which entails having a reliable measure of semantic distance, and the most reliable

source of that information is currently humans. Therefore, a good web page clustering evaluation method requires human input.

After explaining the choice of a gold standard approach, this section explains the desired properties of an ideal clustering, the need for two measures (quality and coverage), and how aggregate results impact evaluation.

4.1.1 Methodology and Approach

There are two broad methodologies for evaluating clusterings. The internal methodology [75, 167] evaluates a clustering only in terms of a function of the clusters themselves. The external methodology [75, 167] evaluates a clustering using external information, such as an ideal clustering. When external information is available, an external methodology is more appropriate because it allows the evaluation to reflect performance relative to the desired output.

There are three main approaches to evaluation using the external methodology: gold-standard [177, 178], task-oriented [177], and user evaluation [203, 64]. Gold-standard approaches compare a manually constructed ideal clustering (ground truth or reference clustering) against the actual clustering. Task-oriented approaches evaluate how well some predefined task is solved. User evaluation approaches involve directly studying the usefulness for users and often involve observation, log file analysis, and user studies, such as those carried out in the user evaluation of Grouper [203].

Task-oriented methods, such as Search Result Reordering [203] and the evaluation method described in chapter 3, evaluate clustering performance relative to a single search goal and therefore do not evaluate the exploratory effectiveness of a clustering. This limitation is typically overcome by combining a task-oriented method with another evaluation method such as a user study [203]. However, user studies are very difficult to reproduce and cannot be reused to evaluate new algorithms, be-

cause they are dependent on the users. Additionally, the large cost and time involved in conducting good user studies is a significant drawback.

Therefore, the evaluation method uses external information in the form of an ideal clustering to define a gold-standard and measures a clustering against this ideal clustering.

4.1.2 Structural Flexibility of the Ideal Clustering

Gold standard methods evaluate clustering performance by measuring the similarity between a clustering and the gold standard, which is also known as the reference clustering or "ideal clustering". An ideal clustering is a set of clusters constructed by a human expert manually grouping the documents from the result set into semantically meaningful clusters that would be comprehensible to a user. To discriminate between the clusters from the clustering and the clusters from the ideal clustering, the latter are referred to as topics. Note that in the literature, topics are often called classes [151, 129]; this thesis does not use this terminology because classes are often disjoint [129], whereas topics may overlap.

To avoid bias, the structure of the ideal clustering should mimic the desired result of clustering and it should not be artificially constrained by the evaluation method or the capabilities of the evaluation measurements. The ideal clustering's structure must be quite flexible since clusterings can be structurally diverse as shown in figure 4.1. The clustering granularity may be coarse, so that there are just a few large clusters covering very broad topics, or fine, so that there are many small clusters covering very focused topics. The clusters may be disjoint and constitute a partition of the results, or the clusters may overlap, so that the same page may appear in several clusters. The clustering may be "flat" so that all clusters are at the same level, or the clustering may be hierarchical so that lower-level clusters are sub-clusters of higher-level clusters.

This is particularly relevant to web page clustering evaluation, because

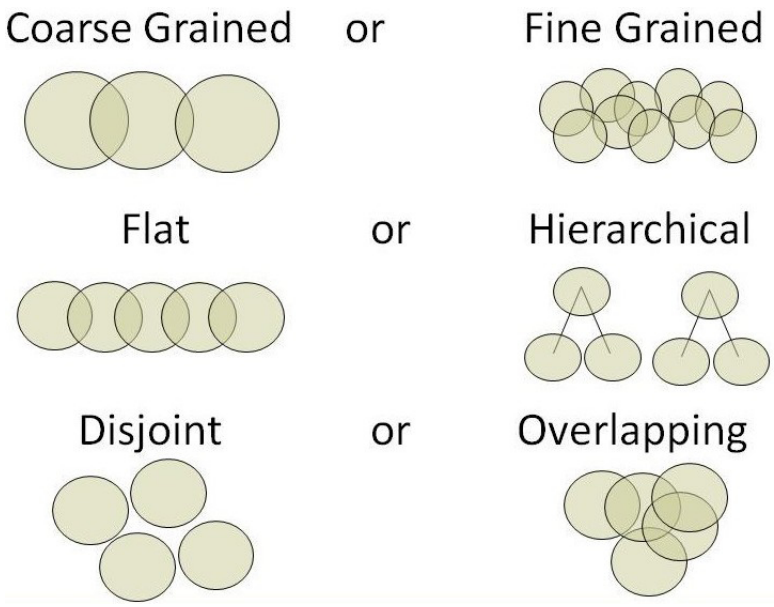

Figure 4.1: Different algorithms can produce structurally different clusterings

an individual text document may belong to many different topics and therefore a complete set of topics for a collection of documents must necessarily have overlapping topics. Requiring an ideal clustering to consist of a partition forces it to leave out many reasonable topics, and causes it to penalize an algorithm that happened to choose a different partitioning of the documents. Therefore, the evaluation method and measurements must account for the different structures, so there is no bias towards clusterings with certain characteristics.

4.1.3 Two Measures: Quality and Coverage

Quality and coverage are two complementary performance measures that compare a clustering to an ideal clustering. Quality measures the fidelity or accuracy of the information captured by the clusters in the clustering (or equivalently, measures how accurately the information from the ideal clustering is reproduced in the clustering). A high quality clustering is one

in which each cluster contains documents relating to a single topic from the ideal clustering. Coverage measures the completeness of the clustering (or equivalently, measures how much of the information from the ideal clustering is reproduced in the clustering). A high coverage clustering is one in which for every distinct topic in the ideal clustering, there exists a cluster that contains most of the documents from its respective topic.

Quality and coverage can measure performance at both a coarse level (entire clustering) and at a finer level (individual clusters). To discriminate between these, the quality of an individual cluster is referred to as "cluster quality" and the quality of an entire clustering is referred to as "clustering quality". Similarly, the coverage of an individual topic is referred to as "topic coverage" and the coverage of an entire ideal clustering is referred to as "clustering coverage".

There is often a trade off between quality and coverage — it is often possible to tune parameters so algorithms perform one well, at the cost of the other. Depending on their needs, users and applications weight the importance of quality and coverage differently. For instance, mobile users may want high quality, but accept low coverage because they do not want to spend time filtering out irrelevant results and are unlikely to use additional results because their time is limited; in contrast, researchers may want high coverage, but accept low quality because they want exhaustive information and are willing to spend time filtering out the irrelevant results. Therefore, good evaluation methods must measure quality and coverage separately. If a single measure of performance is desired, the measurements can be combined later to give a single application specific measurement.

4.1.4 Avoiding Bias from Aggregate Results

Clustering quality and clustering coverage implicitly (and often explicitly) aggregate the individual measures of cluster quality and topic cover-

age. Depending on the implementation of this aggregation, this inevitably leads to bias towards large clusters and large topics or towards small clusters and small topics. A good evaluation method should avoid bias or failing that, should make the bias and its magnitude explicit. For simplicity, as identical principles and arguments apply to topics and coverage, the remainder of this section will only address clusters and quality.

At one extreme are weighted measures, which weight each cluster's contribution according to its size. Weighted measures are justified because they give each document equal importance. However, weighted measures are biased towards the performance of large clusters because even large changes in the quality of small clusters make only a negligible difference in clustering quality.

At the other extreme are inverse weighted measures, which weight each cluster's contribution according to the inverse of its size. Inverse weighted measures are inherently biased towards small clusters. However, large clusters should be at least as important as small clusters and therefore inverse weighted measures are of no practical concern.

In between these extremes are the average measures, which weight each cluster's contribution equally. Average measures are justified because they give each cluster equal importance. However, average measures are biased towards the performance of small clusters when they significantly outnumber the large clusters. For example, when there are 20 small clusters and 2 large clusters, the majority of the pages (those in the large clusters) could be misclassified with only negligible impact on clustering quality. Note: average measures are not biased towards small clusters when the clustering is dominated by large clusters, e.g. 2 small clusters and 10 large clusters.

Aggregating individual measures of cluster quality inherently causes bias either towards large clusters (with weighted measures) or towards small clusters (with average measures in some situations). To make these biases explicit and to show the magnitude of the bias, a good evaluation

method should present both weighted measures and average measures of clustering quality. When both measures agree, all clusters have comparably good or bad performance. When the measures disagree, the magnitude of the difference between the performance of the small and the large clusters is reflected by the difference between the measures. When a single overall measure is required, the two measures can be combined together.

4.2 Properties of Good Measures of Quality and Coverage

Numerous measures of quality and coverage are conceivable, but only those meeting certain requirements are good measures of quality and coverage. This section describes the properties that good measures of quality and coverage should satisfy when comparing clusterings against ideal clusterings.

4.2.1 Must Measure Quality and Coverage

An obvious requirement is that the measures actually measure quality or coverage. For the quality measure, performance improves when the accuracy of the information captured by the clusters increases. Accuracy increases when the ratio of relevant to irrelevant pages increases because the user is less likely to encounter irrelevant pages within a cluster. For the coverage measure, performance improves when the quantity of information captured by the clusters increases. Information quantity increases when previously unrepresented pages are added because the user is less likely to run out of relevant pages.

Figure 4.2 depicts these basic properties. As in all the figures in this section, the circular objects represent clusters; the different colours represent the different topics of the pages in the cluster, and the bracketed numbers inside clusters represent the number of pages of a particular topic.

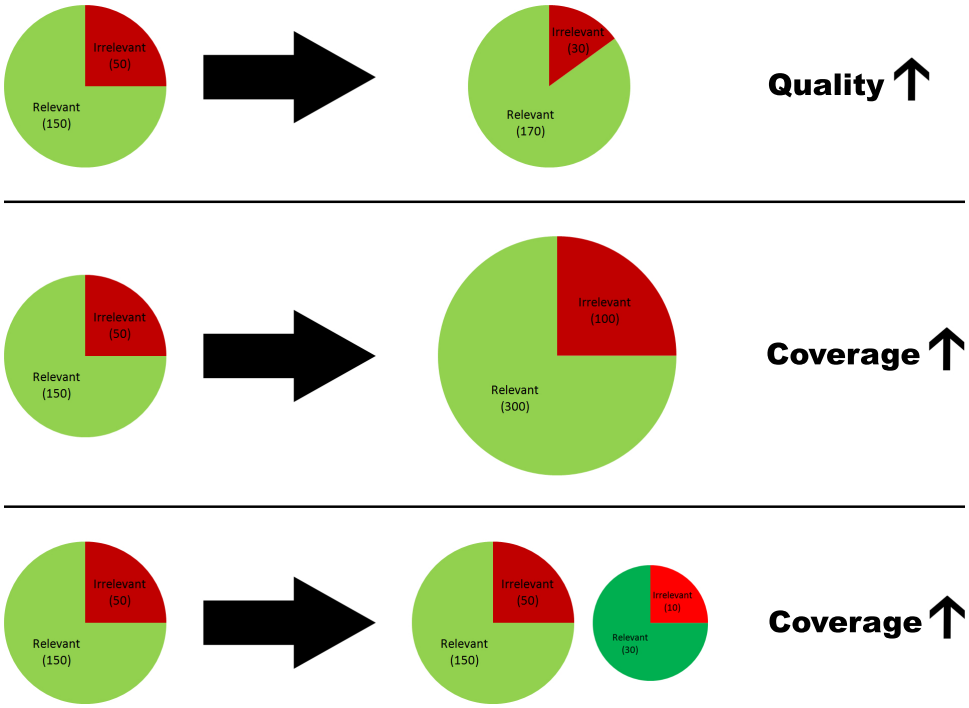

Figure 4.2: The basic properties of quality (first row) and coverage (second and third row) are that quality increases when the ratio of relevant to irrelevant pages increases and coverage increases when the number of distinct pages increases.

4.2.2 Perfect Clusterings

Another obvious requirement is that only perfect clusterings (those that exactly match the ideal clustering[1]) have perfect quality (100%) and perfect coverage (100%). Quality is less than 100% when any cluster contains irrelevant pages. Coverage is less than 100% when any document is absent from any cluster or when any topic is absent from the clustering. Figure 4.3 shows the properties of a perfect clustering and how its coverage and quality can become imperfect.

[1]Coverage is more complicated when the ideal clustering is hierarchical and there are additional situations where coverage is 100%; these situations are discussed later in this section.

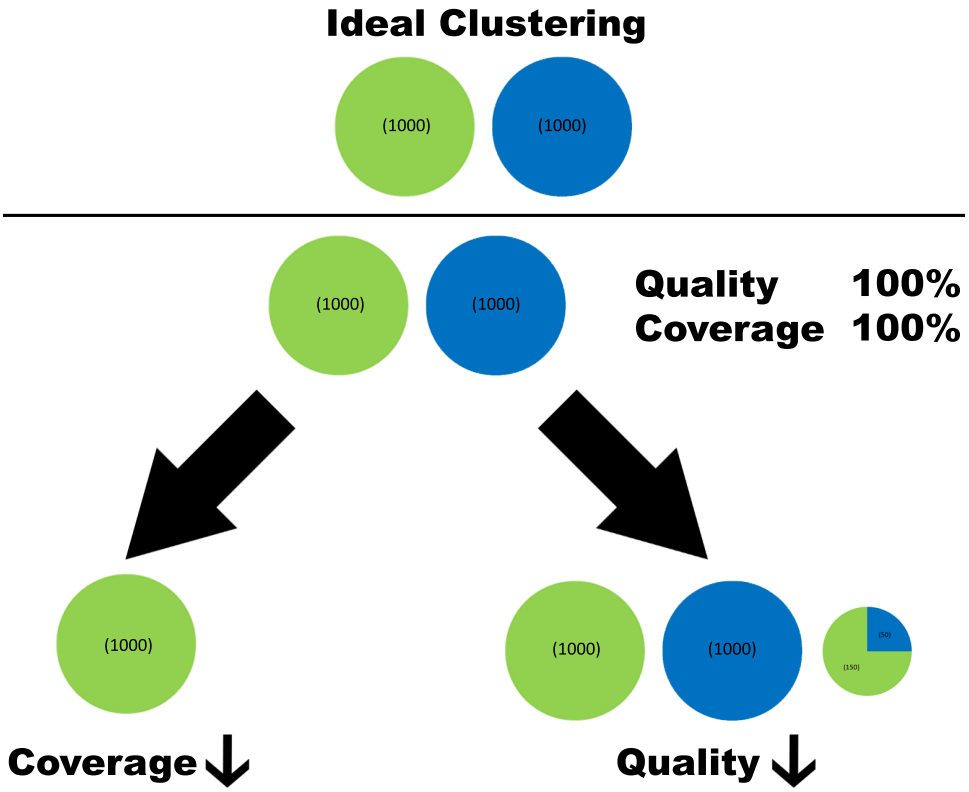

Figure 4.3: Quality and coverage are only 100% when the clustering is an exact match of the ideal clustering.

4.2.3 Worthless Clusterings

Some clusterings are worthless and should have 0% quality and 0% coverage. There are three types of clustering that are worthless: the singleton clustering, the giant cluster clustering, and the random clustering. All three are shown in figure 4.4. A singleton clustering is a bijection between pages and clusters (every document is in exactly one cluster and every cluster contains exactly one document). The giant cluster clustering has every page in one large cluster. The random clustering is where every cluster was formed by randomly selecting pages.

The singleton, giant, and random clusterings are intrinsically worthless because no 'clustering' was performed to obtain any of them. Clustering

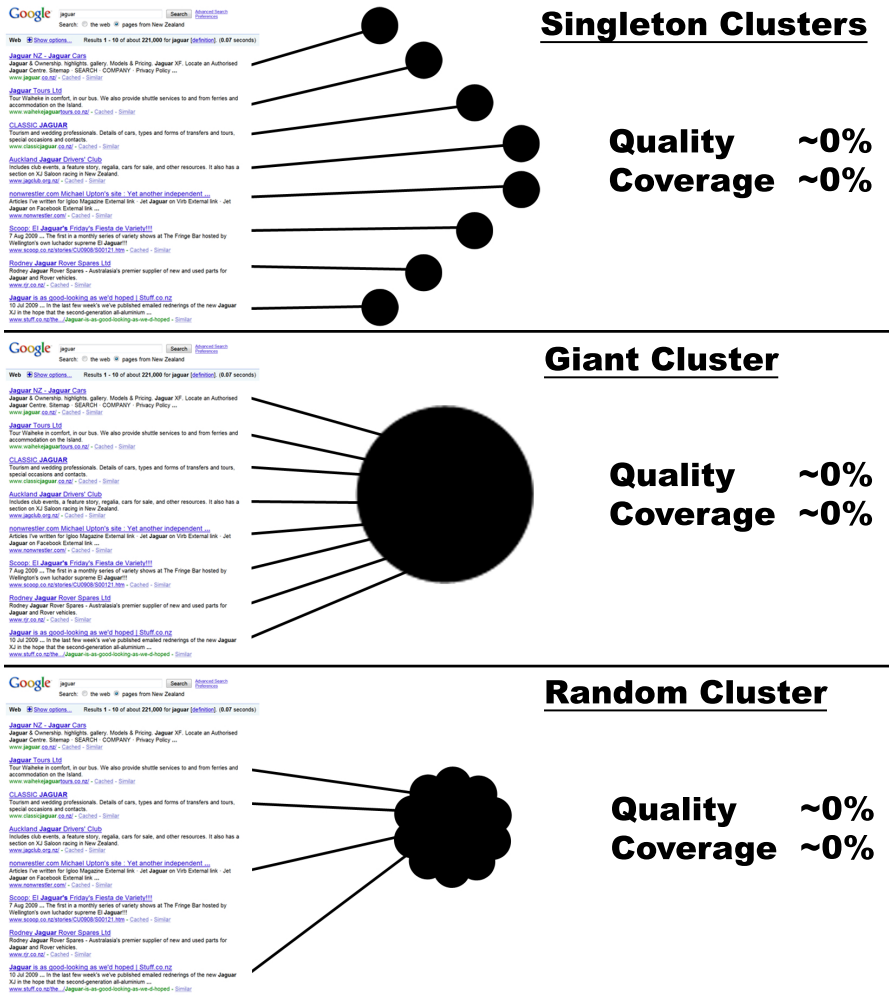

Figure 4.4: Three extreme/boundary cases where the clustering adds no value over the result set and both quality and coverage should be 0%.

involves grouping items according to some inter-item relationships, but these extreme clusterings reflect no relationship between the documents. From the user's point of view, a web clustering is only useful to the extent that it helps them find relevant documents. These three types of clustering are no more useful to the user than the original result set and therefore are completely worthless, which justifies them having 0% quality and 0%

coverage.[2] Random Clusterings are discussed further in section 4.2.8.

4.2.4 Cluster Composition

The composition of a cluster is determined by the distribution of its documents across topics. When clusters contain documents from different topics their quality and coverage should be adversely affected. Clusters are generally identified[3] by their most populous topic and only users seeking that topic are likely to examine the cluster and its documents. To those users, only the documents belonging to the cluster's most populous topic are relevant and hence from the perspective of evaluation, the most populous topic's documents are termed relevant and the remainder irrelevant.

Quality already considers cluster composition because it uses the ratio of relevant to irrelevant pages, which reduces quality when a cluster contains irrelevant pages (documents from different topics). However, this is not sufficient, because it is easier for users to distinguish the relevant documents from the irrelevant documents when the irrelevant documents are dispersed over fewer irrelevant topics. It is easier because users can use similarities between a new irrelevant document and prior documents as a shortcut to determine relevance more quickly than they could evaluate the relevance of a document that was unrelated to the prior documents. Additionally, as the number of categories of irrelevant document grows, the shortcut becomes less useful as users must remember and process more information to determine relevance. Therefore, as depicted in figure 4.5, the quality measure must consider the composition of the irrelevant pages and quality should be lower when the irrelevant pages are distributed across more topics.

The addition of new documents to a cluster increases the coverage of

[2] In practice, quality and coverage can only be near 0% for these extreme clusterings as some ideal clusters might only contain a single document and there are difficulties in identifying random clusters.

[3] labelled with a title or description

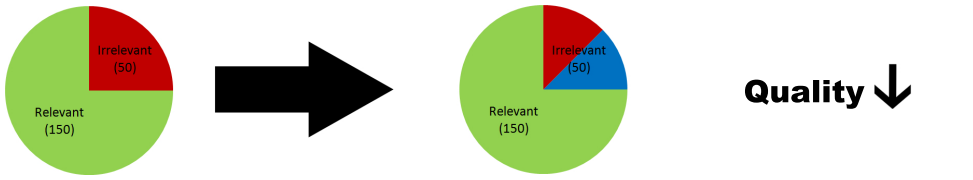

Figure 4.5: The composition of the irrelevant pages in a cluster affects quality.

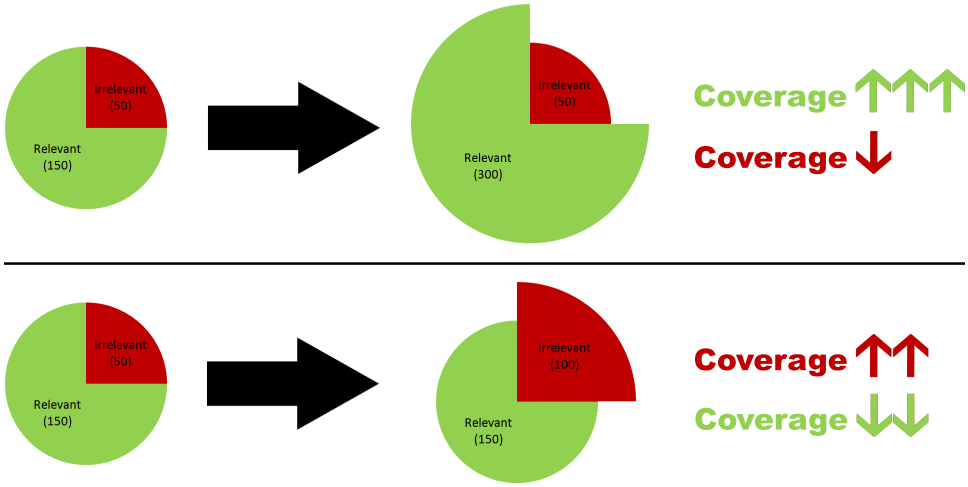

Figure 4.6: The composition of a cluster affects coverage.

those document's topics. However, pages are only useful if the right users can find them and therefore the coverage gained from a relevant page should be higher than the coverage gained from an irrelevant page. More generally, a page's contribution to coverage should be proportional to the relative size of the page's topic in the cluster. The effect of this, as depicted in figure 4.6, is that adding relevant documents increases coverage because both the number and proportion of relevant documents are increasing. While adding irrelevant documents does increase the coverage of the topics of those documents, it significantly lowers the coverage of the relevant documents' topic because it reduces the proportion of documents for that topic.

4.2.5 Segmented Clusters

Users expect different clusters to contain distinctly different documents. When a single topic is segmented into two or more clusters, effort is required to find all related clusters and to combine their results and this affects both quality and coverage.

Quality is affected because segmented clusters reproduce their ideal cluster less accurately — it is left up to the user to work out that the segments reflect the same topic. This disadvantage can be reflected in the quality measure by lowering the quality of clusters that are small relative to their topic. This has the effect of reducing quality when a cluster is split into two comparably composed clusters, as depicted in figure 4.7.

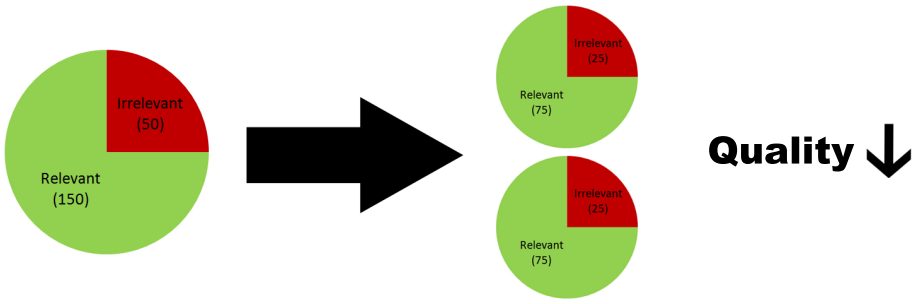

Figure 4.7: Splitting a cluster into two parts reduces quality.

Coverage is affected by segmented clusters too, because users may overlook some relevant clusters (possibly as they have already found another relevant cluster) and therefore miss the relevant documents in those clusters (documents they might have found if the clusters were not segmented). Furthermore, if the clusters are segmented too much, the work involved in identifying relevant clusters may be comparable to the work involved in identifying relevant pages from the original result set and if the clusters are particularly small, not much value will be gained by finding them and therefore they should not be considered covered.

The effect of segmentation on coverage depends on the number of

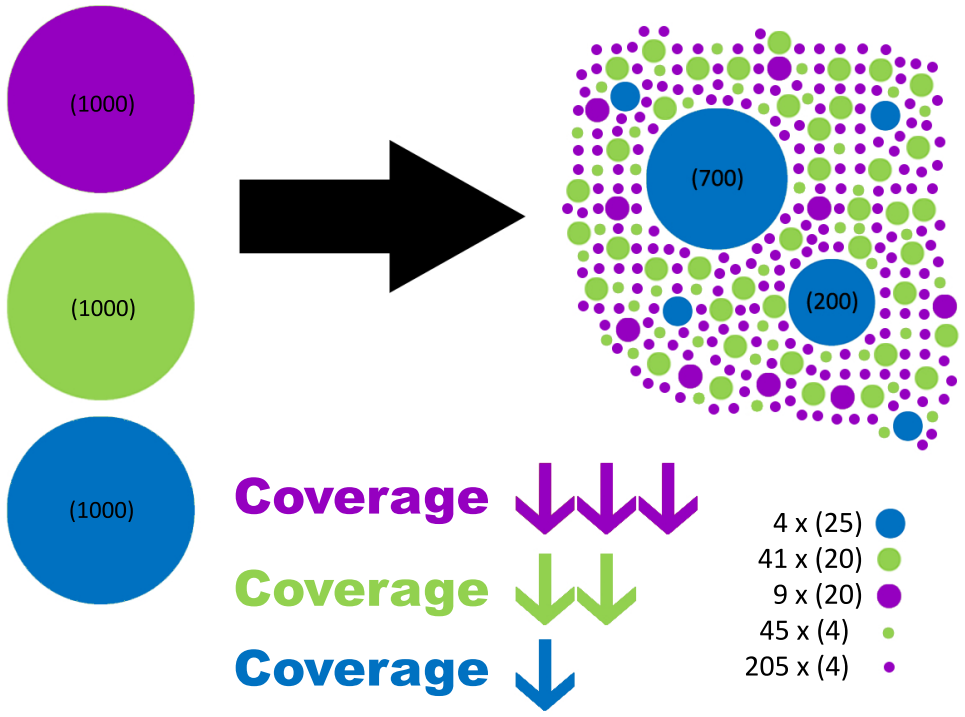

Figure 4.8: When there are many more clusters than topics, all but the largest clusters have relatively lower coverage because users must expend more effort to find the same quantity of information.

clusters and their comparable size as depicted in figure 4.8. The blue clusters are large and are relatively unaffected because they are easy to find due to their size. The green clusters are smaller, but still larger than the purple clusters, which makes them easier to identify and more valuable once identified. The purple clusters are numerous and small, which makes them comparatively harder to find because they are indistinguishable from other small clusters, and once found, only marginally more valuable than individual documents from the original result set.

4.2.6 Overlapping Clusters

Topics overlap when multiple topics contain the same document, which happens when documents (those in the overlap) relate to multiple topics. Since the documents in the overlap are relevant to users interested in any one of the topics, the clusters identified with these topics should also overlap. If the documents were in just a subset of the corresponding clusters, users interested in a topic whose corresponding cluster did not contain the documents would miss out on some relevant documents, meaning coverage should be lower, as shown in figure 4.9.

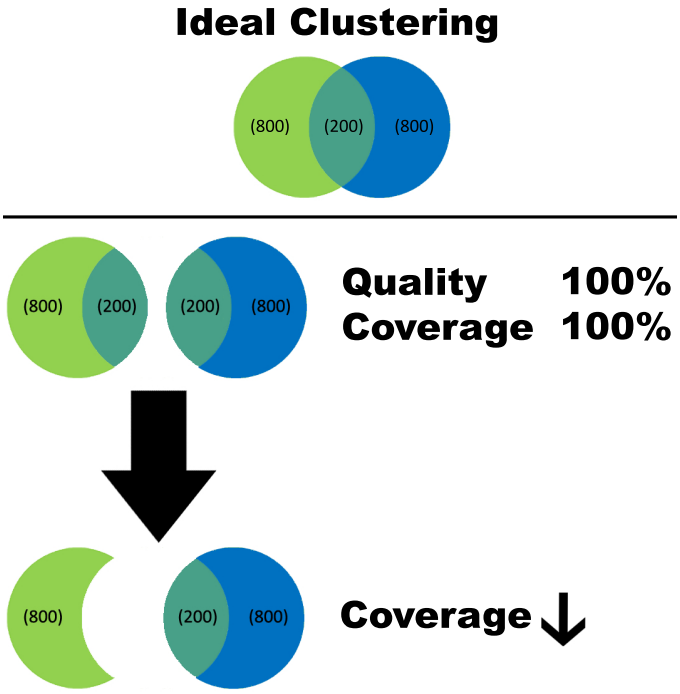

Figure 4.9: To have complete coverage documents must be in all relevant clusters, which means documents must appear in multiple clusters when topics overlap.

When the topics partition the documents, some properties hold that are not true in the more general setting where topics may overlap. Good measures of quality and coverage should avoid these properties so they

work correctly when topics and clusters overlap.

For example, when the topics partition the documents, clusters consisting of disparate topics never have 100% quality, but when topics overlap, clusters with 100% quality must necessarily contain documents from multiple topics. A more useful property that applies to both partitions and overlapping topics is that clusters never have 100% quality when they contain any pair of documents that do not share a common topic.

Similarly, when the topics partition the documents, a clustering can have 100% coverage when each document is in just one cluster, but when topics overlap, documents in the overlap must be in more than one cluster. In general, coverage measures must consider the coverage of individual topics to account for documents in multiple topics.

4.2.7 Hierarchical Clusterings

Hierarchical clusterings introduce a second kind of overlap: overlap between clusters at different levels of the hierarchy. The overlap between clusters of hierarchical clusterings must be treated differently from overlapping clusters because they have different semantics. With overlapping clusters, the overlap represents distinct information that belongs in multiple clusters. With hierarchical clusterings, the overlap represents the same information at different levels of detail or granularity. The examples of hierarchical clusterings in this section all relate to the ideal clustering shown in figure 4.10, which contains a 2-level hierarchy of topics.

A single cluster can only reflect one level of granularity because a cluster cannot be both broad and specific at the same time. To account for this, each cluster should be evaluated against just one level of the topic hierarchy[4] in isolation. The most appropriate level is the one containing the topic that is most similar to the cluster, as shown in figure 4.11.

Evaluating a cluster against the most appropriate level can mean it has

[4]Different clusters might be evaluated against different levels of the hierarchy.

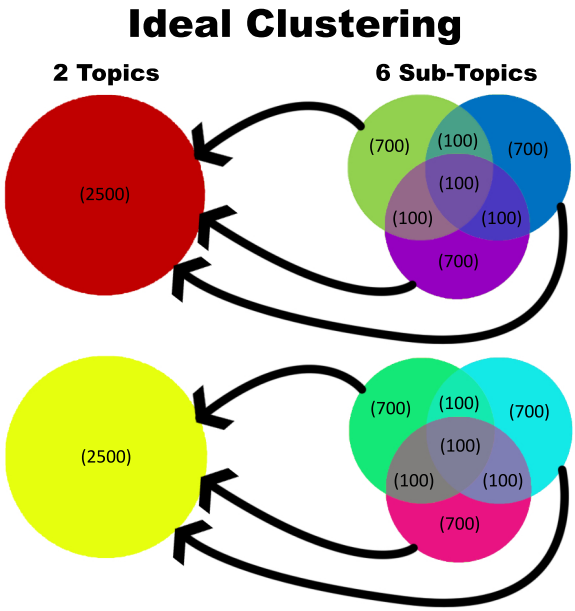

Figure 4.10: An ideal clustering with eight topics arranged in a hierarchy where the red and yellow topics each have three (overlapping) sub-topics.

lower quality than if it were evaluated against another (less appropriate) level. Figure 4.11 shows an example of this situation; had the cluster been compared to the red top-level cluster, it would have had perfect quality.

Beyond narrowing the focus to just a single level of the hierarchy, the quality measure is otherwise unaffected by a hierarchical clustering. In contrast to quality, coverage is significantly affected by hierarchical clusterings. Normally, perfect coverage requires every topic to be covered by a cluster. However, in a hierarchical clustering, topics at different levels represent the same information, so it is only necessary to cover each topic in one level of granularity to capture its information.[5]

Figures 4.12 and 4.13 illustrate four clusterings with perfect quality and perfect coverage. They have perfect quality because every cluster exactly

[5]There is no harm in covering the same topic at multiple levels of granularity, but once a topic has been covered, its coverage cannot be improved further by covering it again at a different level of granularity.

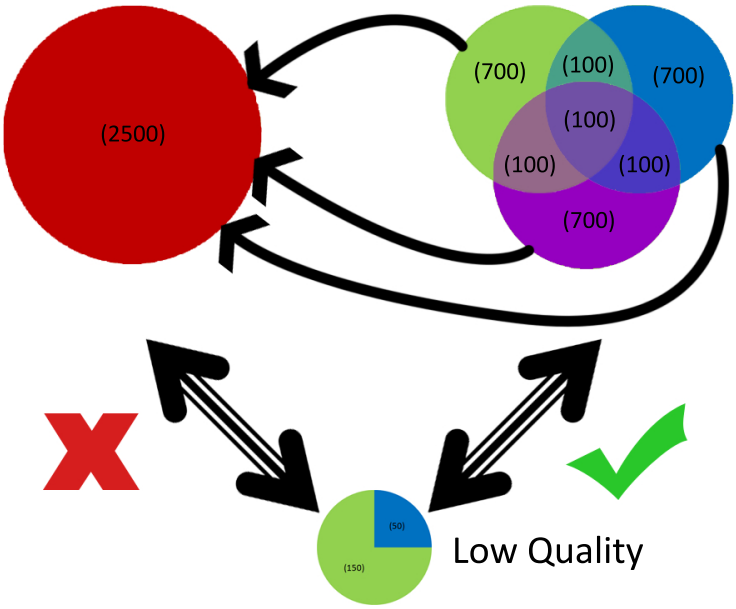

Figure 4.11: Clusters must be evaluated against the most comparable level of the hierarchy. Note that all the cluster's documents are in the red topic, but because it is more similar to the green and blue topics, it must be compared against those, and consequently, it has lower quality.

matches a topic in the ideal clustering. They have perfect coverage because every top-level topic is covered either by a cluster that covers the top-level topic directly or by clusters that cover all the sub-topics of that top-level topic.

4.2.8 Random Clusterings

It is plausible that a measurement could penalize a singleton or giant clustering because they are readily identified and can be singled out. In contrast, it is not as obvious that a measurement could distinguish a random clustering from any other clustering, because any clustering could have been randomly generated.

A random clustering is one in which the clusters have been constructed

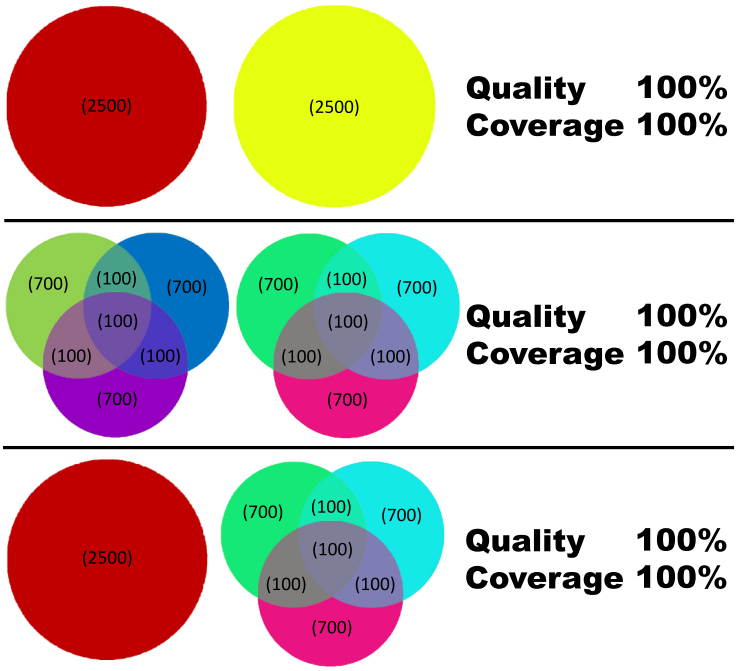

Figure 4.12: Three clusterings with perfect quality and perfect coverage.

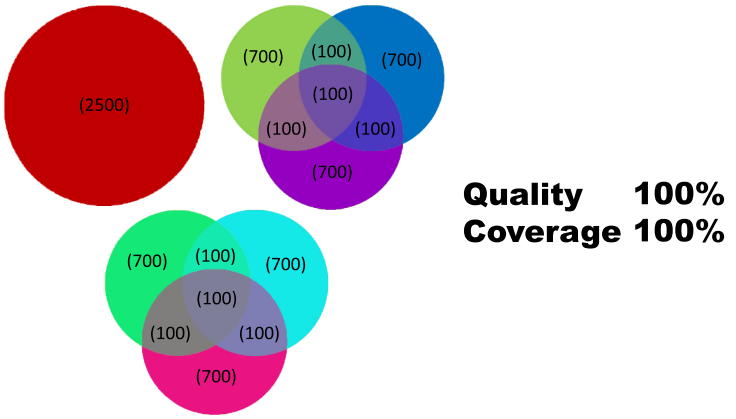

Figure 4.13: Another clustering with perfect quality and perfect coverage — covering documents at multiple levels is not a problem.

randomly. When a cluster is constructed randomly, every document has an equal probability of being included. When the documents are labelled (such as with topics), one can calculate the expected composition of a ran-

dom cluster. The expected composition of a random cluster can then be used as a guide to determine whether a cluster is random.

When a cluster, c, is constructed randomly, the probability of a document being in c is $\frac{|D_c|}{|D|}$, where D_c is the set of documents in cluster c and D is the set of documents. When the documents are partitioned[6] by a set of labels, B, the probability of a document having label b is $\frac{|D_b|}{|D|}$, where D_b is the set of documents with label b. Using these probabilities, a randomly generated cluster of size $|D_c|$ is expected to have $\frac{|D_b|}{|D|} \times |D_c|$ documents with label b and as it turns out, most randomly generated clusters have a very similar composition to the expected composition.

To test this hypothesis, I ran some simulations that randomly generated many clusters. In each simulation, 10,000,000 clusters were generated with a fixed number of documents drawn at random from a pool of labelled documents. Figures 4.14, 4.15, and 4.16 show the results of some of these simulations.

Each figure shows a 2d-transformation of cluster-space that focuses on the region containing random clusters. Each point represents a set of structurally identical clusters and the height (and colour) shows the frequency[7] at which different clusters occurred. Specifically, the x-axis represents the number of documents labelled 1 in the generated cluster and the y-axis represents the spread of documents with labels 2, 3, 4, and 5. (The spread is $(\max\{|D_2|, |D_3|, |D_4|, |D_5|\} - \min\{|D_2|, |D_3|, |D_4|, |D_5|\})$, where $|D_i|$ represents the number of documents labelled i.)

In the simulation shown in figure 4.14, one expects there to be $\frac{1000}{1100} \times 190 = 172.7$ documents labelled 1 and $\frac{25}{1100} \times 190 = 4.3$ documents each of labels 2, 3, 4, and 5. As shown in figure 4.14, the most common composition of a random cluster contained 173 documents labelled 1 and either 4

[6]The labels must partition the documents to ensure that $\sum_{b \in B}\{\frac{|D_b|}{|D|}\} = 1$.

[7]The frequency was normalized to account for the fact that 4 dimensions are being compressed into the y-axis and it was interpolated at some points on $y = 0$ and $y = 1$ where no clusterings exist.

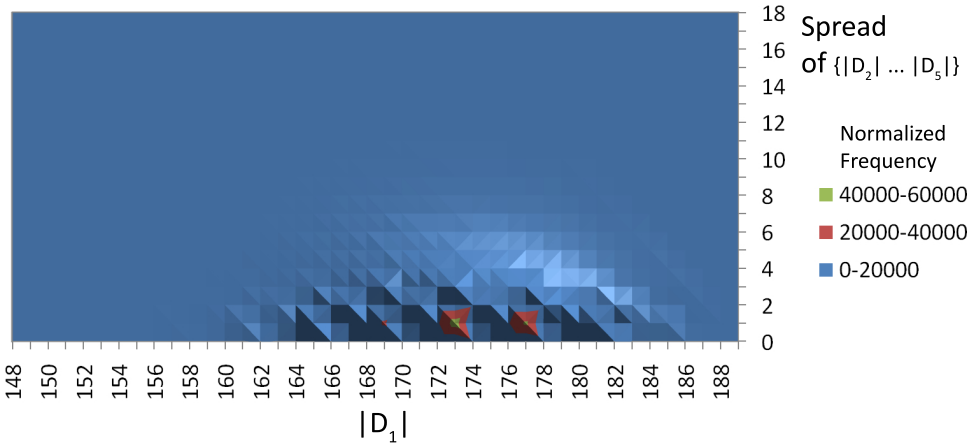

Figure 4.14: The distribution of random clusters containing 190 documents, selected from a pool with 1000 documents labelled 1 and 25 each of labels 2 to 5.

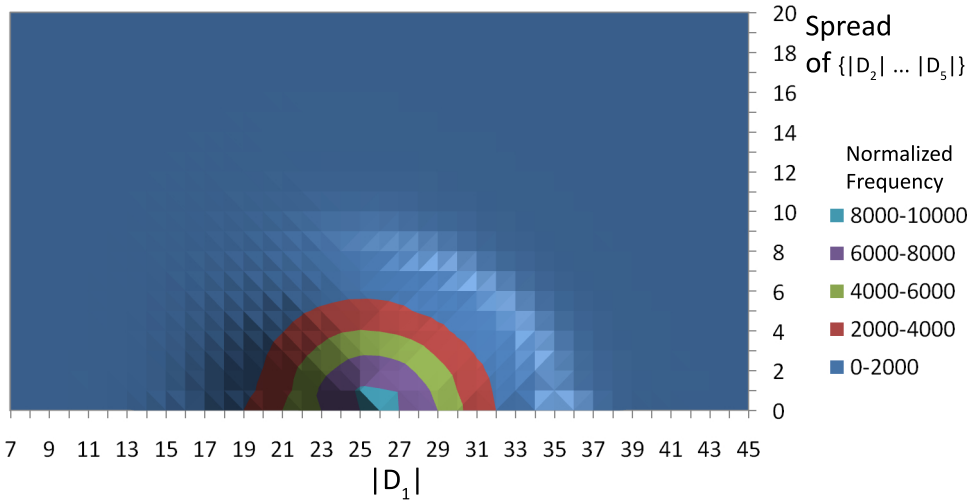

Figure 4.15: The distribution of random clusters containing 50 documents, selected from a pool with 100 documents labelled 1 and 25 each of labels 2 to 5.

or 5 documents each of labels 2, 3, 4, and 5 (giving a spread of 1) — exactly what was calculated as being the expected composition. The results match for the other examples too. Note that figure 4.16 has a hole at (2,1) because it is not possible to have a clustering with this configuration.

As seen in all three simulations, most random clusters are very similar

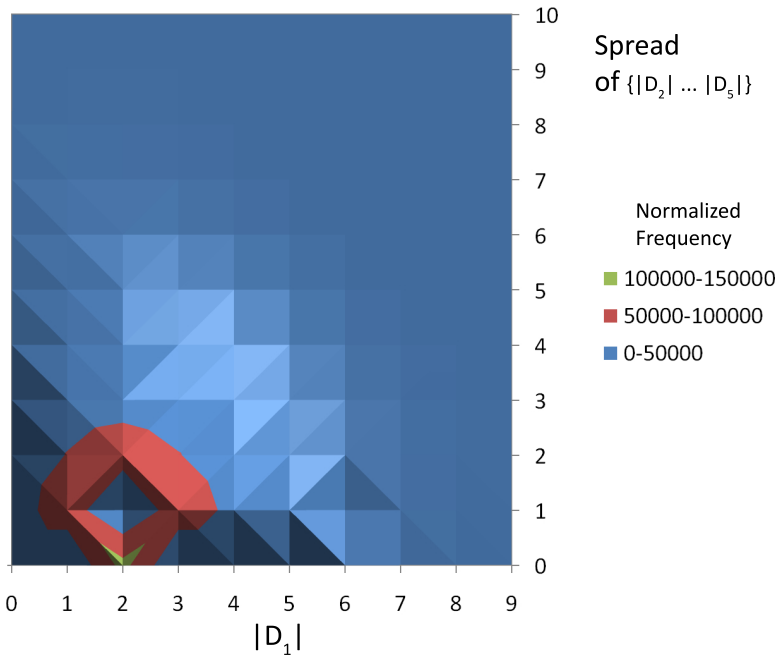

Figure 4.16: The distribution of random clusters containing 10 documents, selected from a pool with 25 documents labelled 1 and 25 each of labels 2 to 5.

to the expected composition and occupy a small part of the space (except when the clusters are small). Therefore, it is plausible that a measurement could penalize random clusters because they can be identified and distinguished from good clusters. Note that a non-random cluster that has a similar composition to the expected random cluster would be equally useless because it gives no more information than the original result set; penalizing it as a random cluster would not be a problem.

4.2.9 Limited User Time

In contrast to the properties discussed so far that apply in general to most domains[8], the impact of limited user time is domain specific. Limited user

[8]While unnecessary, the properties of overlapping clusters and hierarchical clusterings have no effect in domains without these kinds of clusters.

time only applies to applications like web page clustering, which have a user (typically human) that is unlikely to consume all the available information.

In the web page clustering domain, users have limited time to examine the cluster contents. Users are far less likely to look beyond the 1000th document than they are to look past the 10th document (section 2.2). More generally, as the absolute cluster size increases, the probability of a user viewing any particular page in the cluster decreases. The quality of a cluster should reflect the diminishing returns of increasing cluster size and therefore there should be relatively more value in increasing the size of small clusters than in increasing the size of large clusters, as depicted in figure 4.17.

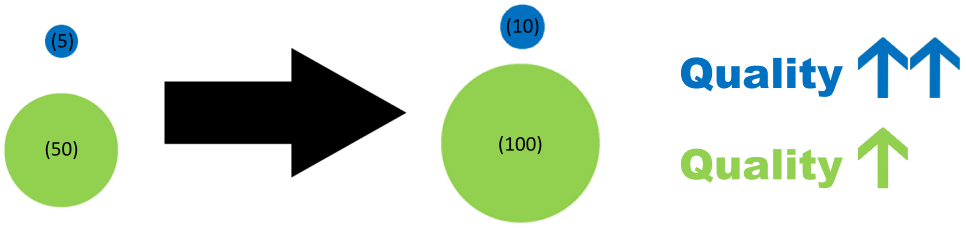

Figure 4.17: Increasing any cluster's size increases its quality (as described in section 4.2.5), but there is greater benefit in increasing the size of smaller clusters because users have limited time.

4.3 Existing Measurements

There are many ways of measuring quality and coverage against an ideal clustering. Some measurements evaluate individual clusters, while others evaluate entire clusterings, and some measurements evaluate just quality or just coverage, while others provide a combined measure of quality and coverage. This section introduces the most commonly used measurements for evaluating web page clusterings.

The rest of this chapter uses the following notation:
C is a set of clusters
T is a set of topics (the clusters of the ideal clustering)
D is a set of pages
c, t, and d are individual elements of C, T, and D respectively
D_c is the pages in cluster c
D_t is the pages in topic t
$D_{c,t}$ is the pages in both cluster c and topic t
C_d is the set of clusters containing page d
$C_t = \{c_i | \operatorname{argmax}_{t_j}(|D_{c_i,t_j}|) = t\}$ is the set of clusters that best match topic t

4.3.1 Quality Measurements

The Precision [174, 145, 187], $P(c,t)$, of a cluster relative to a topic is the fraction of the pages in the cluster that are also in the topic.

$$P(c,t) = \text{Precision} = \frac{|D_{c,t}|}{|D_c|}$$

The Purity [169, 167], $P(c)$, of a cluster is the Precision of the cluster relative to its best matching topic (the topic that the cluster most closely resembles).

$$P(c) = \text{Purity} = \max_{t \in T}\{P(c,t)\}$$

The Entropy[9] [208, 166, 189], $E(c)$, of a cluster is based on information theory [113] and it is the average "narrowness" of the distribution of the pages of the cluster over the topics. More precisely, it is the amount of information required to refine a cluster into the separate topics it represents.

$$E(c) = \text{Entropy} = -\sum_{t \in T} P(c,t) \log_{|T|} P(c,t)$$

[9]It is typical to use a base $|T|$ logarithm in the context of clustering evaluation to normalize $E(c)$ to $[0,1]$.

4.3.2 Coverage Measurements

The Recall [174, 145, 187], $R(c,t)$, of a cluster relative to a topic is the fraction of the pages in the topic that are also in the cluster.

$$R(c,t) = \text{Recall} = \frac{|D_{c,t}|}{|D_t|}$$

Just as Purity extends Precision to evaluate a cluster without reference to a specific topic, Recall can be extended to evaluate a topic without reference to a cluster.

The Best Recall, $R^1(t)$, of a topic is the Recall of the topic relative to its best matching cluster. Recall can also be extended in a different way: the Total Recall, $R^{\Sigma}(t)$, of a topic is the total coverage of the topic among all clusters that best match that topic.

$$R^1(t) = \text{Best Recall} = \max_{c \in C}\{R(c,t)\}$$

$$R^{\Sigma}(t) = \text{Total Recall} = \frac{|\bigcup_{c \in C_t} D_{c,t}|}{|D_t|}$$

4.3.3 Combined Measurements

In contrast to the previous measurements, which measure either quality or coverage, the measurements in this section combine the evaluation of quality and coverage into a single measurement, which is not as good, because it limits the usability of these measurements as discussed in section 4.1.3.

The F-measure [166, 211, 165], $F(c,t)$, of a cluster combines Precision and Recall. Typically, Precision and Recall are given equal weight, although it is possible to weight them unevenly.

$$F(c,t) = \text{F-measure} = \frac{2 \cdot P(c,t) \cdot R(c,t)}{P(c,t) + R(c,t)}$$

As with Precision and Recall, the F-measure can be extended to avoid referencing a specific cluster or specific topic, by computing the F-measure

of the cluster relative to its best matching topic ($F(c)$) or to its best matching cluster ($F(t)$).

$$F(c) = \text{Cluster F-measure} = \max_{t \in T}\{F(c,t)\}$$

$$F(t) = \text{Topic F-measure} = \max_{c \in C}\{F(c,t)\}$$

Mutual Information [169, 167, 167], MI, is based on information theory [113] and it is an average of a measure of correspondence between each possible cluster–topic pair.

$$MI = \frac{2}{|D|} \sum_{c \in C} \sum_{t \in T} |D_{c,t}| \log_{|C||T|}(\frac{|D_{c,t}||D|}{|D_c||D_t|})$$

4.3.4 Overall Measurements

While Mutual Information provides a combined measure of overall clustering quality and clustering coverage, the other measures (Purity, Entropy, Recall, and F-measure) provide measures of individual cluster quality and topic coverage. To transform these measurements into overall measures of clustering quality and clustering coverage that work without reference to a specific cluster or topic, they need to be averaged over all the clusters or topics; this can be done in a weighted or un-weighted manner [166] and this thesis uses the terms Weighted and Average to distinguish the two methods. Note that in the literature, sometimes the weighted average is termed micro-averaging and the un-weighted average is termed macro-averaging [27, 129].

Average Precision (average purity over clusters), Weighted Precision (cluster size weighted average purity over clusters), Average Entropy (average over clusters), and Weighted Entropy (cluster size weighted average over clusters) measure overall clustering quality.

$$AP = \text{Average Precision} = \frac{\sum_{c \in C} P(c)}{|C|}$$

$$WP = \text{Weighted Precision} = \frac{\sum_{c \in C} P(c)|D_c|}{\sum_{c \in C} |D_c|}$$

$$AE = \text{Average Entropy} = \frac{\sum_{c \in C} E(c)}{|C|}$$

$$WE = \text{Weighted Entropy} = \frac{\sum_{c \in C} E(c)|D_c|}{\sum_{c \in C} |D_c|}$$

Average Recall (average over topics) and Weighted Recall (topic size weighted average over topics) [44] measure overall clustering coverage.

$$AR^1 = \text{Average Best Recall} = \frac{\sum_{t \in T} R^1(t)}{|T|}$$

$$WR^1 = \text{Weighted Best Recall} = \frac{\sum_{t \in T} R^1(t)|D_t|}{\sum_{t \in T} |D_t|}$$

$$AR^{\Sigma} = \text{Average Total Recall} = \frac{\sum_{t \in T} R^{\Sigma}(t)}{|T|}$$

$$WR^{\Sigma} = \text{Weighted Total Recall} = \frac{\sum_{t \in T} R^{\Sigma}(t)|D_t|}{\sum_{t \in T} |D_t|}$$

Average F-measure (average F-measure over clusters) and Weighted F-measure (cluster size weighted average over clusters) provide combined measures of overall clustering quality and clustering coverage. As F-measure combines quality and coverage, it can be averaged over either clusters (AF^C and WF^C) or topics (AF^T and WF^T).

$$AF^C = \text{Average Cluster F-measure} = \frac{\sum_{c \in C} F(c)}{|C|}$$

$$WF^C = \text{Weighted Cluster F-measure} = \frac{\sum_{c \in C} F(c)|D_c|}{\sum_{c \in C} |D_c|}$$

$$AF^T = \text{Average Topic F-measure} = \frac{\sum_{t \in T} F(t)}{|T|}$$

$$WF^T = \text{Weighted Topic F-measure} = \frac{\sum_{t \in T} F(t)|D_t|}{\sum_{t \in T} |D_t|}$$

Note that in contrast to the other measurements, a lower Entropy value is better than a higher value and the best possible Entropy value is 0. As a reminder of this inversion, AE and WE will be written as $AE^{\Diamond}$ and $WE^{\Diamond}$ respectively.

4.3.5 Pair Counting Measurements

Pair counting measurements [121] are a third type of measurement that differ from the information theoretic (e.g. Entropy and Mutual Information) and set matching measurements (e.g. Precision, Recall, and F-measure) discussed so far. Pair counting measurements count the number of document pairs on which two clusterings agree or disagree.

Pair counting measurements compute counts of the number of document pairs:

N_{11} pairs in both the same cluster and the same topic

N_{00} pairs in both different clusters and different topics

N_{10} pairs in the same cluster, but different topics

N_{01} pairs in different clusters, but the same topic

They then combine these counts to produce a measurement. Specific pair counting measurements include the Rand Index [142], the Fowlkes-Mallows Index [65], the Jacard Index [126], and the Mirkin Metric [127].

$$\text{Rand} = \frac{N_{11} + N_{00}}{|D|(|D| - 1)/2}$$

$$\text{Fowlkes-Mallows} = \frac{N_{11}}{\sqrt{(N_{11} + N_{10})(N_{11} + N_{01})}}$$

$$\text{Jacard} = \frac{N_{11}}{N_{11} + N_{10} + N_{01}}$$

$$\text{Mirkin} = 2(N_{10} + N_{01})$$

Since pair counting measurements assume the clusters and topics are partitions of the documents, they are not suitable for evaluating overlapping and hierarchical clusterings. Even worse, as Meila [121] points out, these measurements have wildly variable baseline performance and it is not clear whether linearity exists after normalization, which makes it somewhat dubious to compare clusterings using pair counting measurements.

In practice, pair counting measurements are rarely used to evaluate web page clustering algorithms. Due to the significant drawbacks of using pair counting measurements to evaluate clusterings, they are not considered further in this thesis.

4.3.6 Other Measurements

This section has introduced the evaluation measurements most commonly used to evaluate and compare web page clusterings. There are a multitude of possible adaptations of these measurements as well as other measurements such as L, H, and D [121]. However, these adaptations and alternative measurements are very similar to the measurements already introduced.

My experiments with adaptations on Precision, Recall, and F-measure found no notable difference from the more common variations already discussed. L, H, and D are set matching measurements (match each cluster to an individual topic) and suffer from the matching problem [121], which occurs when a measurement ignores the unmatched part (the irrelevant documents) of a cluster or ignores some unmatched clusters or topics entirely. The set matching problem also affects Precision, Recall, and F-measure (other set matching measurements) and is observed in failures of the basic quality, basic coverage, composition, segmentation, and perfect clustering tests in section 4.4. These similarities make it redundant to discuss these adaptations and alternative measurements and they are not considered

further in this thesis.

4.4 Synthetic Evaluation of Existing Measurements

To test the measurements described in section 4.3, I constructed a suite of synthetic clusterings that test the characteristics of good evaluation measurements as described in section 4.2. This section describes the results of these experiments and provides examples of cases where the existing measurements fail.

4.4.1 Measures Quality and Coverage

Most of the existing measurements meet the most basic requirement and measure quality and/or coverage, but Best Recall and F-measure fall short. Figure 4.18 shows an ideal clustering (which will be used throughout this section), figure 4.19 shows five clusterings of the documents from the ideal clustering, and table 4.1 shows the results for each of the measurements.

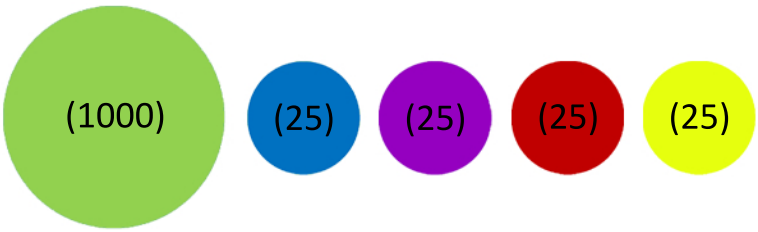

Figure 4.18: An ideal clustering with five topics: four small topics of equal size and one large topic.

Clustering 4.19.E has perfect quality since every cluster exactly matches a topic, while the other four clusterings have imperfect quality because each contains imperfect clusters. Precision and Entropy meet the basic requirement of measuring quality (to measure how accurately a cluster represents a topic) and correctly rate 4.19.E as perfect and 4.19.D as imperfect. However, Precision and Entropy do not account for segmented clusters

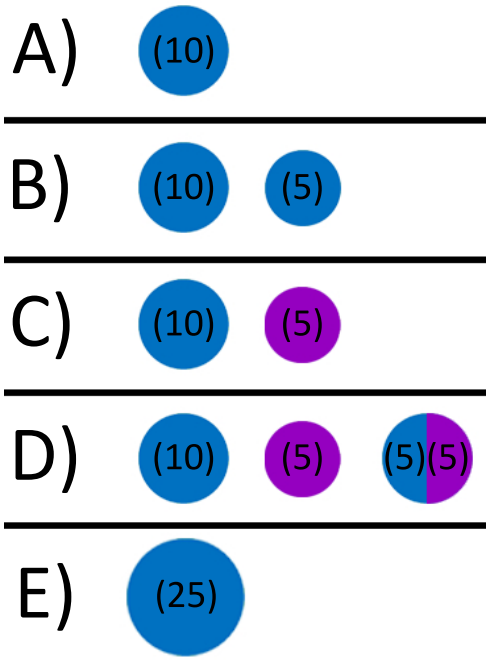

Figure 4.19: Five clusterings of the documents in figure 4.18. A provides a baseline. B and C have greater coverage than A. D has greater coverage than B and C, but lower quality. E has perfect quality and the best coverage.

(as section 4.4.6 will show), and consequently, they both incorrectly rate 4.19.A, 4.19.B, and 4.19.C as perfect.

All five clusterings have poor coverage since they leave out the vast majority of the documents and two or more of the topics (including the largest topic). The coverage measures rate them lower, but Best Recall fails to measure the increased coverage of clustering 4.19.B relative to 4.19.A because it only considers the best cluster for each topic and therefore ignores the coverage provided by the smaller cluster.

The combined measures are more difficult to interpret because they combine quality and coverage and each measure implicitly weights quality and coverage differently. Cluster F-measure tends towards quality, Topic F-measure tends towards coverage, and Mutual Information tends towards coverage. This bias explains why it is valid for Mutual Informa-

Table 4.1: Synthetic test case results for the five clusterings in figure 4.19

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.19.A	1	1	0	0	
4.19.B	1	1	0	0	
4.19.C	1	1	0	0	
4.19.D	0.833	0.8	0.144	0.172	
4.19.E	1	1	0	0	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.19.A	0.08	0.009	0.08	0.009	
4.19.B	0.08	0.009	0.12	0.014	
4.19.C	0.12	0.014	0.12	0.014	
4.19.D	0.12	0.014	0.2	0.023	
4.19.E	0.2	0.023	0.2	0.023	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.19.A	0.571	0.571	0.114	0.013	0.043
4.19.B	0.452	0.492	0.114	0.013	0.045
4.19.C	0.452	0.492	0.181	0.021	0.045
4.19.D	0.397	0.41	0.181	0.021	0.059
4.19.E	1	1	0.2	0.023	0.107

tion to be higher for clusterings 4.19.B and 4.19.C and higher again for 4.19.D, while it is also valid for Cluster F-measure to be lower for 4.19.B and 4.19.C and lower again for 4.19.D — the coverage of 4.19.B and 4.19.C are higher than 4.19.A and 4.19.D is higher again, while the quality of 4.19.B and 4.19.C are lower (due to segmentation[10]) than 4.19.A and 4.19.D is lower again.

Like Best Recall, Topic F-measure ignores the coverage provided by the smaller cluster in 4.19.B and therefore fails to measure the increased

[10]described in section 4.2.5

coverage. Additionally, Topic F-measure ignores the low quality cluster in 4.19.D and therefore fails to measure the decreased quality. Cluster F-measure also fails to measure coverage, because it considers 4.19.E to be perfect, yet the clustering fails to cover most topics.

4.4.2 Size Bias

When results are aggregated there is an inevitable bias towards either small or large clusters and towards either small or large topics (as discussed in section 4.1.4). When examining just a single measure, this bias can lead us to believe one of two falsehoods:

- that two clusterings are comparably good, when in fact they have radically different utility
- that two clusterings are radically different in performance, when in fact they are very comparable

This conundrum can only be resolved by examining at least two measures, one biased towards small clusters or topics and one biased towards large clusters or topics.

Precision, Recall, F-measure, and Entropy can all be aggregated in a weighted or un-weighted manner and consequently the issues of cluster and topic size bias can be resolved by looking at both the average and weighted measures. In contrast, Mutual Information is just a single measure and therefore does not permit the distinction between small or large clusters and small or large topics.

Figure 4.20 shows three reasonable clusterings of high quality, but varying coverage and table 4.2 shows the results for each measurement.

Although the clusterings are very accurate and therefore have high quality, they do not have perfect quality due to segmentation. However, Precision and Entropy incorrectly rate all the clusterings as perfect, because they do not account for segmentation (as section 4.4.6 will show).

The utility of a clustering depends on the user's needs (depth *vs* breadth). Average Recall suggests that clusterings 4.20.B and 4.20.C are equally good, while Weighted Recall suggests that 4.20.B and 4.20.C are radically different in performance; the opposite is true when considering 4.20.A and 4.20.B. Interpreting the result for a given user correctly requires both Average Recall and Weighted Recall. Mutual Information permits no distinction between clusterings 4.20.B and 4.20.C and therefore the result cannot be interpreted with respect to the user's needs.

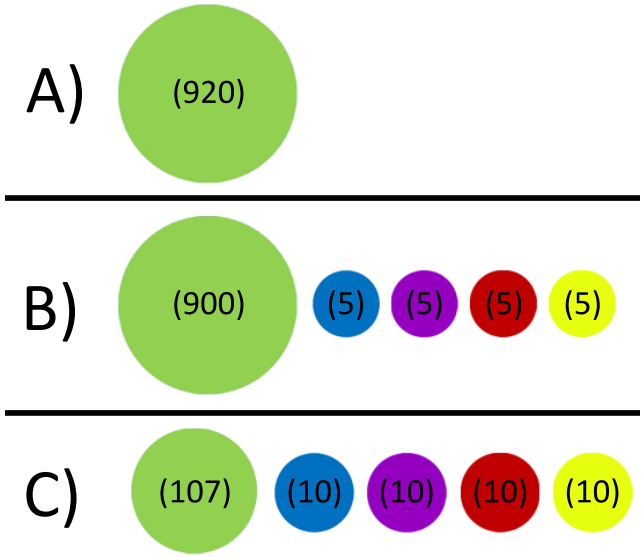

Figure 4.20: Three reasonable clusterings of the documents in figure 4.18. A misses the smaller topics and some documents from the large topic. B covers some of the small topics and misses more documents from the large topic. C covers more of the small topics and misses many documents from the large topic.

Other clusterings (where the quality of small and large clusters differs) show similar results for the average and weighted pairs of Precision and Entropy. The results for F-measure mirror those of Recall for clusterings 4.20.B and 4.20.C and the results for Topic F-measure mirror those

Table 4.2: Synthetic test case results for the three reasonable clusterings in figure 4.20

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.20.A	1	1	0	0	
4.20.B	1	1	0	0	
4.20.C	1	1	0	0	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.20.A	0.184	0.836	0.184	0.836	
4.20.B	0.34	0.836	0.34	0.836	
4.20.C	0.341	0.134	0.341	0.134	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.20.A	0.958	0.958	0.192	0.871	0.099
4.20.B	0.456	0.934	0.456	0.892	0.091
4.20.C	0.496	0.296	0.496	0.228	0.091

of Recall for 4.20.A and 4.20.B.[11] The Cluster F-measure results for 4.20.A may be surprising because they are quite different from those for Topic F-measure and Recall. The results differ because Cluster F-measure takes the average over clusters (of which there is just one in 4.20.A), whereas Topic F-measure and Recall take the average over topics (of which there are five in the ideal clustering).

4.4.3 Perfect Clusterings

Only perfect clusterings should get the best possible value as explained in section 4.2.2. Excluding the cases where Precision and Entropy gave perfect quality to segmented clusters, most of the existing measurements give

[11] except for a small discrepancy that occurs due to the differences between F-measure and Recall — though with small changes to the clusterings, this discrepancy could be eliminated

the best possible values to perfect clusters and lower values to less than perfect clusters. As shown in figure 4.21, there are three exceptions among the existing measurements: Mutual Information, Cluster F-measure, and Topic F-measure.

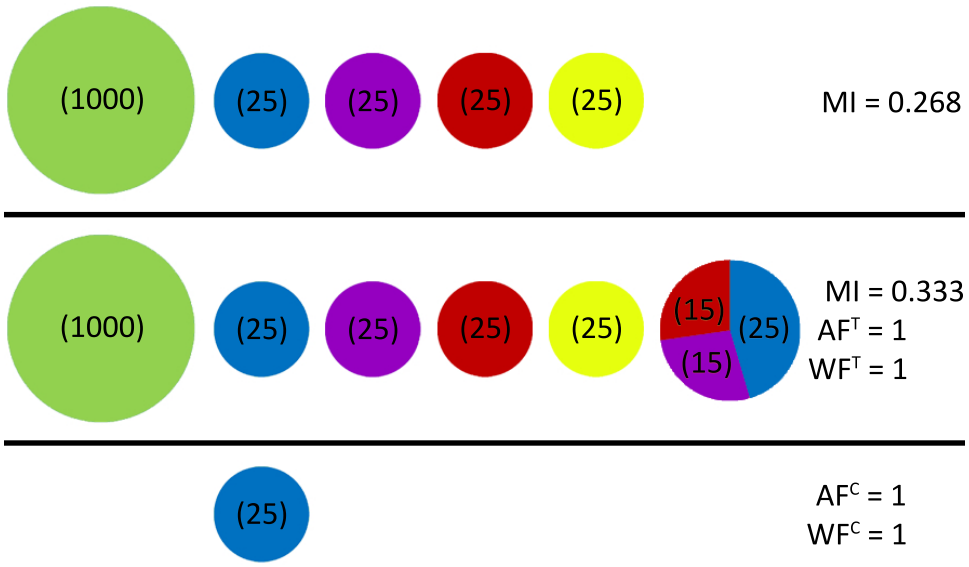

Figure 4.21: A case where Mutual Information gives a lower value to the perfect clustering than it gives to an inferior clustering and where F-measure gives perfect values to imperfect clusterings.

Mutual Information can give a better value to an imperfect clustering than it gives to a perfect clustering. This problem stems from the fact that Mutual Information assumes the clusters partition the pages — none of the other measurements considered have this restriction. This problem with Mutual Information is compounded by another limitation of Mutual Information: its maximum value is variable and is dependent on the ideal clustering. This is in contrast to Precision, Recall, and F-measure that have a best possible value of 1 and Entropy that has a best possible value of 0.

Cluster F-measure and Topic F-measure can give the maximum value (1) to clusterings with imperfect coverage and imperfect quality respectively. This problem occurs because Cluster F-measure and Topic F-measure

do not necessarily consider all the topics and clusters respectively, which is necessary when evaluating coverage and quality respectively. The problem occurs for Cluster F-measure when all the clusters are perfect, but the clustering as a whole is imperfect by not covering all the topics from the ideal clustering. Similarly, the problem occurs for Topic F-measure when the clustering is a superset of the ideal clustering, but the clustering as a whole is imperfect by containing imperfect clusters, which makes it harder for the user to find the good clusters.

4.4.4 Worthless Clusterings

As discussed in section 4.2.3, there are three main types of worthless clustering: singleton, giant, and random. Table 4.3 shows the evaluation results of the three worthless clusterings shown in figure 4.22 and the two reasonable clusterings shown in figure 4.23 — all are being compared against the ideal clustering shown in figure 4.18.

The singleton clustering consists of one cluster for each and every document in the ideal clustering. The giant clustering consists of a single cluster containing every document in the ideal clustering. The random clustering consists of 5 clusters, each containing 190 documents (170 documents from the large cluster and 5 documents from each of the four small clusters) — this cluster represents one that is likely to result from a random clustering algorithm as discussed in section 4.2.8. Ideally, the measurements should give a maximally bad value to these three clusterings (that is, 1 for $AE^{\Diamond}$ and $WE^{\Diamond}$ and 0 for all the others) and they should certainly give a worse value to these clusterings than to the reasonable clusterings shown in figure 4.23.

Clustering 4.23.A has higher coverage than 4.23.B because it enables the user to find relatively more documents from the small topics with minimal effort — documents that would have been particularly hard to find in the original result set. Clustering 4.23.B obviously has higher quality,

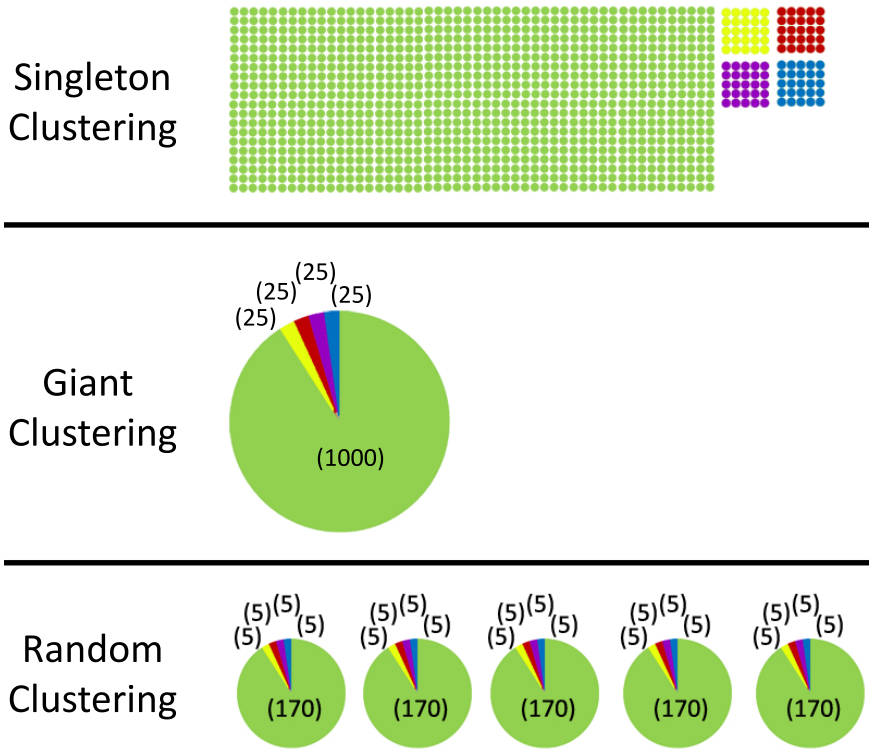

Figure 4.22: Three worthless clusterings that have maximally bad quality and coverage.

because all its clusters are pure (none contain any irrelevant documents). Both clustering 4.23.A and 4.23.B should have higher quality and coverage than the worthless clusterings: although the worthless clusterings let users find documents from the large topic easily, so too did the original result set, and therefore the worthless clusterings offer no improvement over the original result set.

Average and weighted Precision, Total Recall, and Entropy perform particularly badly with all three types of worthless clustering and in the case of the singleton clustering are notably bad — they consider the worthless singleton clustering to be perfect!

Mutual Information does better and has no problems identifying the giant and random clusterings as worthless. However, although not as se-

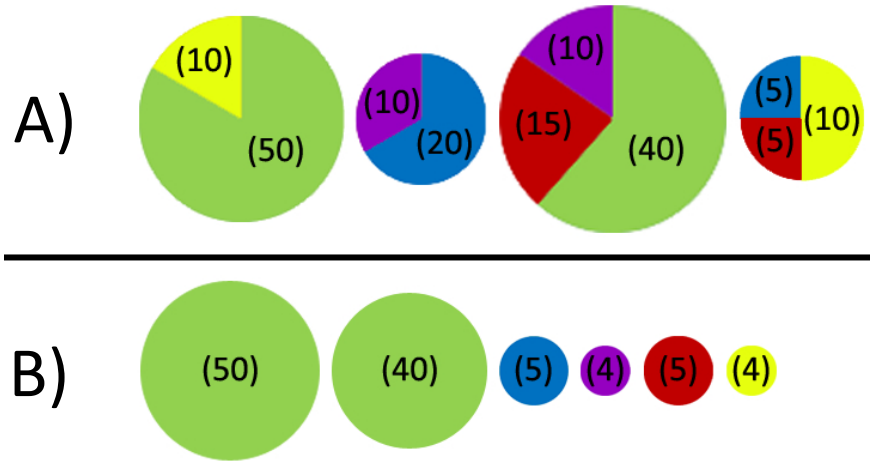

Figure 4.23: Two reasonable clusterings — A has higher coverage and B has higher quality.

vere, it too considers the singleton clustering to be better than other clusterings that are far from worthless. In particular, Mutual Information considers the singleton clustering to be better than the reasonable clustering 4.23.B and better than all three of the reasonable clusterings shown in figure 4.20 and evaluated in table 4.2.

Despite being a combination of Precision and Recall, F-measure surprisingly does the complete opposite of those measurements on the singleton clustering and correctly identifies it as worthless. Best Recall also identifies the singleton clustering as worthless. However, F-measure and Best Recall perform badly on giant and random clusterings with Best Recall considering the worthless giant clustering to be perfect.

Precision, Total Recall, and Entropy suffer on the singleton clustering because they are biased towards small clusters — the Singleton Clustering represents the extreme case of segmentation and these measurements fail to account for segmentation (as section 4.4.6 will discuss).

Precision, Recall, F-measure, and Entropy suffer on the giant and random clusterings because these measures are dependent on the ideal clustering. Specifically, they are related to the proportion of pages in the largest

Table 4.3: Synthetic test case results for Singleton Clustering, Giant Clustering, and Random Clustering

Quality	AP	WP	$AE^{\diamond}$	$WE^{\diamond}$	
Singleton	1	1	0	0	
Giant	0.909	0.909	0.268	0.268	
Random	0.895	0.895	0.3	0.3	
4.23.A	0.654	0.686	0.474	0.451	
4.23.B	1	1	0	0	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
Singleton	0.032	0.005	1	1	
Giant	1	1	0.2	0.909	
Random	0.194	0.173	0.17	0.773	
4.23.A	0.45	0.095	0.258	0.109	
4.23.B	0.154	0.062	0.162	0.098	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
Singleton	0.009	0.009	0.062	0.009	0.1
Giant	0.952	0.952	0.226	0.87	0
Random	0.286	0.286	0.094	0.264	0.001
4.23.A	0.435	0.38	0.393	0.128	0.123
4.23.B	0.232	0.124	0.263	0.114	0.041

topic, which can lead to very high performance for worthless clusterings that offer no improvement over the original result set. While this example exemplifies the problem as it has one topic that is substantially larger than the others, the problems with giant and random clusterings exist on balanced clusterings too, although they are less severe.

4.4.5 Cluster Composition

The composition of a cluster affects both quality and coverage, as discussed in section 4.2.4. In particular, quality should be lower when the irrelevant documents in a cluster are dispersed over more topics, whereas coverage should be lower when there are many irrelevant documents in a cluster. Note that adding an irrelevant document might (and often should) increase the coverage of that document's respective topic, but it will decrease the coverage of the cluster's topic (the topic of the relevant documents in the cluster).

Figure 4.24: An ideal clustering with five topics of equal size.

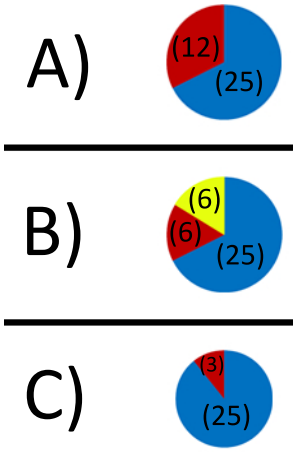

Figure 4.25: Three clusterings of the documents in figure 4.24. A contains a cluster where 32% of the documents are irrelevant. B contains a cluster with the same fraction of irrelevant documents, but dispersed over more topics. C contains a cluster where only 11% are irrelevant.

Table 4.4: Synthetic test case results for the three clusterings in figure 4.25

Quality	AP	WP	$AE^{\diamond}$	$WE^{\diamond}$	
4.25.A	0.676	0.676	0.391	0.391	
4.25.B	0.676	0.676	0.531	0.531	
4.25.C	0.893	0.893	0.212	0.212	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.25.A	0.296	0.296	0.2	0.2	
4.25.B	0.296	0.296	0.2	0.2	
4.25.C	0.224	0.224	0.2	0.2	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.25.A	0.806	0.806	0.239	0.239	0.36
4.25.B	0.806	0.806	0.239	0.239	0.278
4.25.C	0.943	0.943	0.211	0.211	0.353

Table 4.4 shows the evaluation results of the three clusterings shown in figure 4.25 compared against the ideal clustering shown in figure 4.24, which has five topics of equal size. Clustering 4.25.B has lower quality than 4.25.A, even though both have the same number of irrelevant documents, because the cluster in 4.25.B disperses the irrelevant documents over more topics. Clustering 4.25.C has higher coverage than 4.25.A, because there are fewer irrelevant documents in the cluster.

As quality measures, Precision and F-measure fail to consider cluster composition and do not penalize clustering 4.25.B, while Entropy and Mutual Information correctly penalize clustering 4.25.B.

As coverage measures, Recall, Topic F-measure, and Mutual Information fail to consider the irrelevant documents and do not increase coverage in clustering 4.25.C. Cluster F-measure correctly increases for clustering 4.25.C (although this is merely the result of Cluster F-measure being a combined measure and correctly capturing the increase in quality).

Surprisingly, Best Recall, Topic F-measure, and Mutual Information ac-

tually decrease for clustering 4.25.C, even though both quality and coverage have improved relative to 4.25.A. This occurs because these measurements have over emphasized the mere inclusion of documents and while sound from an information theoretic viewpoint for Mutual Information, it is undesirable when measuring quality and coverage.

4.4.6 Segmented Clusters

Quality and coverage are reduced when a topic is segmented (or split) into multiple smaller clusters, as discussed in section 4.2.5.

Table 4.5 shows the evaluation results of the two clusterings shown in figure 4.26. Clustering 4.26.B has lower quality than 4.26.A because its clusters reproduce their topic less accurately. F-measure and Mutual Information correctly give lower scores to 4.26.B, while Precision and Entropy incorrectly give the same score to both clusterings.

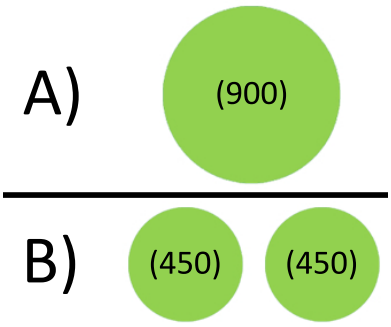

Figure 4.26: Two clusterings of the documents in figure 4.18. A contains a cluster with 900 documents and B contains the same documents split between two clusters.

Table 4.6 shows a set of six clusterings based on the ideal clustering shown on the left hand side of figure 4.8. The clusterings have varying degrees of segmentation for either full coverage or 33% coverage. Table 4.7 shows the evaluation results of the six clusterings. The ideal clustering (4.8 LHS) should have a perfect score, while the clustering with a single

Table 4.5: Synthetic test case results for the two clusterings in figure 4.26

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.26.A	1	1	0	0	
4.26.B	1	1	0	0	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.26.A	0.947	0.947	0.189	0.861	0.097
4.26.B	0.621	0.621	0.124	0.564	0.068

perfect cluster (4.8 LHS Blue) should have a lower score. 4.8 RHS should have the 2nd highest score because it covers more documents than 4.8 LHS Blue and it covers the blue topic using relatively few clusters. The three clusterings that split a single topic's documents into multiple clusters (4.8 RHS Blue, 4.8 RHS Green, and 4.8 RHS Purple) should have progressively lower scores.

Total Recall completely ignores segmentation: it gives a perfect score to both the ideal clustering (4.8 LHS) and the segmented clustering (4.8 RHS), and gives identical scores to 4.8 LHS Blue and the three progressively more segmented clusterings (4.8 RHS Blue, 4.8 RHS Green, and 4.8 RHS Purple).

Best Recall and Topic F-measure also ignore segmentation, although this is less obvious because they assign lower scores to the segmented clusterings. The lower scores occur because the best clusters in these clusterings are worse, not because the clusters are segmented. This is evident by examining the results for the two most segmented clusterings (4.8 RHS Green and 4.8 RHS Purple), Best Recall and Topic F-measure give equal scores to these clusterings because the best clusters in both are the same.

Cluster F-measure and Mutual Information correctly give progressively lower scores to 4.8 RHS Blue, 4.8 RHS Green, and 4.8 RHS Purple. It might seem strange that Cluster F-measure and Mutual Information give 4.8 RHS a lower score than 4.8 LHS Blue, since 4.8 RHS has higher coverage. However, 4.8 RHS has much lower quality than 4.8 LHS Blue due

Table 4.6: Six clusterings based on figure 4.8 with an ideal clustering consisting of 3 topics, each with 1000 documents. The numbers indicate the number of clusters of a given number of documents, for instance, 41x 20 means 41 clusters with 20 documents. The columns indicate the respective topic of the clusters. The cluster names relate the clusterings to figure 4.8, where LHS and RHS correspond to the clusters on the left and right side respectively and the colours reference the subset of clusters included in a clustering. For example, RHS Green is a clustering containing just the green clusters from the right hand side, while LHS is a clustering containing all the clusters on the left hand side.

	Blue	Green	Purple
4.8 LHS	1x 1000	1x 1000	1x 1000
4.8 RHS	1x 700 1x 200 4x 25	41x 20 45x 4	9x 20 205x 4
4.8 LHS Blue	1x 1000		
4.8 RHS Blue	1x 700 1x 200 4x 25		
4.8 RHS Green		41x 20 45x 4	
4.8 RHS Purple			9x 20 205x 4

to segmentation, and Cluster F-measure and Mutual Information measure both coverage and quality.

Cluster F-measure incorrectly gives the imperfect clustering 4.8 LHS Blue a perfect score, this is another example of Cluster F-measure's failure with perfect clusterings that was explained earlier in section 4.4.3.

Table 4.7: Synthetic test case results for the six clusterings in table 4.6

Coverage	AR^1	WR^1	AR^Σ	WR^Σ	
4.8 LHS	1	1	1	1	
4.8 RHS	0.247	0.247	1	1	
4.8 LHS Blue	0.333	0.333	0.333	0.333	
4.8 RHS Blue	0.233	0.233	0.333	0.333	
4.8 RHS Green	0.007	0.007	0.333	0.333	
4.8 RHS Purple	0.007	0.007	0.333	0.333	
Combined	AF^C	WF^C	AF^T	WF^T	MI
4.8 LHS	1	1	1	1	1
4.8 RHS	0.017	0.232	0.301	0.301	0.322
4.8 LHS Blue	1	1	0.333	0.333	0.667
4.8 RHS Blue	0.225	0.648	0.275	0.275	0.253
4.8 RHS Green	0.023	0.034	0.013	0.013	0.132
4.8 RHS Purple	0.009	0.014	0.013	0.013	0.113

4.4.7 Overlapping Clusters

The measurements should continue to measure quality and coverage correctly when the clusters or topics overlap, as discussed in section 4.2.6.

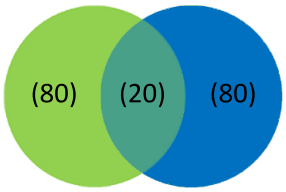

Figure 4.27: An ideal clustering with two overlapping topics, each topic contains 100 documents with 20 that overlap.

Figure 4.27 shows an ideal clustering with two overlapping clusters and figure 4.28 shows three possible clusterings of the same data. Table 4.8 shows the evaluation results of the three clusterings. Clustering 4.28.B

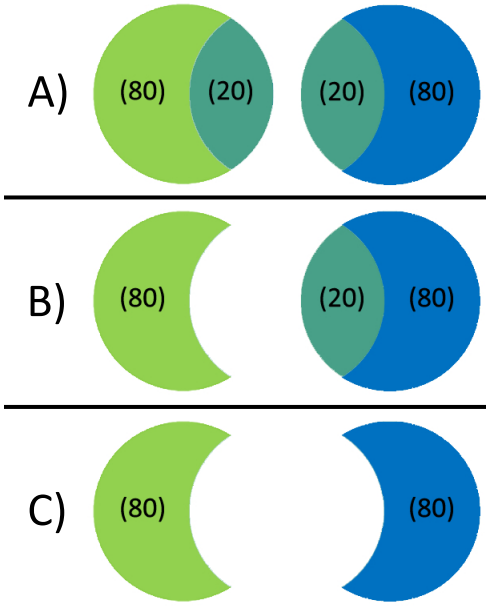

Figure 4.28: Three clusterings of the documents in figure 4.27. A is the perfect clustering and exactly matches the ideal clustering, B has less coverage with no clusters overlapping, and C has even less coverage with no clusters containing any documents belonging to multiple topics.

has lower coverage than 4.28.A because the green cluster does not cover the documents that overlap. Clustering 4.28.C has even lower coverage because neither cluster covers the documents that overlap. None of the clusterings contain any irrelevant documents and so all three have high quality, although due to segmentation, clusterings 4.28.B and 4.28.C have slightly lower quality.

Recall and F-measure correctly reflect the reduced coverage of clusterings 4.28.B and 4.28.C. Precision does not reflect the reduced quality since Precision does not identify segmented clusters.

Entropy and Mutual Information fail to handle overlapping topics and actually give inverted scores claiming that 4.28.B is better than 4.28.A and that 4.28.C is better again. This occurs because both Entropy and Mutual Information overcount the documents from the overlapping topics and

Table 4.8: Synthetic test case results for the three clusterings in figure 4.28

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.28.A	1	1	0.464	0.464	
4.28.B	1	1	0.232	0.258	
4.28.C	1	1	0	0	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.28.A	1	1	1	1	
4.28.B	0.9	0.9	0.9	0.9	
4.28.C	0.8	0.8	0.8	0.8	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.28.A	1	1	1	1	0.615
4.28.B	0.944	0.951	0.944	0.944	0.684
4.28.C	0.889	0.889	0.889	0.889	0.754

treat them as both relevant and irrelevant in the same cluster. As identified in section 4.4.3, Mutual Information can also fail when the clusters overlap, but the topics do not.

4.4.8 Hierarchical Clusterings

The measurements should also continue to measure quality and coverage correctly when the topics form a hierarchy, as discussed in section 4.2.7.

Table 4.9 shows the evaluation results of six clusterings of the documents from the hierarchical ideal clustering shown in figure 4.29. Clustering 4.30.A has poor coverage because many topics are missing and it has poor quality because it contains an impure cluster. 4.30.B has perfect quality, but has imperfect coverage because it is missing the purple and aqua topics and it has not covered those documents using clusters at another level of the hierarchy. Clusterings 4.31.A, 4.31.B, 4.31.C, and 4.31.D are all perfect because every cluster exactly matches a cluster from the ideal clus-

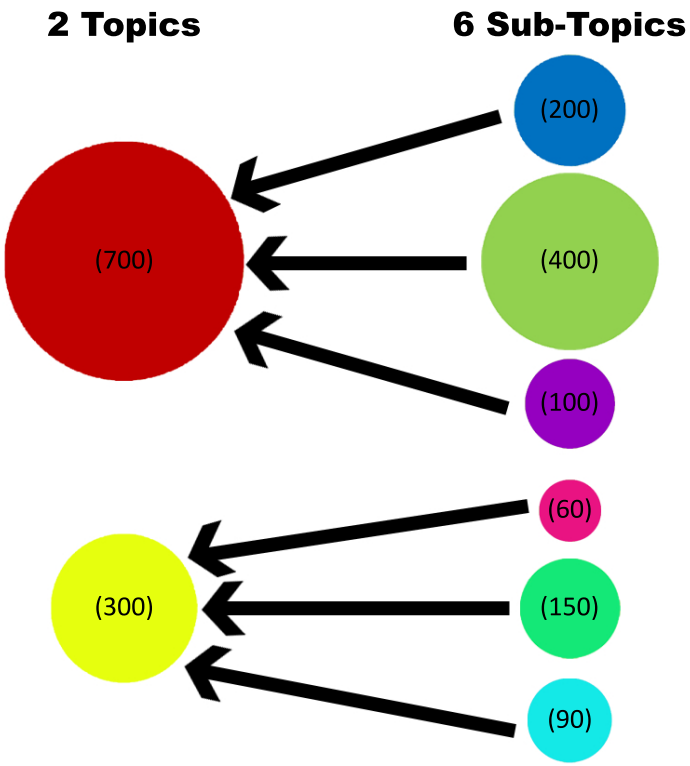

Figure 4.29: An ideal clustering with two top-level topics, each with three sub-topics.

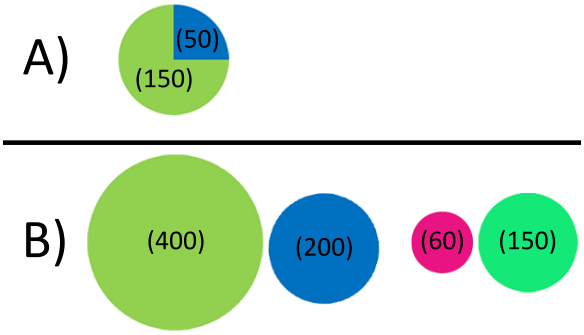

Figure 4.30: Two clusterings of the documents in figure 4.29. A has poor coverage and consists of an impure cluster that mixes 2nd-level topics. B has greater coverage than A, but still not perfect and all the clusters in B have perfect quality.

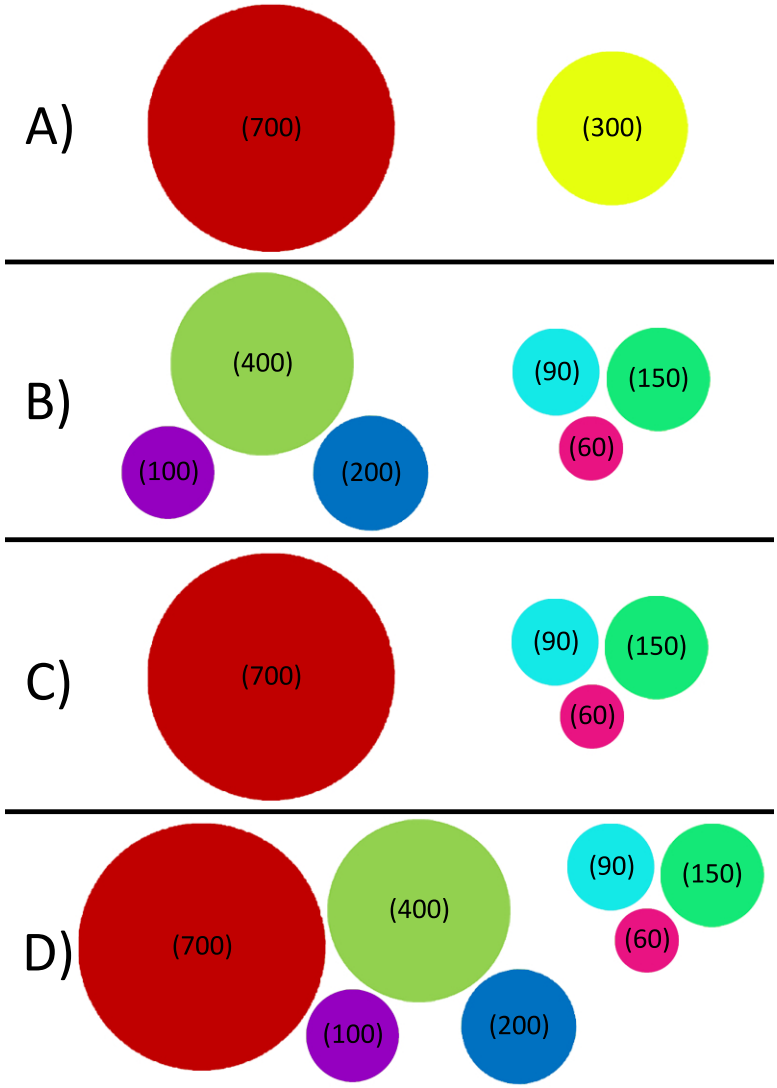

Figure 4.31: Four different perfect clusterings of the documents in figure 4.29. A exactly matches the top-level of the ideal clustering, B exactly matches the 2nd-level, C matches some top-level topics and some 2nd-level topics, and D matches all 2nd-level topics and additionally matches one of the top-level topics.

Table 4.9: Synthetic test case results for the six clusterings shown in figures 4.31 and 4.30

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.30.A	1	1	0.27	0.27	
4.30.B	1	1	0	0	
4.31.A	1	1	0.477	0.47	
4.31.B	1	1	0	0	
4.31.C	1	1	0.115	0.322	
4.31.D	1	1	0.066	0.189	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.30.A	0.114	0.2	0.083	0.175	
4.30.B	0.634	0.68	0.695	0.81	
4.31.A	1	1	0.5	0.775	
4.31.B	0.884	0.775	1	1	
4.31.C	0.938	0.925	0.75	0.85	
4.31.D	0.938	0.925	1	1	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.30.A	0.5	0.5	0.149	0.281	0.17
4.30.B	1	1	0.674	0.76	0.928
4.31.A	1	1	0.61	0.783	0.881
4.31.B	1	1	0.924	0.855	1.136
4.31.C	1	1	0.761	0.802	0.883
4.31.D	1	1	0.958	0.95	1.341

tering and either every top-level topic is covered or all the corresponding sub-topics are covered.

Note that if the cluster in 4.30.A were compared against the red cluster it would be considered pure. However, since the cluster is most similar to the 2nd-level topics, it should be compared to the topics at that level (as discussed in section 4.2.7) and consequently it has poor quality.

Entropy, Recall, Topic F-measure, and Mutual Information all fail to give perfect scores to all the perfect clusterings. Furthermore, in some instances they give better scores to imperfect clusterings than to perfect clusterings, as happens with the imperfect clustering 4.30.B relative to perfect clustering 4.31.A. Best Recall works correctly with top-level clusters (4.31.A) and Total Recall works correctly with low-level clusters (4.31.B and 4.31.D).

Precision incorrectly measures the quality of hierarchical clusterings because it gives perfect quality to the imperfect clustering 4.30.A. Cluster F-measure incorrectly gives perfect coverage to 4.30.C, but this error is unrelated to the hierarchical nature of the clustering because Cluster F-measure incorrectly measures coverage on similar non-hierarchical clusterings too (section 4.4.1).

4.4.9 Limited User Time

In domains where the users of clusters have limited time, there is more relative value in increasing the size of small clusters than in increasing the size of larger clusters, as discussed in section 4.2.9.

Table 4.10 shows the evaluation results of the four clusterings shown in figure 4.32. The improvement of 4.32.B over 4.32.A should be relatively greater than the improvement of 4.32.D over 4.32.C because a user with limited time is less likely to examine the extra documents in 4.32.D than those in 4.32.B.

Precision and Entropy cannot account for user time because they do not have the segmentation property and therefore ignore the size of clusters. Recall and Mutual Information could account for user time, but do not and the relative improvement of 4.32.D over 4.32.C (double from double the documents) is the same as the relative improvement of 4.32.B over 4.32.A.

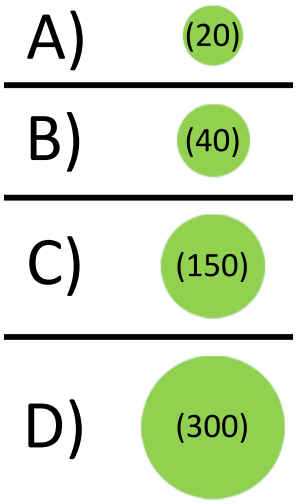

Figure 4.32: Four clusterings of the documents in figure 4.18. The clusterings progressively cover more documents, but all from the same topic.

F-measure correctly accounts for user time because the harmonic mean[12] is sensitive to smaller values. For example, with AF^C, the relative improvement of 4.32.D over 4.32.C is 1.77, while the relative improvement of 4.32.B over 4.32.A is 1.97 — there is relatively more value in doubling the size of the small cluster than doubling the size of the large cluster.

[12]F-measure is the harmonic mean of precision and recall.

Table 4.10: Synthetic test case results for the four clusterings in figure 4.32

Quality	AP	WP	$AE^{\Diamond}$	$WE^{\Diamond}$	
4.32.A	1	1	0	0	
4.32.B	1	1	0	0	
4.32.C	1	1	0	0	
4.32.D	1	1	0	0	
Coverage	AR^{1}	WR^{1}	AR^{Σ}	WR^{Σ}	
4.32.A	0.004	0.018	0.004	0.018	
4.32.B	0.008	0.036	0.008	0.036	
4.32.C	0.03	0.136	0.03	0.136	
4.32.D	0.06	0.273	0.06	0.273	
Combined	AF^{C}	WF^{C}	AF^{T}	WF^{T}	MI
4.32.A	0.039	0.039	0.008	0.036	0.002
4.32.B	0.077	0.077	0.015	0.07	0.004
4.32.C	0.261	0.261	0.052	0.237	0.016
4.32.D	0.462	0.462	0.092	0.42	0.032

4.4.10 Summary

None of the existing measures meets all the requirements of a good measure of quality or coverage. In fact, as shown in table 4.11, all but one measure fails more than half the requirements; Cluster F-measure is the exception, but it still fails 6 of the 16 requirements.

Table 4.11: A summary of what properties each evaluation measurement satisfies. Green indicates the measurement has the property, red indicates it does not, and white indicates the property is not applicable to the measurement.

	P	E	R^1	R^Σ	F^C	F^T	MI
Separate Measures							
Basic Quality							
Basic Coverage							
Size Bias							
Perfect Clusterings							
Worthless - Singleton							
Worthless - Giant							
Worthless - Random							
Composition - Quality							
Composition - Coverage							
Segmentation - Quality							
Segmentation - Coverage							
Overlapping Clusters							
Overlapping Topics							
Hierarchical Topics							
Limited User Time							

4.5 Approaches used in practice

In practice, researchers have used a wide assortment of evaluation methods and as noted by Ferragina et al. [64], there is little consensus on what should be used. With no evaluation method suitable for comparing all types of web page clustering, researchers must often choose between somewhat dubious methods that are suited to their algorithm's special characteristics and other methods that are unsuitable for their algorithm. More commonly, they must include both and explain the discrepancy between the results.

4.5.1 Suffix Tree Clustering

Zamir [203] faced this tradeoff in evaluating Suffix Tree Clustering (STC) [201] and Grouper [202]. Grouper introduced a distinct user interface that required user studies to evaluate, while STC generates overlapping clusters, which makes the common-place merge-then-cluster approach unsuitable because it creates ideal clusterings that are partitions. Instead, Zamir chose to fall back on a task-oriented method called Search Result Reordering, where the most relevant clusters for a particular search goal are used to reorder the search results. The reordered search results are then evaluated using standard search engine evaluation methods such as those discussed in chapter 3.

The Search Result Reordering method is not without its limitations, as Zamir notes in conclusion "The IR approaches impose an unrealistic user model and then apply (controversial) metrics developed for a different task.". More specifically, Zamir notes that the one-cluster model of Search Result Reordering is biased against small clusters and so they use the multi-cluster model. However, akin to the discussion of average and weighted measurements in section 4.1.4, the multi-cluster model is arguably biased towards small clusters. As it happens, STC tends to create small clusters and so under the one-cluster model it performs poorly,

while under the multi-cluster model it performs well. To balance the limitations of Search Result Reordering, Zamir included a less contentious merge-then-cluster evaluation that used precision and pairwise accuracy. However, this used synthetic ideal clusterings that were partitions and so STC, which creates overlapping clusters, was at a natural disadvantage, and perhaps unsurprisingly, was outperformed by k-means.

4.5.2 Lingo

Osinski et al. evaluated Lingo in two ways: in [137] they conducted a very small user study, and in [135] they conducted a qualitative evaluation using a merge-then-cluster approach. They chose to perform a qualitative evaluation because they were discouraged by the limitations of their earlier quantitative evaluation in [164]. Specifically, they felt limited by having to use a constrained ideal clustering because they felt these unfairly penalize equally justifiable clustering decisions. For example, as depicted in figure 4.33, a variety of structurally different clusterings may be equally good, but all are penalized for differing from the ideal clustering, which has been artificially constrained to a coarse single-level disjoint clustering.

In [164] they used Byron Dom's entropy measure, an information theoretic measure that is similar to Mutual Information, to evaluate STC. However, since the measurement could not cope with overlapping clusters, they had to restrict STC to constructing a partition, a serious limitation that undoubtedly affected STC's performance very significantly as shown by Zamir earlier in [203].

4.5.3 Merge-Then-Cluster

Merge-then-cluster is an approach that enables the rapid construction of ideal clusterings. Osinski et al. [135] used the merge-then-cluster approach by finding sets of pre-categorized documents using the Open Directory Project (ODP) and then merging them together to form sets of doc-

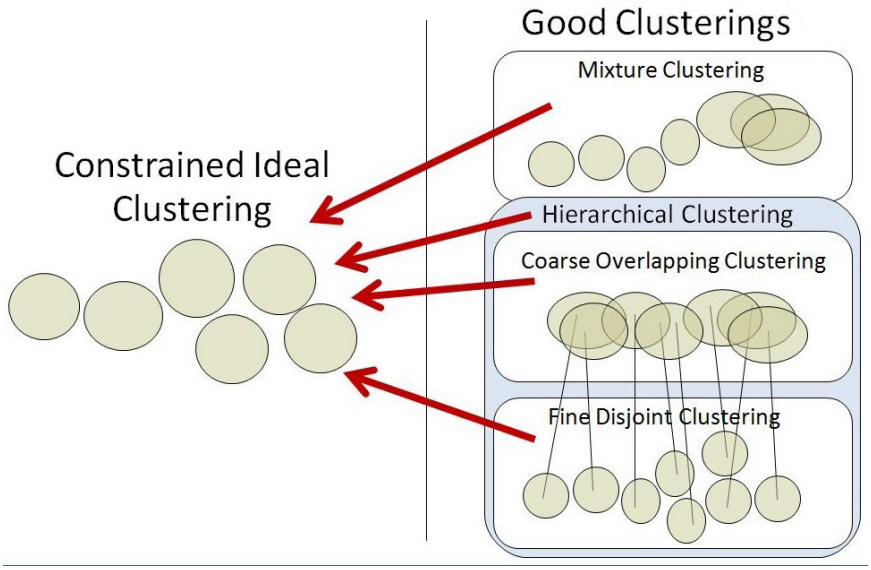

Figure 4.33: Comparing clusterings against a constrained ideal clustering leads to equally justifiable clusterings being penalized.

uments to cluster (another source for pre-categorized documents is newsgroups as used by Strehl [167]). The categorizations from the ODP form a partition of the documents and represent an ideal clustering. Different ideal clusterings can be found by merging different sets of clusters. To evaluate Lingo, Osinski et al. selected some diverse categories such as movies, health, and computer science, as well as some related categories such as MySQL and PostgreSQL and then merged these together in various combinations.

In using merge-then-cluster, there is a tradeoff between the realism of the document set and the accuracy of the ideal clustering. When several very distinct categories are combined, such as movies and health, the resulting document set is not very natural because it doesn't reflect the typical result set encountered when performing web search. However, the corresponding ideal clustering is very accurate as the probability of there being overlapping documents between such diverse topics is low. An unnatural result set is particularly bad for evaluation, as it is significantly

easier for clustering algorithms to separate distinct documents than to separate similar documents.

Conversely, when very similar categories are combined, such as MySQL and PostgreSQL, the resulting document set is more natural (in fact the first result from Google for the query "open source databases" is MySQL and the second is PostgreSQL). However, the corresponding ideal clustering is generally inaccurate because there is likely to be a lot of overlap between these topics, overlap that is not reflected in the ideal clustering. These inaccuracies with the ideal clustering cause bias that can unjustly penalize some algorithms.

While merge-then-cluster has problems, it is widely used [203, 135, 76, 167, 33] because it provides an extremely efficient method for constructing result sets and their corresponding "ideal" clusterings. Constructing ideal clusterings manually from a set of search results is much better because it enables the identification of overlapping topics, the construction of hierarchies, and the inclusion of clusters at different levels of granularity, but as discovered when constructing the ideal clusterings used in this thesis, it is very laborious and very time consuming. One important avenue for future research is to find an efficient method for constructing natural result sets with accurate ideal clusterings.

4.5.4 Modifying Measurements

Researchers have used a wide variety of measurements for gold standard evaluations. Strehl [167] used Purity, Entropy, Precision, Recall, F-measure, and Mutual Information, Halkidi et al. [76] used F-measure and the Rand Index, and as noted already, Zamir [203] used Precision and pairwise accuracy. However, many researchers have had to make minor modifications to the measurements to obtain non-spurious results.

Hou et al. [87] used Purity, but compared against the top-level topics and the lower-level topics separately to account for the hierarchical nature

of their algorithm. Wang et al. [188] used Precision, Recall, and Entropy, but removed clusters with fewer than 3 documents to avoid problems with worthless singleton clusters. Schenker et al. [155] used Mutual Information and the Rand Index, but altered the ideal clustering to remove overlap. Cheng et al. [33] used F-measure, Entropy, and Purity, but modified them to take account of the hierarchical nature of their non-overlapping ideal clustering.

4.5.5 Summary

Researchers use a hodgepodge of methods to evaluate their algorithms and often face tradeoffs between bias towards one type of clustering or another. They often work around these limitations as best they can by making small modifications to the measurements, by restricting the result set, by restricting the ideal clustering, by restricting the clusterings, or by limiting some algorithms. Although these "solutions" give some indication of the effectiveness of a clustering algorithm, they are certainly not ideal.

The rest of this chapter details one possible implementation of quality and coverage measurements that enable the unbiased comparison of algorithms that produce structurally different clusterings against a gold standard, while having all the properties of good measures discussed so far in this chapter.

4.6 New Ideal Clustering Representation

4.6.1 A Structurally Richer Ideal Clustering

QC4 is a measure that uses a richer ideal clustering that is structurally flexible, thereby meeting the requirements of a good evaluation method as discussed in section 4.1.2. This structural flexibility enables clusterings

to be compared without bias towards any specific characteristics present in the ideal clustering, as shown in figure 4.34.

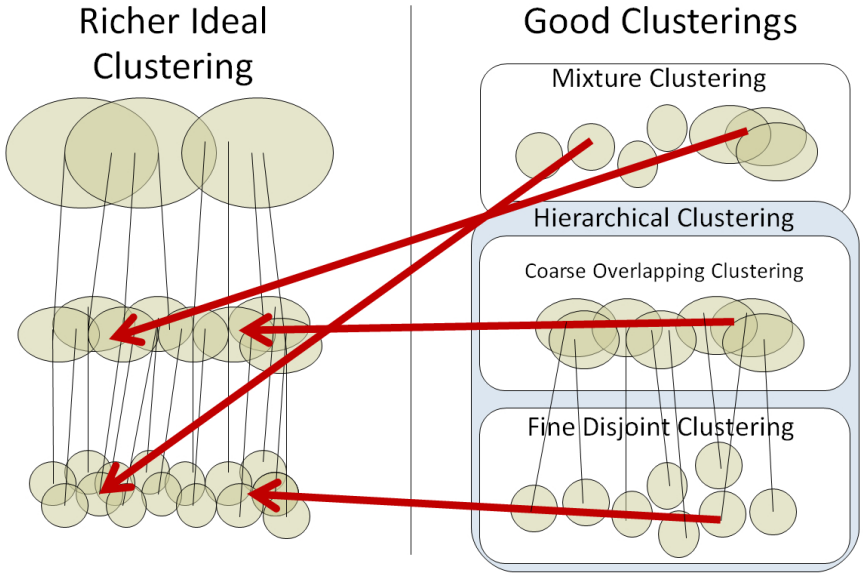

Figure 4.34: Clusterings of any structure can be compared against a rich ideal clustering without bias towards certain characteristics.

QC4's ideal clustering structure is a hierarchy of topics, organized in levels, so that the set of topics at the top-level represents a coarse categorization of the pages, and the sets of topics at lower levels represent progressively finer categorizations. This allows QC4 to fairly compare algorithms that produce clusterings of different granularity and to compare algorithms that generate hierarchical clusterings.

Topics may overlap other topics (at the same and different levels), since real pages may belong to multiple topics. However, all pages must be contained in at least one topic at each level. This makes it possible to specify the full range of semantically meaningful topics found in the data, without being constrained to just disjoint topics.

Since search engines often return outliers — pages that are unrelated to all the other pages — the ideal clustering may contain a single outlier

topic (present at every level) that contains all the outliers. The outlier topic must be disjoint from every other topic. QC4 ignores the outlier topic when computing quality and coverage and ignores clusters that contain a majority of outliers when computing quality.

4.6.2 Measuring Overall Performance using Quality and Coverage

Gold standard methods compare clustering algorithms by comparing their output clusterings to the ideal clustering. As discussed in section 4.1.3, this comparison should occur on two dimensions: quality and coverage.

In addition to the notation in section 4.3, the rest of this chapter uses the following notation:

L is the set of levels from the topic hierarchy (e.g. 1, 2, 3)
l is an individual level from L
T^l is the set of topics at level l
T_d is the set of topics containing document d
$sub(t)$ is the set of topics containing t and all descendants of topic t
$lvl(t)$ is the level of topic t

QC4 defines four overall measurements of performance, two based on Cluster Quality $QU(c)$ and two based on Topic Coverage $CV(t)$, which will be defined in sections 4.7 and 4.8 respectively. The overall measurements of Clustering Quality, AQ and WQ are the average of the cluster qualities, but in WQ they are weighted by cluster size.

$$AQ = \text{average quality} = \frac{\sum_{c \in C} QU(c)}{|C|}$$

$$WQ = \text{weighted quality} = \frac{\sum_{c \in C} QU(c)|D_c|}{\sum_{c \in C} |D_c|}$$

Similarly, the overall measurements of Clustering Coverage, AC and WC are the average of the topic coverages, but in WC they are weighted by topic size. However, as the Topic Coverage measure incorporates the

coverage of each topic's children, the Topic Coverage is averaged over just the top-level topics from the ideal clustering.

$$AC = \text{average coverage} = \frac{\sum_{t \in T^1} CV(t)}{|T^1|}$$

$$WC = \text{weighted coverage} = \frac{\sum_{t \in T^1} CV(t)|D_t|}{\sum_{t \in T^1} |D_t|}$$

The average measures give a fairer measure of the smaller, fine grained clusters and topics; the weighted measures give a fairer measure of the larger, broad clusters and topics. This avoids bias from aggregate results as discussed in section 4.1.4.

4.7 New Quality Measure

As section 4.4 explained, none of the existing measurements has all the properties required of a good quality measure as set out in section 4.2. Therefore, QC4 introduces a new measure of cluster quality — an improved version of Entropy that

1. is computed at the level of granularity that contains the single best matching topic to account for hierarchical topics;
2. uses a modified Precision measure to account for overlapping clusters;
3. is discounted to account for random clusterings, singleton clusterings, giant clusters, segmented clusters, and limited user time.

4.7.1 An Improved Version of Entropy

QC4's measure of cluster quality, $QU(c)$, is based on a modified Entropy measure, $E(c)$. Standard Entropy is unsuitable because it fails on worthless clusterings, segmented clusters, overlapping topics, hierarchical top-

ics, and does not account for limited user time. However, as an information theoretic measure it does consider cluster composition, unlike Precision and F-measure, which makes Entropy a good starting point for a good measure of cluster quality.

The standard Entropy measure does not work with overlapping topics since pages in multiple topics are overcounted. There are two kinds of overlap: overlap of topics at different levels, and overlap of topics at the same level.

4.7.1.1 Overlap between Levels

Overlap between levels can be addressed by computing the Entropy over the topics in a single level as shown in figure 4.35. QC4 chooses the level[13], $L(c)$, containing the topic that is the most similar to the cluster as measured by the standard F-measure, $F(c,t)$, defined in section 4.3. Then it computes the Entropy using a modified Precision measure, $P'(c,t,t_{best})$, which addresses overlap within levels as section 4.7.1.2 will explain.

$$L(c) = \text{cluster-level} = lvl\left(\underset{t \in T}{\operatorname{argmax}}\{F(c,t)\}\right)$$

$$E(c) = \min_{t_{best} \in T^{L(c)}} \left\{ -\sum_{t \in T^{L(c)}} P'(c,t,t_{best}) \log_{|T^{L(c)}|} P'(c,t,t_{best}) \right\}$$

By computing Entropy over a single level, QC4 compares clusters against topics of a similar level of granularity. This enables the evaluation of hierarchical clusterings and algorithms that produce clusterings at different levels of granularity.

4.7.1.2 Overlap within Levels

The overlap of topics within a level is addressed by the modified Precision measure, $P'(c,t,t_{best})$. To eliminate overcounting, the modified Precision

[13] If multiple topics maximize F, the one with lowest level is selected.

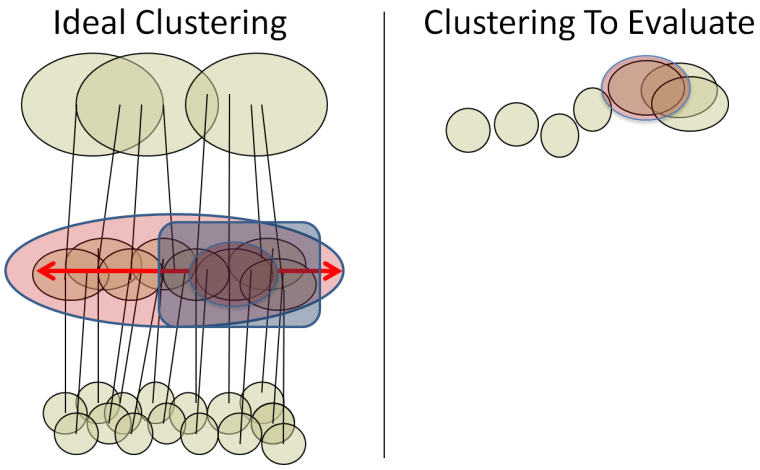

Figure 4.35: QC4 calculates the modified Entropy of a cluster across the topics at the best matching level of granularity.

measure counts pages in the topic that best matches the cluster (t_{best}) as if they were only in that topic as shown in figure 4.36, and then normalizes the Precision to remove the effect of any other overcounting as shown in figure 4.37.

$$P'(c,t,t_{best}) = \begin{cases} \dfrac{|D_{c,t}|}{|D_c|} & if\{t = t_{best}\} \\ \dfrac{(|D_c| - |D_{c,t_{best}}|)\ |D_{c,t} \setminus D_{c,t_{best}}|}{|D_c| \sum\limits_{t' \in T^{L(c)} \setminus \{t_{best}\}} |D_{c,t'} \setminus D_{c,t_{best}}|} & otherwise \end{cases}$$

With these modifications $E(c)$ compares clusters against the topics of an appropriate level of granularity and allows both disjoint and overlapping topics to be handled fairly, thus resolving the problems the standard Entropy measure had with overlapping topics and hierarchical topics.

4.7.1.3 Justification of modified Precision

The normalization process used in the modified Precision measure ensures that modified Precision behaves on overlapping topics in a similar way to

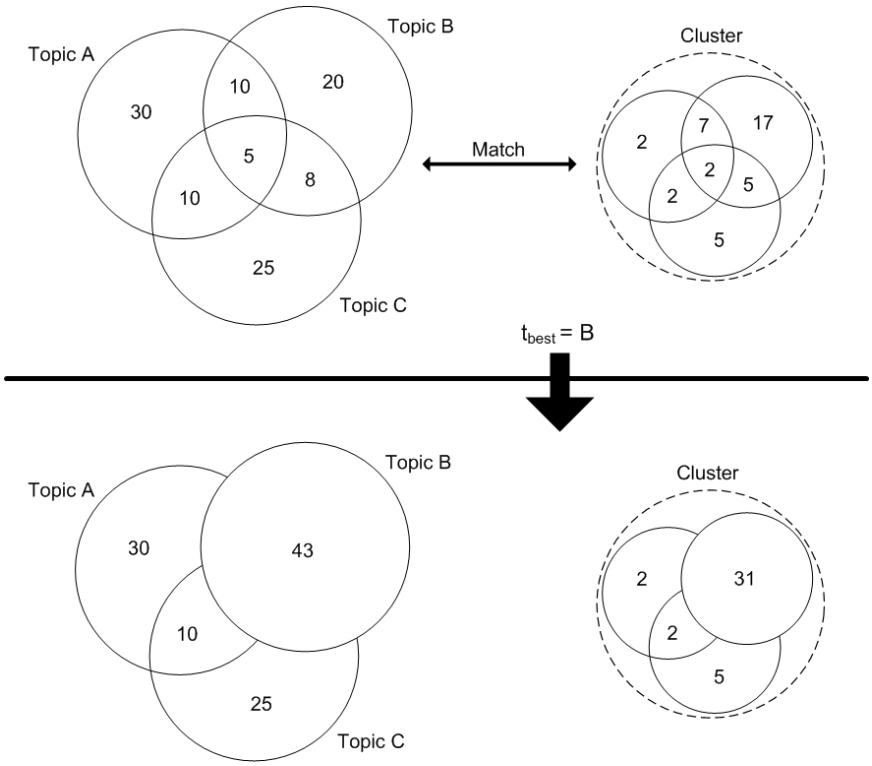

Figure 4.36: The modified Precision measure counts pages in the best matching topic as if they were only in that topic.

standard Precision on disjoint topics.

Standard Precision has two useful properties on disjoint topics:

1. $\sum_{t \in T} P(c,t) = 1$
2. the Precision of a pure[14] cluster is 1 for its corresponding topic and 0 for all other topics

This enables Precision to be viewed as measuring three factors:

1. the purity of a cluster with respect to its best matching topic
2. the total impurity ($1 -$ purity) of a cluster

[14] a subset of a single topic

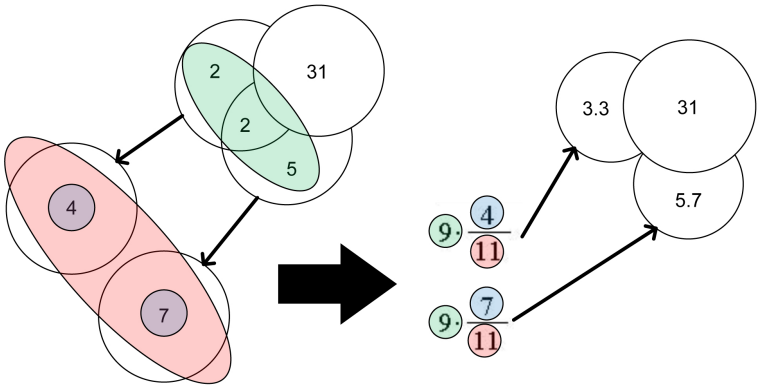

Figure 4.37: The modified Precision measure normalizes the counts to remove the effect of overcounting for any remaining overlap between topics.

3. how that impurity is distributed between the remaining topics

The modified Precision measure mimics standard Precision, while extending it to overlapping topics and has the analogous properties:

1. $\sum_{t \in T^{L(c)}} P'(c, t, t_{best}) = 1$ (as shown in proof 4.1)

2. the modified Precision of a pure cluster is 1 for its best matching topic and 0 for all other topics (as shown in proof 4.2)

Different components of the modified Precision measure capture the three factors measured by Precision. As highlighted in equation 4.1, the blue, green, and brown components capture factors 1, 2, and 3 respectively.

$$P'(c, t, t_{best}) = \begin{cases} \frac{|D_{c,t}|}{|D_c|} & if\{t = t_{best}\} \\ \frac{(|D_c| - |D_{c,t_{best}}|)\ |D_{c,t} \setminus D_{c,t_{best}}|}{|D_c| \sum_{t' \in T^{L(c)} \setminus \{t_{best}\}} |D_{c,t'} \setminus D_{c,t_{best}}|} & otherwise \end{cases} \tag{4.1}$$

$$
\begin{aligned}
& \sum_{t \in T^{L(c)}} P'(c, t, t_{best}) \\
= \quad & \frac{|D_{c,t_{best}}|}{|D_c|} + \sum_{t \in T^{L(c)} \setminus \{t_{best}\}} \left\{ \frac{(|D_c| - |D_{c,t_{best}}|) \ |D_{c,t} \setminus D_{c,t_{best}}|}{|D_c| \sum_{t' \in T^{L(c)} \setminus \{t_{best}\}} |D_{c,t'} \setminus D_{c,t_{best}}|} \right\} \\
= \quad & \frac{|D_{c,t_{best}}|}{|D_c|} + \frac{(|D_c| - |D_{c,t_{best}}|)}{|D_c|} \frac{\sum_{t \in T^{L(c)} \setminus \{t_{best}\}} |D_{c,t} \setminus D_{c,t_{best}}|}{\sum_{t' \in T^{L(c)} \setminus \{t_{best}\}} |D_{c,t'} \setminus D_{c,t_{best}}|} \\
= \quad & \frac{|D_{c,t_{best}}| + |D_c| - |D_{c,t_{best}}|}{|D_c|} \\
= \quad & 1
\end{aligned}
$$

Proof 4.1: The sum of purity and total impurity is 1 for the modified Precision measure

Assuming cluster c is pure with respect to its best matching topic, t_{best}, then

	$\lvert D_{c,t_{best}}\rvert = \lvert D_c\rvert$	(1) Since $c \subset t_{best}$
$\therefore$	$\frac{\lvert D_{c,t_{best}}\rvert}{\lvert D_c\rvert} = 1$	(2)
$\therefore$	$P'(c, t_{best}, t_{best}) = 1$	(3)
	$\lvert D_c\rvert - \lvert D_{c,t_{best}}\rvert = 0$	(4) From (1)
$\therefore$	$\forall t \in T^{L(c)}, t \neq t_{best}$ $\implies P'(c, t, t_{best}) = 0$	(5)

Proof 4.2: The modified Precision of a pure cluster is 1 for its best matching topic and 0 for all other topics

4.7.1.4 Fixing Entropy's Remaining Problems

Even with its improvements, the modified Entropy measure still fails on worthless clusterings, segmented clusters, and does not account for limited user time.

Cluster Quality, $QU(c)$, accounts for the remaining faults with the modified[15] Entropy measure, $E(c)$, by scaling it down using two new measures: $S_{random}(c)$ and $S_{segment}(c)$. $S_{random}(c)$ accounts for worthless random clusterings and worthless giant clusters, while $S_{segment}(c)$ accounts for segmented clusters, worthless singleton clusterings, and limited user time. Cluster Quality also inverts the modified Entropy measure so that 0 represents a worthless clustering and 1 represents a perfect clustering.

$$QU(c) = (1 - E(c)) \min\{1, S_{random}(c), S_{segment}(c)\}$$

4.7.2 Discounting Random Clusterings

Random clusters are similar in composition to that of the expected random cluster, as shown in section 4.2.8. QC4 uses this property to discount the performance of worthless random clusterings and worthless giant clusters.

QC4 only needs to consider random clusters, as a giant cluster (a cluster that contains every document) is merely a special kind of random cluster. Specifically, a giant cluster is a random cluster containing close to $|D|$ documents and in fact, a giant cluster with exactly $|D|$ documents has exactly the composition of the expected random cluster.

4.7.2.1 Partitions and Expected Random Clusters

The $S_{random}(c)$ measure should maximize the probability of penalizing random clusters and minimize the probability of penalizing good clusters. The measure's performance depends on the distance between the expected

[15]The standard Entropy measure also exhibits these faults.

random cluster and good clusters, which depends on the labelled partition (section 4.2.8) of the documents.

Regardless of the partition, random clusters will generally be similar in composition to the expected random cluster and penalized accordingly. However, with a labelled partition that has no semantic basis, even good clusters could be similar in composition to the expected random cluster.

Using the ideal clustering as the partition for constructing the expected random cluster ensures that good clusters will not be close to the expected random cluster, since clusters close to the expected random cluster provide no more information than the original result set and therefore would not be good clusters. Therefore, QC4 bases the partition it uses to construct the expected random cluster on the ideal clustering.

4.7.2.2 Transforming Overlapping Topics into Disjoint Regions

The topics from the ideal clustering provide a set of labels, but they do not necessarily form a partition because they might contain overlapping topics. However, it is possible to transform the topics into a partition by considering each unique intersection of topics as a region. $Reg(l)$ defines such a transformation of the topics at level l of the hierarchy.

$$Reg(l) = \{r \subseteq D | (\exists T_\alpha \subseteq T^l)(|r| > 0 \wedge r = \bigcap_{t' \in T_\alpha} D_{t'} \setminus \bigcup_{t'' \in T^l \setminus T_\alpha} D_{t''})\}$$

where r is a disjoint set of documents, T_α is a subset of the topics at level l, t' and t'' are topics, and where $D_{t'}$ and $D_{t''}$ are sets of documents.

Figure 4.38 shows the seven regions given by the region transformation function, $Reg(l)$, applied to three overlapping topics. Each disjoint region can be expressed as the difference between the intersection of a set of topics and the union of the remaining topics, as shown in table 4.12.

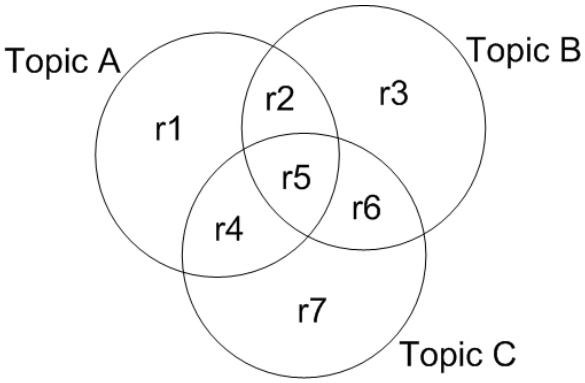

Figure 4.38: Three overlapping topics can be divided into at most seven disjoint regions

Table 4.12: Each disjoint region of figure 4.38 can be expressed as the difference between the intersection of T_α and the union of $T^l \setminus T_\alpha$.

r	$=$	$\bigcap_{r' \in T_\alpha} D_{r'}$	$\setminus$	$\bigcup_{r'' \in T^l \setminus T_\alpha} D_{r''}$
$r1$	$=$	A	$\setminus$	$(B \cup C)$
$r2$	$=$	$(A \cap B)$	$\setminus$	C
$r3$	$=$	B	$\setminus$	$(A \cup C)$
$r4$	$=$	$(A \cap C)$	$\setminus$	B
$r5$	$=$	$A \cap B \cap C$		
$r6$	$=$	$(B \cap C)$	$\setminus$	A
$r7$	$=$	C	$\setminus$	$(A \cup B)$

4.7.2.3 Penalizing Random Clusters

To avoid penalizing non-random clusters, QC4 should penalize only clusters close to the expected random cluster and the penalty should decrease with the distance from the expected random cluster.

The normalized region fraction, $NRF(c, r)$, is the fraction of documents from a region, r, in a cluster, c, normalized so that $\sum_{r \in Reg(L(c))} NRF(c, r) =$

1.

$$NRF(c,r) = \frac{\frac{|D_c \bigcap r|}{|r|}}{\sum_{r \in Reg(L(c))} \frac{|D_c \bigcap r|}{|r|}}$$

where $L(c)$ is the level of cluster c (section 4.7.1.1). In the expected random cluster, the fraction of documents from each region is the same. In non-random clusters, and particularly in good clusters, the fraction of documents from each region is very different. This allows the differentiation of random clusters and non-random clusters based on how uniformly a cluster disperses its documents over the regions.

QC4 measures the uniformity, $U(c)$, of a cluster using an Entropy based measure.

$$U(c) = -\sum_{r \in Reg(L(c))} NRF(c,r) \log_{|Reg(L(c))|} NRF(c,r)$$

A cluster that has documents from lots of different regions will have higher entropy than a cluster whose documents are all from the same region. $U(c)$ has a range of $[0,1]$, is maximized by the expected random cluster (which has uniform NRF values of $\frac{1}{|Reg(L(c))|}$ for every region), and is minimized by pure clusters. The smaller the value of $U(c)$, the greater the distance from the expected random cluster to the cluster, and the smaller the penalty applied to the cluster.

Finally, QC4 inverts the uniformity measure to discount uniform clusters, and applies a threshold of 5%, so that only the most uniform clusters (those that are most likely to be random) are discounted.

$$S_{random}(c) = \frac{1 - U(c)}{0.05}$$

4.7.2.4 Accuracy of the Random Cluster Penalty

To test the accuracy of the Random Cluster Penalty, $S_{random}(c)$, I ran a number of experiments on different ideal clusterings. Each experiment

enumerated every distinct cluster composition and calculated the random cluster penalty. The composition of a cluster depends on the number of documents from each topic, but not the documents themselves. For example, a cluster with 5 documents from topic one and 10 from topic two was considered distinct from one with 5 from topic one and 11 from topic two, but not distinct from a cluster with 5 different documents from topic one and 10 from topic two.

Figure 4.39 shows the results for three of these experiments.[16] The top row reflects a large cluster selected from a skewed ideal clustering, the middle row reflects a normal cluster selected from a balanced ideal clustering, and the bottom row reflects a small cluster selected from a balanced ideal clustering. The first column shows the frequency of randomly generated clusters as per the simulations and figures in section 4.2.8. The second and third columns show the penalty applied to the clusters by QC4 and an alternate method based on Mutual Information respectively. The amount of penalty corresponds to the colour band, with aqua, purple, green, red, light blue, and dark blue corresponding to 80–100%, 60–80%, 40–60%, 20–40%, 0–20%, and 0% respectively.

The left column shows where random clusters occur most frequently, and consequently, where it is most important to penalize random clusters. The middle column shows the penalty applied by QC4. In all three cases, QC4 applied the strongest penalty to the most frequently occurring random clusters.

In the top row, QC4 applied a weak penalty to many clusters in the bottom left corner of the plot (the region that corresponds to the black outline in the top-left plot). Although the relative frequency of random clusters in this region is low, the raw data shows that many random clusters do occur here, and consequently, QC4 is correct to penalize them. Furthermore, even if some real clusters did occur in this region, they would be useless

[16]The three experiments correspond to the random cluster simulations discussed in section 4.2.8.

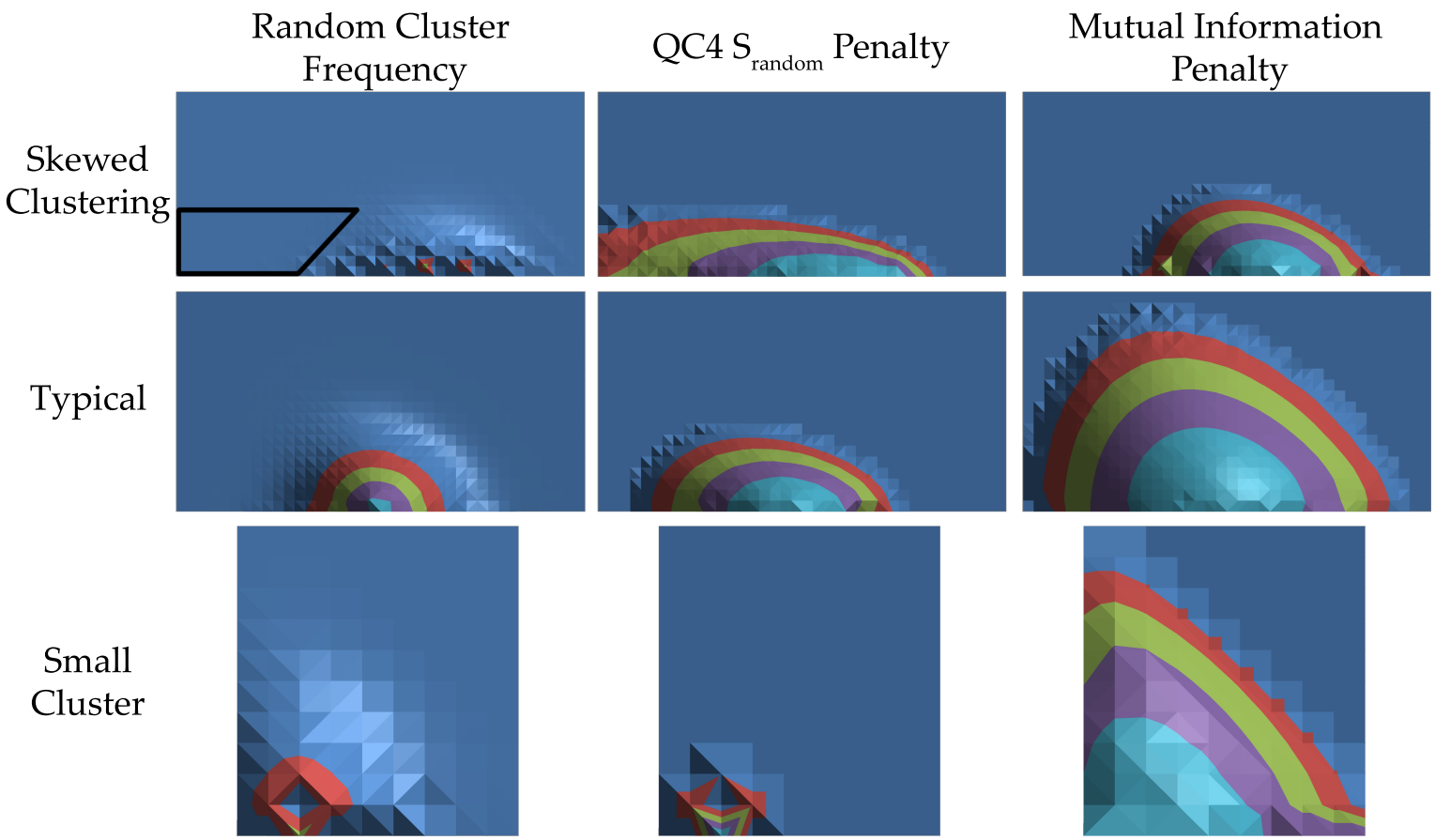

Figure 4.39: A 2d-transformation of cluster-space that focuses on the region containing random clusters. Each point represents a set of structurally identical clusters (section 4.2.8) and the height reflects the frequency of (column 1), or the degree of penalty (columns 2 and 3) applied to, the clusters in each set. The region outlined in black also contains random clusters, but the plot lacks the resolution needed to display them.

because they would be similar to the original result set, and therefore there is little risk in penalizing them.

In the bottom row, QC4 applied no penalty to many clusters that occur at random frequently. Because the clusters are small, there is less distance between the good clusters and the random clusters, and therefore, the random generation of clusters is more likely to produce good clusters. In fact, in this case, some of the random clusters were good clusters. Consequently, it is correct for QC4 to leave many of the random clusters unpenalized in this case.

For comparison, the right column shows the penalty applied by an alternate method based on Mutual Information. Mutual Information was the only standard measurement to identify the giant and random clusterings correctly in the synthetic test cases described in section 4.4.4. The method was normalized to account for its variable maximum score, the ideal clustering was transformed into a partition to account for overlap, and a threshold was applied so that only the worst clusters were penalized.

In all three cases, the alternate method based on Mutual Information applied the strongest penalty to the most frequently occurring random clusters. However, in the middle row and the bottom row, it penalized many real clusters and many good clusters. In the top row, it penalized some reasonable clusters near the bottom right corner of the plot and failed to penalize many of the less frequent random clusters in the bottom left corner.

4.7.3 Discounting Segmented Clusters

Segmented clusters contain a subset of the documents from a topic and as explained in section 4.2.5, should have lower quality. To account for segmented clusters, QC4 scales down the quality measure of clusters that are much smaller than the topic they focus on using the segmentation mea-

sure $S_{segment}(c)$. As noted in section 4.4.4, the worthless singleton clustering represents the extreme case of segmentation (where clusters contain just a single document) and so in addressing segmentation QC4 addresses the singleton clustering too.

4.7.3.1 Segmentation Measure

Assume the individual segment measure, $S(c,t)$, measures how much of topic t is contained in cluster c. $S(c,t)$ would be 1 if c contains all of t, and 0 if c contains none of t. If a cluster were pure, the individual segment measure would be a good way to discount for segmentation. However, clusters are often impure and contain documents from multiple topics. To account for this impurity the segmentation measure, $S_{segment}(c)$, averages the individual segment measure over all topics and weights them by Modified Precision, $P'(c,t,t_{best})$, to account for the degree to which the cluster focuses on each topic. The segmentation measure uses Modified Precision to account for overlapping topics and computes the average over all topics at the level of the cluster, $T^{L(c)}$, to account for hierarchical topics.

$$S_{segment}(c) = \max_{t_{best} \in T^{L(c)}} \left\{ \sum_{t \in T^{L(c)}} P'(c,t,t_{best}) S(c,t) \right\}$$

4.7.3.2 Individual Segment Measure

The individual segment measure, $S(c,t)$, should be lower when the segment is smaller. If the relationship between the segment size and the topic size were linear, Recall, $R(c,t)$, would make a good individual segment measure. However, the relationship is non-linear because a cluster with 90% of a topic's documents is only marginally more useful than one with 80%, whereas one with 20% is much more useful than one with 10%. Therefore, the individual segment measure should have a decreasing first derivative.

The situation is complicated further because particularly small clusters

are inherently less useful, as the range of documents in a very small cluster is unlikely to be representative of its topic and users may have trouble understanding the cluster's intention. With a decreasing first derivative, the marginal value of adding a document is highest for the first document and lowers for each subsequent document. However, the marginal value of the first document is less than the second, which is less than the third, and so on, until the cluster is large enough to represent the topic. Therefore, the individual segment measure's first derivative should increase until an inflection point that occurs when the cluster is sufficiently large and then decrease.

A first version of the individual segment measure, $S_1(c,t)$, meets the aforementioned requirements and as shown by the blue curve in figure 4.40, it has a first derivative of the desired type.

$$S_1(c,t) = 2 \cdot 2^{\left(\frac{-1}{R(c,t)}\right)} = 2^{\left(1 - \frac{1}{R(c,t)}\right)}$$

The inverted Recall represents the value left to capture from the topic and has a range of $[1, \infty)$. Exponentiating the negative of this creates a function with a range of $[0, 0.5]$ with the desired first derivative. Multiplying by 2 normalizes the range to $[0, 1]$.

4.7.3.3 Accounting for Limited User Time

The absolute cluster size is also important, because the time users have to examine the cluster contents is limited. As discussed in section 4.2.9, quality should account for limited user time. Specifically, where two clusters contain the same fraction of their respective topics, the cluster with the larger topic should have higher quality, and consequently, less discount due to segmentation.

For a given fraction, $S_1(c,t)$ is invariant to changes in $|D_t|$, and consequently, does not account for limited user time. Additionally, the inflection point of $S_1(c,t)$ always occurs when $R(c,t) = \frac{1}{2}\log_e 2 \approx 0.35$, but for larger

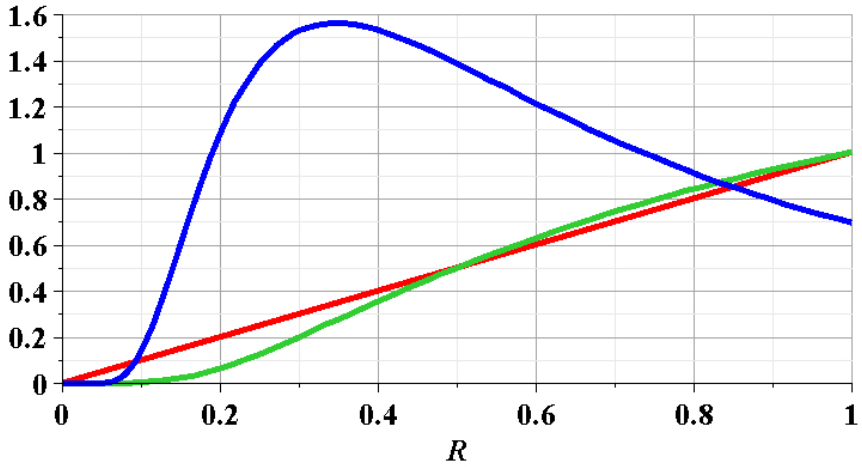

Figure 4.40: A plot showing clusters with various levels of recall R. The red curve shows Recall, $R(c,t)$, the green curve shows $S_1(c,t)$, and the blue curve shows the first derivative of $S_1(c,t)$.

topics, a smaller fraction of documents will be representative of the topic. To account for both of these problems, QC4 creates $S_2(c,t)$, which adjusts the value left to capture by a measure of the value in capturing a fixed portion of the topic, $\frac{1}{\log_2 |D_t|}$.[17]

$$S_2(c,t) = 2^{\left(\frac{-1}{R(c,t)\log_2 |D_t|}\right)}$$

$S_2(c,t)$ is then normalized to $[0,1]$ to give the individual segment measure, $S(c,t)$.

$$\begin{aligned} S(c,t) &= 2^{\left(\frac{1}{\log_2 |D_t|}\right)} \cdot 2^{\left(\frac{-1}{R(c,t)\log_2 |D_t|}\right)} \\ &= 2^{\left(\frac{1-\frac{1}{R(c,t)}}{\log_2 |D_t|}\right)} \end{aligned}$$

[17] When the topic is a singleton, the value is redefined as 1.

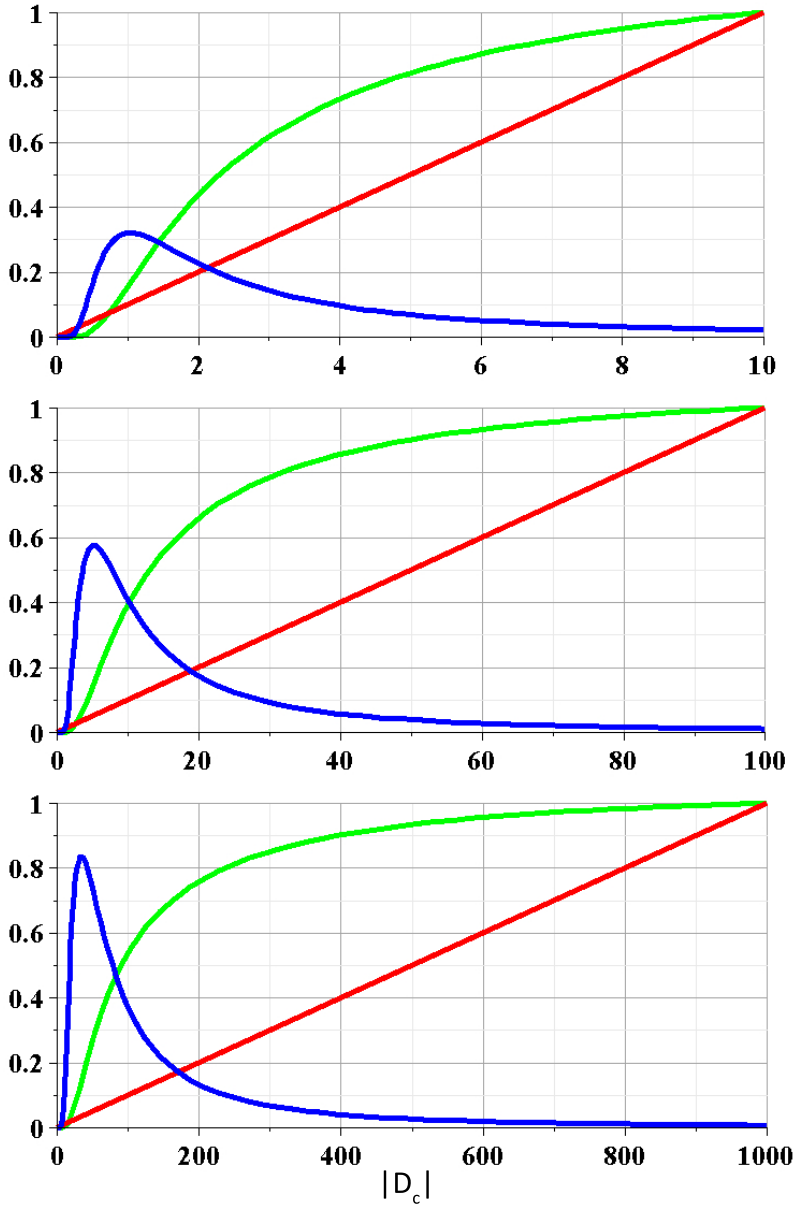

Figure 4.41: The three plots correspond to topics with 10, 100, and 1000 documents respectively and show clusters of varying size, $|D_c|$. The red curves show Recall, $R(c,t)$, the green curves show the individual segment measure, $S(c,t)$, and the blue curves show the first derivative of the individual segment measure, scaled to fit the plots, $|D_t| \cdot \frac{\partial}{\partial |D_c|} S(c,t)$.

Figure 4.41 shows that $S(c, t)$ does account for limited user time. Specifically, for a given fraction, the quality is higher for larger topics, as shown by the green curve being higher at all levels of Recall (the red curves) for larger topics. Additionally, in larger topics, a smaller fraction is considered representative, as shown by the inflection points in the blue curves shifting left for larger topics — for 10, 100, and 1000 documents the inflection points occur at 10.4%, 5.2%, and 3.5% Recall respectively.

4.8 New Coverage Measure

As section 4.4 explained, none of the existing measurements has all the properties required of a good coverage measure as set out in section 4.2. Therefore, QC4 introduces a new measure of topic coverage.

QC4's measure of topic coverage is an improved version of Recall that

1. uses Precision to account for the composition of clusters.

2. is computed recursively at the level of individual documents to account for overlapping and hierarchical topics.

3. is discounted to account for random clusterings, singleton clusterings, giant clusters, and segmented clusters.

4.8.1 An Improved Version of Recall

QC4 accounts for overlapping and hierarchical topics by computing overall coverage using the topic coverage of just the top-level topics, as described in section 4.6.2. This assumes the measure of individual topic coverage incorporates the coverage of child topics and addresses any overlap between child topics. QC4's measure of Topic Coverage must account for this requirement in addition to those of a good coverage measure described in section 4.2.

QC4's Topic Coverage is based on Recall, $R(c,t)$, in that it measures the fraction of a topic that is covered. Standard Recall is suitable when the clusters are pure and the topics are flat (non-hierarchical), but fails on worthless clusterings, segmented clusters, hierarchical topics, and does not account for cluster composition. QC4 must account for these deficiencies.

Recall uses the presence of a document in a cluster to indicate a document is covered. Topic Coverage, $CV(t)$, extends this notion and measures the extent to which each individual document is covered using Document Coverage, $DC(d,t)$, which accounts for coverage in a cluster representing the topic or a sub-topic, overlap between sub-topics, and addresses the deficiencies of Recall.

$$CV(t) = \frac{\sum_{d \in D_t} DC(d,t)}{|D_t|}$$

Document Coverage, $DC(d,t)$, is computed recursively down the levels of the topic hierarchy, using a measure of the Individual Document Coverage, $IDC(d,t)$, which will be defined in section 4.8.4.

4.8.2 Accounting for Cluster Composition

As described in section 4.2.4, QC4 must account for cluster composition when computing Topic Coverage.

For a document in a topic to be covered, it must be discoverable by a user, and therefore, a document is covered to some extent provided some cluster contains it. However, users do not explore all clusters; instead, they examine clusters that represent the topic of interest, and consequently, they are unlikely to find a document in a cluster that appears to be associated with a different topic. Therefore, a document will be better covered if it is contained in a cluster that matches a topic containing the document and the better the match, the better the coverage.

In reality, users determine what clusters represent using cluster labels or descriptions, which are often generated by another algorithm that should

be subject to an independent evaluation. Separating these concerns requires a proxy for the likelihood of a user examining a cluster when seeking a given topic. In general, algorithms generate cluster labels based on cluster composition. The probability that users can determine that a cluster is associated with a topic is dependent on the fraction of the cluster from the topic, or more specifically, the Precision, $P(c,t)$, of the cluster with respect to the topic.

The first version of Individual Document Coverage, $IDC_1(d,t)$, accounts for cluster composition by finding the Precision of the best matching cluster that contains the document.

$$IDC_1(d,t) = \max_{c \in C_d} \{P(c,t)\}$$

where C_d is the set of clusters containing document d.

4.8.3 Computing Document Coverage Recursively

If cluster composition were the only concern, $IDC_1(d,t)$ would be a good measure of Document Coverage. However, Document Coverage must also account for overlapping and hierarchical topics, as discussed in sections 4.2.6 and 4.2.7 respectively.

4.8.3.1 Accounting for Hierarchical Topics

When presented with hierarchical clusters, users start by choosing the most specific cluster that is relevant and if they fail to find sufficiently many relevant documents, they proceed to look at progressively less specific, but still relevant clusters. For example, a user who was interested in Indy Car Racing that was presented with clusters corresponding to the hierarchical topics in figure 4.42, would start by looking at the Indy cluster, and failing[18] that, the Racing cluster, and finally, the Car cluster.

[18]Failure may occur because a cluster does not exist or because it contains insufficient relevant documents.

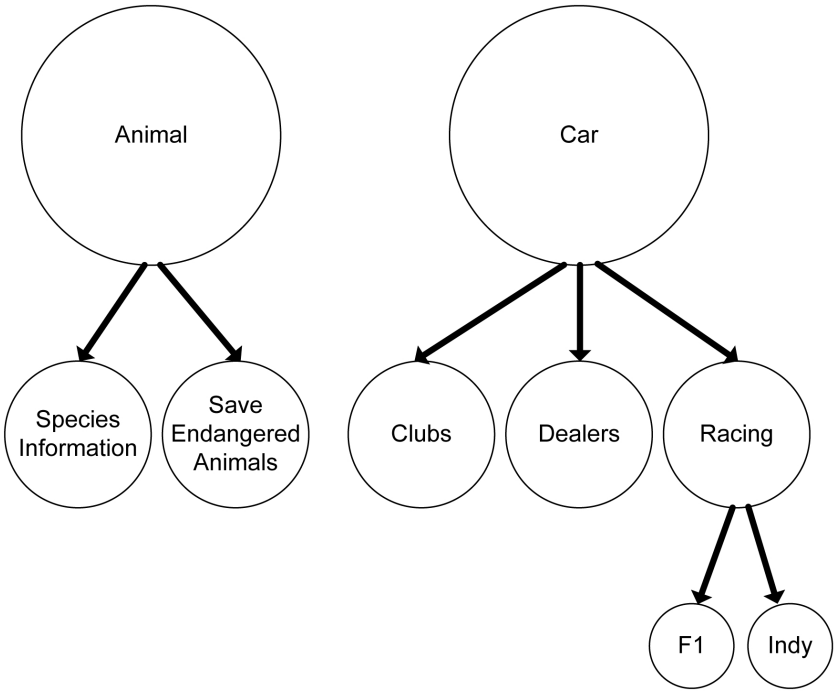

Figure 4.42: A subset of one possible topic hierarchy for the query "jaguar"

Consequently, a document in a topic is covered to some extent when a cluster contains the document and the cluster represents either the topic, or a sub-topic (that contains the document) of the topic. This definition applies recursively, as the document will be discoverable at any level of the hierarchy. For example, a document relating to Indy will be covered in the Car topic of figure 4.42 if it is contained in at least one of the Car, Racing, or Indy clusters.

Based on this recursive definition, the first version of Document Coverage, $DC_1(d,t)$, finds the maximum individual document coverage over the topic and all sub-topics (that contain the document) of the topic. $DC_1(d,t)$ uses a second version of Individual Document Coverage, $IDC_2(d,t)$, which is the first version, $IDC_1(d,t)$, modified to account for the hierarchy by

only considering clusters at the same level as the topic.

$$DC_1(d,t) = \max_{t\prime \in T_d \cap sub(t)} \left\{ IDC_2(d,t\prime) \right\}$$

$$IDC_2(d,t) = \max_{c \in \{c_i \in C_d | L(c_i) = lvl(t)\}} \left\{ P(c,t) \right\}$$

where T_d is the set of topics containing document d, $sub(t)$ is the set of topics containing t and all descendants of topic t, and where $lvl(t)$ is the level of topic t.

4.8.3.2 Accounting for Overlap

Document Coverage must also account for overlap between clusters and between topics. $DC_1(d,t)$ accounts for overlap between clusters by evaluating all of them and using the one that provides maximum coverage, but does not account for overlap between topics.

QC4 accounts for the overlap between top-level topics using the overall coverage measures, which aggregate the coverage of individual top-level topics and therefore require documents in multiple top-level topics to be covered by multiple clusters, representing the different topics, if complete coverage is to be achieved. This is in contrast to $DC_1(d,t)$, where documents in multiple lower-level topics need only be covered by a cluster representing one of them. For example, if the home page of an Indy Club were contained in a cluster representing the Racing topic, but not in any clusters representing the Clubs topic; the document will only be covered for half the potential users[19] of the document, yet $DC_1(d,t)$ may give it complete coverage.

To account for overlap, the definition of when a document is covered requires refinement. Specifically, a document in a topic is covered to some extent when clusters contain the document and the clusters represent either the topic, or all the direct sub-topics (that contain the document) of the

[19]This assumes an equal number of users are interested in the Clubs and Racing topics.

topic. This revised definition continues to apply recursively. For example, the home page of an Indy Club is covered if it is contained in

- clusters representing Car, OR
- clusters representing Clubs, AND either
  - clusters representing Racing, OR
  - clusters representing Indy.

To account for this, the final version of Document Coverage, $DC(d,t)$, recursively computes the coverage of a document in a topic using an average of the Individual Document Coverage of the topic's direct sub-topics. (Note that $IDC(d,t)$ is the final version of the Individual Document Coverage measure that will be defined in section 4.8.4.)

$$DC(d,t) = DC(d,t,1)$$

$$DC(d,t,l) = \frac{\sum_{t_\prime \in T^l \cap T_d \cap sub(t)} \max\left\{IDC(d,t_\prime), DC(d,t_\prime,l+1)\right\}}{|T^l \cap T_d \cap sub(t)|}$$

where l denotes the level in the topic hierarchy.

With some simplifying assumptions, figures 4.43 and 4.44 illustrate the recursive process of computing Document Coverage using $DC(d,t)$, for all the documents in a clustering.

4.8.4 Discounting Worthless Clusterings

Although Document Coverage, $DC(d,t)$, accounts for cluster composition, overlapping clusters, overlapping topics, and hierarchical topics, if it used $IDC_2(d,t)$ as its measure of Individual Document Coverage, it would still fail on worthless clusterings such as random clusterings, singleton clusterings, giant clusters, and would still not account for segmented clusters.

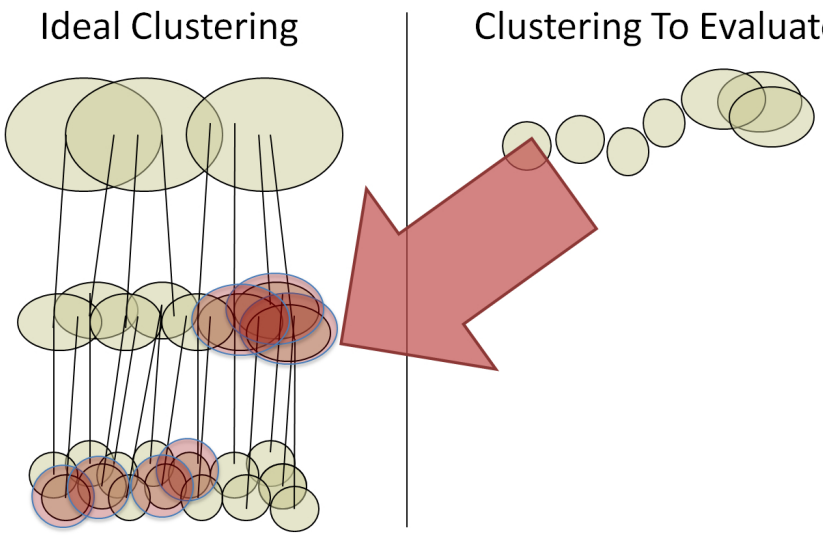

Figure 4.43: Step 1: Find the best matching between topics and clusters for each document

The final Individual Document Coverage measure, $IDC(d,t)$, accounts for the remaining faults by scaling down the Individual Document Coverage using two measures: $S_{random}(c)$ and $S_{tranche}(c)$.

$$IDC(d,t) = \max_{c \in \{c_i \in C_d | L(c_i) = lvl(t)\}} \left\{ P(c,t) \min\{1, S_{random}(c), S_{tranche}(c)\} \right\}$$

$S_{random}(c)$ was defined previously as part of QC4's quality measure in section 4.7.2 and accounts for worthless random clusterings and worthless giant clusters, while $S_{tranche}(c)$ is new and accounts for segmented clusters and worthless singleton clusterings.

4.8.5 Accounting for Segmentation

Coverage measures assume that users can find the most relevant cluster easily. When there are many clusters, this is no longer true, because the time users have to examine the cluster labels and descriptions is limited. However, relatively large clusters stand out, so users can quickly determine if they are relevant. Additionally, larger clusters are more valuable

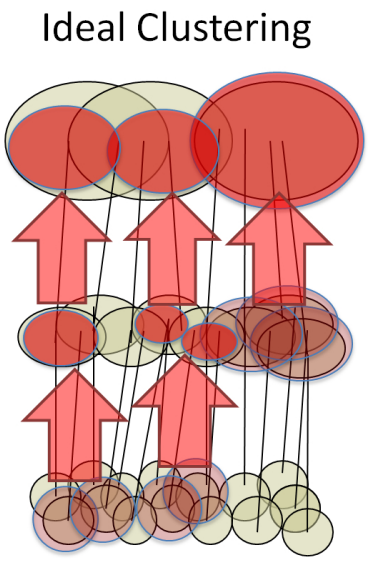

Figure 4.44: Step 2: Bubble the Individual Document Coverage up the topic hierarchy to find the top-level Document Coverage

once identified, as explained in section 4.2.5. Therefore, when discounting for segmentation (using $S_{tranche}(c)$), large clusters should get relatively smaller discounts than small clusters: the discount for a cluster should depend on both the number of clusters and its relative size.

As explained in section 4.7.3 for QC4's quality measure, discounting segmented clusters also addresses the worthless singleton clustering because this represents the extreme case of segmentation.

4.8.5.1 Divide Clusters into Tranches

As discussed in section 4.2.5, clusters of a similar size are difficult for the user to distinguish and should therefore get comparable discounts. For example, clusters with 9, 10, 11, or 12 documents are comparable, but are quite different from a 2-document cluster and a 100-document cluster. Furthermore, the relationship between cluster size and similarity is non-linear. For example, clusters with 10 documents and 100 documents

are quite different, but clusters with 100,000 documents and 200,000 documents are quite similar.

To represent the relative size of clusters, QC4 divides the clusters into tranches with boundaries based on the powers of 4.

$$tranche(c) = \lceil \log_4 |D_c| \rceil$$

The effect on clusters with up to 1000 documents is to divide them into 6 tranches containing clusters with 1, 2–4, 5–16, 17–64, 65–256, and 257–1024 documents respectively.

The rank of a tranche, $rank(n)$, is defined to be the rank of its largest cluster when all clusters are ordered by size. For example, if tranche one (with the smallest clusters) had 10 clusters, tranche two had 15 clusters, and tranche three (with the largest clusters) had 5 clusters, then the rank of tranche three would be 0, the rank of tranche two would be 5, and the rank of tranche one would be 20.

$$rank(n) = \sum_{i=n+1}^{\infty} |\{c \in C | tranche(c) = i\}|$$

4.8.5.2 Discount Segmented Clusters

Generally, users have sufficient time to examine the largest clusters and they should expect to examine at least as many clusters as there are topics. Consequently, QC4 should not penalize the largest clusters and it achieves this by not penalizing clusters in tranches with a rank less than or equal to the larger of 15 and the number of topics.

$$rank(tranche(c)) \leq \max\{15, |T|\}$$

Ideally, QC4 would penalize clusters in tranches with a rank greater than the number of topics, but this would unfairly penalize existing algorithms that create clusterings with a small fixed number of clusters. Therefore, at present, QC4 does not penalize algorithms that create fewer than

15 clusters. Arguably, it is reasonable for algorithms to create superfluous clusters, provided the total number of clusters is small, on the basis that users can successfully comprehend a small number of items simultaneously [125]. However, users probably prefer clusterings without superfluous clusters and therefore, as algorithms move to better dynamic cluster selection strategies, QC4 should penalize algorithms that produce even a small number of superfluous clusters.

As larger clusters have more value once identified, the penalty should also account for cluster size, $|D_c|$. The penalty for segmented clusters, $S_{tranche}(c)$, is the ratio of cluster size to the rank of the cluster's tranche, because clusters with a larger rank are harder to find. Therefore, clusters with low rank and clusters with a large size relative to their rank are not penalized, while the remainder are discounted according to their size and relative rank.

$$S_{tranche}(c) = \begin{cases} 1 & if\{rank(tranche(c)) \leq \max\{15, |T|\}\} \\ \\ \dfrac{|D_c|}{rank(tranche(c))} & otherwise \end{cases}$$

4.9 Evaluation

I evaluated QC4 in three ways: synthetic testing, real world web page clustering, and comparative hierarchical clustering. In all three, QC4 always produces the expected result and draws the correct conclusion; none of the existing measurements can make this claim.

The synthetic evaluation mirrors the evaluation of the existing measurements discussed in section 4.4 and shows that QC4 meets all the requirements of good evaluation measurements from section 4.2, unlike the existing measurements that fail many of those requirements.

On typical clusterings, for the most part, the existing measurements produce valid conclusions regarding the relative performance of different clustering algorithms. The real world web page clustering evaluation

shows that QC4 mostly agrees with the existing measurements and therefore, it too produces valid conclusions. Where disagreements occur, qualitative analysis shows that QC4 is correct as to the relative performance of algorithms.

The comparative hierarchical clustering evaluation compares real world clusterings against just the top-level or second-level topics. Good measurements should show a corresponding effect on quality and coverage, but only QC4 and Cluster F-measure perform correctly.

4.9.1 Synthetic Testing

To test the characteristics of good evaluation measurements as described in section 4.2, I constructed a suite of synthetic clusterings that cover edge and boundary cases, as well as other more normal clusterings. None of the existing measurements passed all these tests as shown earlier by table 4.11. QC4 passed all the synthetic clustering tests as shown by table 4.13.

Section 4.4 presented many interesting test cases that proved problematic for some measurements; tables 4.14 and 4.15 present QC4's results for those cases for comparison and completeness.

While the existing measurements can still produce meaningful results and conclusions, the synthetic tests show that there are conditions under which the existing measurements can produce inaccurate results. While most algorithms do not produce worthless clusterings, the other problem cases are concerning because web page clusterings often have these characteristics. Consequently, the conclusions drawn from the existing measurements are questionable and it is safer to evaluate web page clustering algorithms using QC4 or another measurement that meets all these properties.

Table 4.13: A summary of the properties satisfied by QC4. Green indicates the measurement has the property, red indicates it does not, and white indicates the property is not applicable to the measurement.

	AQ	WQ	AC	WC
Separate Measures				
Basic Quality				
Basic Coverage				
Size Bias				
Perfect Clusterings				
Worthless - Singleton				
Worthless - Giant				
Worthless - Random				
Composition - Quality				
Composition - Coverage				
Segmentation - Quality				
Segmentation - Coverage				
Overlapping Clusters				
Overlapping Topics				
Hierarchical Topics				
Limited User Time				

4.9.2 Real World Web Page Clustering

The real world evaluation used 17 measurements (Mutual Information, and Average and Weighted versions of QC4 Quality, QC4 Coverage, Precision, Total Recall, Best Recall, Cluster F-measure, Topic F-measure, and Entropy) to compare the clustering performance of five web page clustering algorithms (Random Clustering, K-means [166], STC [203], ESTC [44], and QDC[20] [46]) on eight data sets: search results of four different

[20]QDC was developed as part of this thesis and is discussed in chapter 5.

Table 4.14: Synthetic test case results for QC4

Quality	AQ	WQ	AC	WC
4.19.A	0.799	0.799	0.08	0.009
4.19.B	0.675	0.716	0.12	0.014
4.19.C	0.675	0.716	0.12	0.014
4.19.D	0.554	0.555	0.16	0.018
4.19.E	1	1	0.2	0.023
4.20.A	0.994	0.994	0.184	0.836
4.20.B	0.639	0.983	0.34	0.836
4.20.C	0.751	0.625	0.341	0.134
Singleton	0.003	0.003	0.001	0.001
Giant	0	0	0	0
Random	0.017	0.017	0.004	0.017
4.23.A	0.309	0.28	0.26	0.088
4.23.B	0.412	0.278	0.162	0.098
4.25.A	0.579	0.579	0.166	0.166
4.25.B	0.412	0.412	0.151	0.151
4.25.C	0.732	0.732	0.181	0.181
4.26.A	0.992	0.992	0.18	0.818
4.26.B	0.919	0.919	0.18	0.818

queries (“salsa”, “jaguar”, “gp”, and “victoria university”) using both full-page and snippet data.

Figures 4.45, 4.46, and 4.47 show the results averaged across the clustering tasks and with the average and weighted versions of each measurement combined by averaging them. The Overall QC4 measurement shown in figure 4.47 is the average of QC4 Quality (Q) and QC4 Coverage (C). The figures show the algorithms in order of overall performance according to the relevant research literature [203, 44, 46], which ranks the algorithms as QDC, ESTC, STC, K-means, and finally Random Clustering. As

Table 4.15: More synthetic test case results for QC4

Quality	AQ	WQ	AC	WC
4.8 LHS	1	1	1	1
4.8 RHS	0.012	0.29	0.438	0.438
4.8 LHS Blue	1	1	0.333	0.333
4.8 RHS Blue	0.332	0.838	0.333	0.333
4.8 RHS Green	0.016	0.027	0.136	0.136
4.8 RHS Purple	0.001	0.006	0.065	0.065
4.28.A	1.0	1.0	1.0	1.0
4.28.B	0.987	0.989	0.92	0.92
4.28.C	0.974	0.974	0.8	0.8
4.30.A	0.581	0.581	0.089	0.125
4.30.B	1	1	0.779	0.81
4.31.A	1	1	1	1
4.31.B	1	1	1	1
4.31.C	1	1	1	1
4.31.D	1	1	1	1
4.32.A	0.033	0.033	0.004	0.018
4.32.B	0.188	0.188	0.008	0.036
4.32.C	0.674	0.674	0.03	0.136
4.32.D	0.85	0.85	0.06	0.273

shown by figure 4.47, QC4 ranks the algorithms correctly. The algorithms STC [203] and ESTC [44] are optimized for snippets and full text respectively, and consequently, they should perform better on those; QC4 ranks ESTC and STC correctly, ranking ESTC–Full Text above ESTC–Snippet and STC–Snippet above STC–Full Text.

These clustering tasks represent typical web page clustering problems where existing measurements work reasonably well and generally form valid conclusions. Consequently, the conclusions of QC4 should gener-

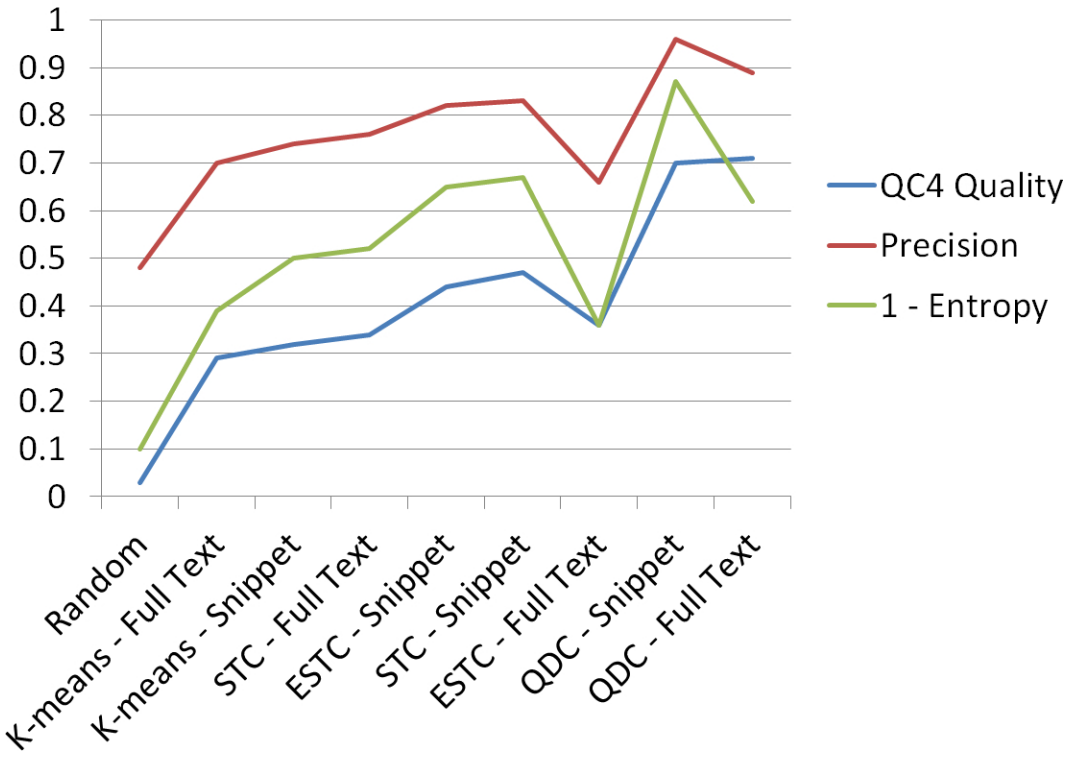

Figure 4.45: Comparison of evaluation results for different Quality measures

ally agree with the conclusions of the existing measurements. Table 4.16 confirms they do agree by reporting the degree of pairwise agreement between the measurements, where two measurements agree for a pair of algorithms if they order them in the same way. Specifically, the average pairwise agreement between QC4 and the other measurements (85%) is not significantly different from the average pairwise agreement between the other measurements (86%). Therefore, QC4 generally makes the same conclusions as other measurements regarding the relative performance of different clustering algorithms.

In many cases the disagreement between two measurements is insignificant[21], accounting for these, the pairwise agreement for QC4 is 94% and

[21] *i.e.* Under both measurements, the performance difference between the algorithms under disagreement is not significant ($p > 0.05$).

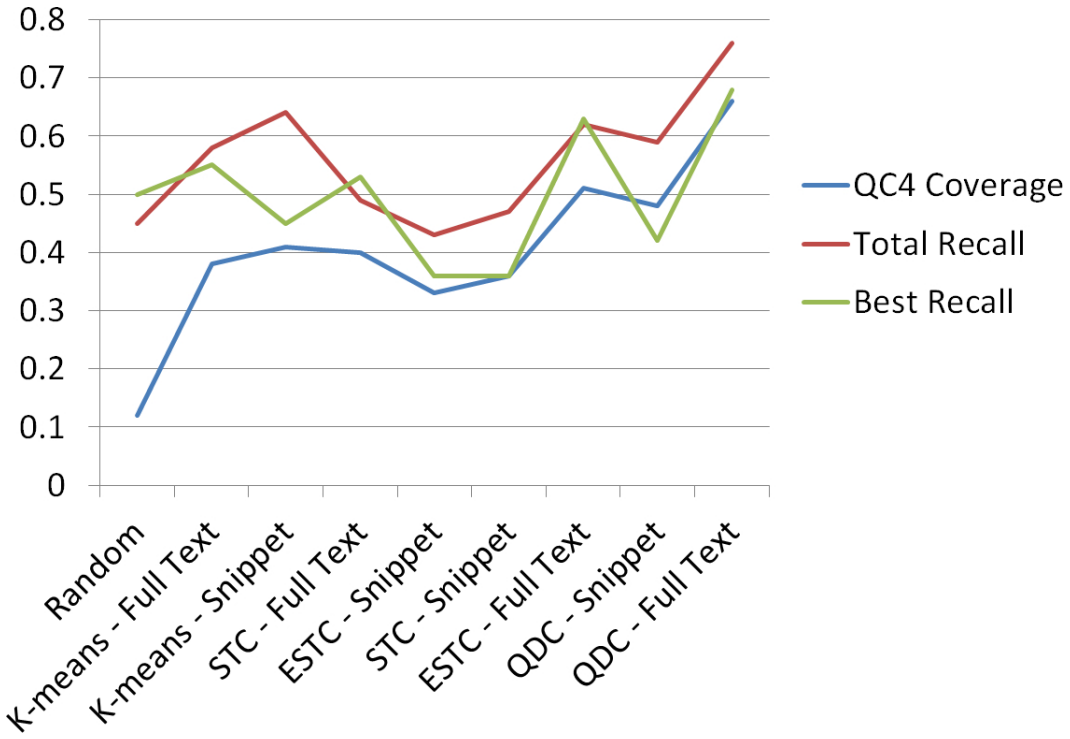

Figure 4.46: Comparison of evaluation results for different Coverage measures

for the other measurements it is 92%. Much of the remaining disagreement occurs because some of the measurements incorrectly evaluate the Random Clustering. Best Recall considers the Random Clustering algorithm superior to many better algorithms, Total Recall and Cluster F-measure consider it superior to some better algorithms, and while Precision and Topic F-measure do not consider it superior to any better algorithms, they give it unjustifiably high performance. After accounting for the Random Clustering algorithm, the pairwise agreement for QC4 is 97% and for the other measurements it is 96%.

In the remaining cases where QC4 disagrees with other measurements, qualitative analysis shows that QC4 reaches the correct conclusion when other measurements do not. QC4 correctly finds that ESTC outperforms STC, whereas Mutual Information finds the converse for full text data

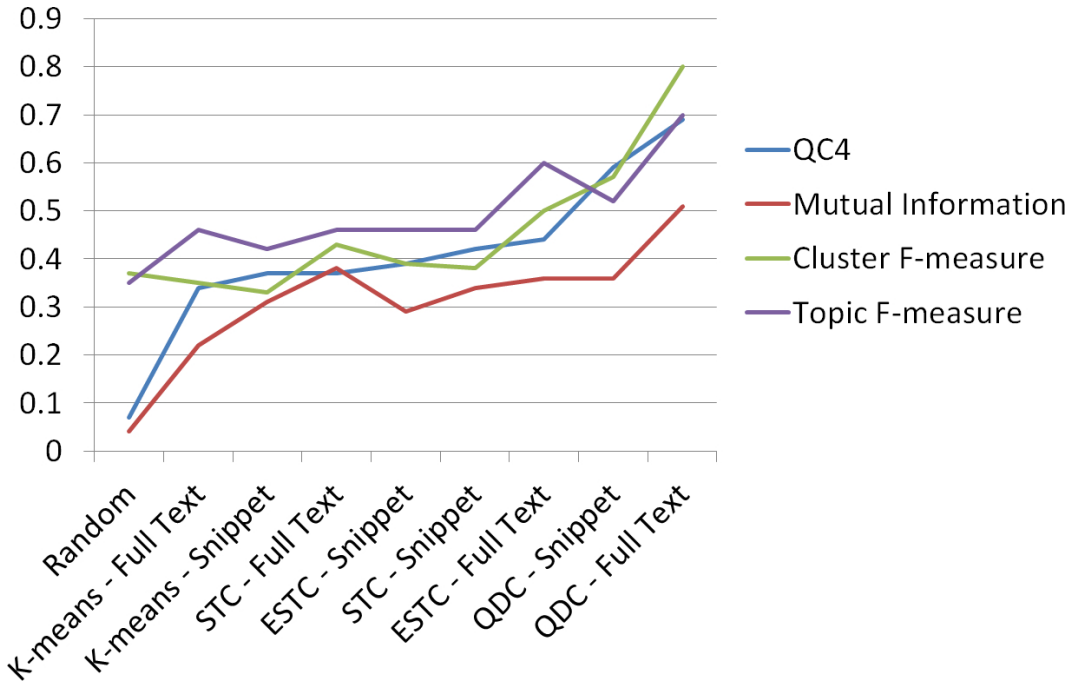

Figure 4.47: Comparison of evaluation results for different Overall measures

because Mutual Information fails on the overlapping clusters produced by STC. QC4 correctly identifies that QDC–Snippet outperforms ESTC–Full Text, whereas Topic F-measure finds the converse because Topic F-measure overemphasizes the mere appearance, rather than correct placement of documents in clusters.[22] For a similar reason Total Recall gives unjustifiably high performance to K-means and incorrectly finds K-means–Snippet to outperform ESTC–Full Text and QDC–Snippet. Best Recall's results lack validity because it finds the Random Clustering algorithm superior to many other algorithms and this shows up in its level of disagreement with other measurements: its agreement with QDC is only 72% and just 78% with other measurements, which is the lowest level of agreement between any measurements.

[22]This relates to Topic F-measure's failures with Cluster Composition as discussed in section 4.4.5.

Table 4.16: The degree of pairwise agreement between measurements

	Q	P	E	
Q	100%	89%	83%	
P		100%	94%	
E			100%	
	C	R^{Σ}	R^1	
C	100%	89%	72%	
R^{Σ}		100%	78%	
R^1			100%	
	QC4	MI	F^C	F^T
QC4	100%	86%	83%	94%
MI		100%	81%	86%
F^C			100%	92%
F^T				100%

The experiments suggest that QC4 makes correct conclusions on real world clustering tasks, even when the existing measurements have problems. The unjustifiably high performance of random clusters under Precision, Best Recall, Total Recall, Cluster F-measure, and Topic F-measure is concerning and should be seriously considered whenever these measurements are used. Their susceptibility depends on the ideal clustering and some configurations can produce dubious results. It is advisable to avoid Best Recall completely due to its lack of validity for a wide variety of ideal clustering configurations.

4.9.3 Comparative Hierarchical Clustering

When a measurement properly accounts for hierarchical topics, the performance of a cluster should differ when compared to just top-level topics or just second-level topics. The comparative hierarchical evaluation uses this

property to compare the results of measurements for the entire hierarchy, just the top-level, and just the second-level. When the results do not differ as expected, the measurement is not accounting for the hierarchy.

The performance of a cluster should differ as shown in table 4.17. When comparing top-level clusters to second-level topics, quality and coverage should decrease as the documents in each cluster will divide among more topics. When comparing second-level clusters to top-level topics, coverage should increase as previously irrelevant documents become relevant. When comparing second-level clusters to top-level topics, the impact on quality is indeterminate as two opposing factors interact. Quality increases as previously irrelevant documents become relevant, but quality decreases due to increased segmentation: second-level topics are smaller than top-level topics, so second-level clusters will typically represent a smaller fraction of a top-level topic and consequently have greater segmentation. In general, the segmentation effect is more significant and the net effect will be for quality to decrease.

Table 4.17: The effect on quality and coverage of comparing a cluster against a single level of the hierarchy

Cluster	Topics	Quality	Coverage
Top-level	Top-level	Same	Same
Top-level	Second-level	Down	Down
Second-level	Top-level	Indeterminate	Up
Second-level	Second-level	Same	Same

QDC, ESTC, and STC produce predominantly top-level clusters — 100%, 100%, and 80% respectively — while K-means produces predominantly second-level clusters, between 70% – 80%. Figures 4.48, 4.49, and 4.50 show the difference, for each measurement, between the result of comparing the algorithm's clusterings to the entire hierarchy and either the top-level topics (first column) or second-level topics (second column). A

yellow dash indicates the results are the same (insignificant difference), a red downwards arrow indicates the measure decreased significantly, and a green upwards arrow indicates the measure increased significantly.

	QDC 100% Top-Level		ESTC 100% Top-Level		STC 80% Top-Level	
	Top	2nd	Top	2nd	Top	2nd
AQ	0	-0.293	-0.009	-0.139	-0.003	-0.033
WQ	0	-0.383	-0.001	-0.223	0	-0.084
AP	0	-0.434	0	-0.378	0	-0.228
WP	0	-0.544	0	-0.416	0	-0.307
AE◊	0.36	0.018	0.398	0.124	0.283	0.166
WE◊	0.495	0.03	0.471	0.131	0.4	0.206

	K-means 78% Second-Level		K-means 70% Second-Level	
	Top	2nd	Top	2nd
AQ	-0.068	-0.005	-0.122	0.014
WQ	-0.027	0.058	-0.045	0.055
AP	0	-0.161	0	-0.1
WP	0	-0.111	0	-0.115
AE◊	0.241	0.19	0.218	0.181
WE◊	0.35	0.216	0.343	0.211

Figure 4.48: Quality results for the comparative hierarchical evaluation

Figure 4.48 shows the results for the quality measures. Since QDC, ESTC, and STC produce mostly top-level clusters, their quality should stay the same when compared to only top-level topics and reduce when compared to only second-level topics. QC4's Average and Weighted Quality measures, highlighted in grey, produce the expected result. Although the decrease when comparing STC to the second-level topics is not significant, it is correct because 20% of STC's clusters are second-level and they offset the result. Since K-means produces mostly second-level clusters, its qual-

ity should generally decrease when compared to top-level topics and stay the same when compared to second-level topics. QC4's measures produce the expected result.

Precision and Entropy produce the incorrect results because they do not account for hierarchical topics. Although Precision produces the expected result for QDC, ESTC, and STC, it produces the incorrect result for K-means. When comparing to the entire hierarchy, Precision always compares to just the top-level topics because it compares to the best performing topics, rather than the best matching topics. Consequently, there is no change when comparing to just the top-level topics and there is always a decrease when comparing to second-level topics. Entropy increases in all instances, because it does not account for overlapping topics and therefore underestimates performance when comparing to the entire hierarchy, since there is overlap between topics at different levels.

Figure 4.49 shows the results for the coverage measures. Since QDC, ESTC, and STC produce mostly top-level clusters, their coverage should stay the same when compared to top-level topics and decrease when compared to second-level topics. Since K-means produces mostly second-level clusters, its coverage should increase when compared to top-level topics and stay the same when compared to second-level topics. QC4's Average and Weighted Coverage measures, highlighted in grey, produce the expected result.

Total Recall and Best Recall produce the incorrect results because they do not account for hierarchical topics. Total Recall always increases when compared to top-level topics and generally decreases when compared to second-level topics. Best Recall generally stays the same when compared to either top-level topics or second-level topics.

Figure 4.49 shows the results for the overall measures. As overall measures combine quality and coverage, their effects should combine too. Since QDC, ESTC, and STC produce mostly top-level clusters, they should stay the same when compared to top-level topics and decrease when compared

	QDC 100% Top-Level		ESTC 100% Top-Level		STC 80% Top-Level	
	Top	2nd	Top	2nd	Top	2nd
AC	0	-0.445	0.002	-0.403	0.027	-0.336
WC	0	-0.505	0.005	-0.485	0.026	-0.398
AR^{Σ}	0.396	-0.076	0.333	-0.064	0.373	-0.071
WR^{Σ}	0.245	-0.229	0.213	-0.198	0.237	-0.22
AR^{1}	-0.127	0.024	-0.086	0.017	-0.082	0.016
WR^{1}	0.008	-0.007	-0.016	0.015	-0.015	0.014

	K-means 78% Second-Level		K-means 70% Second-Level	
	Top	2nd	Top	2nd
AC	0.202	-0.055	0.15	-0.069
WC	0.259	-0.034	0.181	-0.092
AR^{Σ}	0.22	-0.012	0.252	0
WR^{Σ}	0.214	-0.147	0.243	-0.114
AR^{1}	-0.114	0.023	-0.106	0.02
WR^{1}	-0.025	0.024	-0.026	0.024

Figure 4.49: Coverage results for the comparative hierarchical evaluation

to second-level topics. Since K-means produces mostly second-level clusters, it is indeterminate what should happen when compared to top-level topics and it should stay the same when compared to second level topics. Average and Weighted Cluster F-measure produce the expected result. Although the decrease when comparing K-means 70% to the second-level topics is significant, it is correct because 30% of the clusters are top-level and they offset the result.

Topic F-measure and Mutual Information produce incorrect results because they do not account for hierarchical topics. Topic F-measure mirrors Total Recall and increases when comparing clusters to top-level topics and decreases when comparing clusters to second-level topics. Mutual Infor-

	QDC 100% Top-Level		ESTC 100% Top-Level		STC 80% Top-Level	
	Top	2nd	Top	2nd	Top	2nd
AF^C	0	-0.259	-0.024	-0.169	-0.006	-0.127
WF^C	0	-0.393	-0.002	-0.251	-0.001	-0.222
AF^T	0.384	-0.074	0.321	-0.062	0.301	-0.058
WF^T	0.258	-0.241	0.222	-0.207	0.221	-0.205
MI	-0.174	-0.306	-0.44	-0.433	-0.331	-0.323

	K-means 78% Second-Level		K-means 70% Second-Level	
	Top	2nd	Top	2nd
AF^C	-0.116	-0.022	-0.12	-0.028
WF^C	-0.058	-0.057	-0.046	-0.106
AF^T	0.163	-0.065	0.108	-0.031
WF^T	0.106	-0.124	0.094	-0.097
MI	-0.209	-0.114	-0.222	-0.12

Figure 4.50: Overall results for the comparative hierarchical evaluation

mation decreases in all instances because like Entropy, it does not account for overlapping topics and additionally its maximum value is variable as discussed in section 4.4.3 and depends on the ideal clustering, which changes when comparing to just the top-level or just the bottom-level.

In summary, only QC4 and Cluster F-measure produce the correct result. This was expected, since the synthetic evaluation showed they were the only measures to account for hierarchical topics. These results provide further evidence that the problems with existing evaluation measurements can significantly affect conclusions about performance. The combination of evidence from the synthetic evaluation, the real world evaluation, and the comparative hierarchical evaluation justify the use and development of better measurements like QC4.

Chapter 5

Query Directed Clustering

Queries are often ambiguous: words and phrases are frequently polysemantic and search goals are often narrower in scope than the queries used to express them. This ambiguity leads to result sets containing distinct groups of pages that meet different search goals. Clustering can exploit the internal similarity between the vocabularies of these distinct groups to identify them and can then present these groups to the user to aide query refinement.

For ambiguous queries for easy searches (easy ambiguous queries), clustering methods produce some good clusters, albeit interspersed with ambiguous, overly specific, low value, and incomprehensible clusters. In contrast, clustering methods have largely proven ineffective for the interrelated results of non-ambiguous queries and hard ambiguous queries (the subject of later chapters in this thesis).

This chapter presents QDC, a query directed web page clustering algorithm that produces higher quality clusterings than other clustering algorithms for easy ambiguous queries. QDC has five key innovations:

1. A new query directed cluster quality guide that uses the relationship between clusters and the query to improve cluster quality

2. An improved cluster merging method that enhances cluster cover-

age while still generating semantically coherent clusters by using cluster description similarity in additional to cluster overlap

3. A new cluster splitting method that improves cluster quality by addressing the cluster chaining (cluster drift) problem

4. An improved heuristic for cluster selection that uses a query directed cluster quality guide to select a good representation of the top-level of a hierarchical clustering

5. A new method that improves clusters by ranking the pages by relevance to the cluster

5.1 Clustering and Easy Ambiguous Queries

Web Page Clustering has been applied to many kinds of queries. For most of these, clustering has some measurable value, but the value is too small and has no obvious benefit for users. In contrast, clustering is very effective (due to the distinct vocabulary) and clearly useful (clusters are semantically meaningful to users) for easy ambiguous queries.

5.1.1 Limitations of Clustering

Users must often refine their search by modifying the query to filter out the irrelevant results. To refine queries effectively users must understand the result set, but this is time consuming if the result set is unorganized.

A common approach to organize data is clustering. Web page clustering has been widely used to organize search results [14, 201, 123] and most approaches perform acceptably on the typical test set (easy ambiguous queries), which includes queries like “Jaguar”, “Apple”, “Saturn”, “Jordan”, “Tiger”, and “Salsa” [204, 70]. However, clustering performs poorly on many interesting queries (non-ambiguous queries and hard ambiguous queries). Specifically, researchers have found that clustering works well

for ambiguous terms, but not for entity names and general terms [204] and that it works well when the target clusters are unrelated (for instance, Lord of the Rings and MySQL), but performs poorly when the clusters are closely related (for instance, MySQL and PostgreSQL) [135].

5.1.2 Cluster Vocabularies

Clustering algorithms try to divide a result set into distinct clusters according to the vocabulary of the documents. They generally succeed if there are different groups of documents that have distinct vocabulary and these groups match concepts that are meaningful to users.

Ambiguous queries should be ideal candidates for clustering: they contain distinct document groups related to different query interpretations, which correspond to different search goals, so the groups would be meaningful to users. However, not all ambiguous queries are ideal for clustering, because the document groups related to different query interpretations do not necessarily have distinct vocabulary and may instead share a single common vocabulary.

Ambiguous queries divide into two main classes (easy and hard) based on the vocabulary overlap of different interpretations. For example, "Jaguar" is an easy ambiguous query with several interpretations including "car" and "animal". The different interpretations of "Jaguar" have little vocabulary in common; for instance, "car" documents use terms like "automobile" and "fuel", while "animal" documents use terms like "mammal" and "cat". "black bear attacks" is a hard ambiguous query with several interpretations including "information about black bears" and "instances of attacks on humans"; but the different interpretations of "black bear attacks" use a single vocabulary, for instance, terms like "wildlife" and "habitat".

Non-ambiguous queries have only one interpretation and therefore all documents share a single vocabulary, making the situation analogous to that of hard ambiguous queries. This leaves easy ambiguous queries as the

only case where different semantically meaningful document groups have distinct vocabulary, and consequently it is the only case where clustering can be expected to perform well.

5.2 Related Work

QDC reuses the best parts of several simple clustering algorithms to solve problems faced by the best web page clustering algorithms. QDC also exploits a semantic relatedness measure, Google Distance [40], to assess the quality of clusters.

5.2.1 Standard Clustering Algorithms

Researchers have applied all standard data clustering methods [14, 92, 166] to web page clustering: hierarchical (agglomerative and divisive), partitioning (probabilistic, k-Medoids, k-Means), Fuzzy c-Means, Bayesian, Kohonen self-organising maps, density based, and many more.

Hierarchical clustering algorithms [209, 68, 31] come in two varieties: agglomerative (merging) and divisive (subdividing). Agglomerative methods are bottom-up: they start with one data item per cluster and iteratively merge clusters according to some merging criteria. Divisive methods are top-down: they start with all data items in one big cluster and recursively split clusters according to some splitting criteria. Often the process is stopped when the desired number of clusters exist on the fringe — vagueness of termination is often considered a disadvantage.

Partitional clustering algorithms [17, 75, 203] decompose the data points into clusters that optimize some objective function. Greedy heuristics are typically used to create partitional clustering algorithms that find clusters by starting with some clustering and then iteratively relocating points between clusters [14].

While the standard algorithms perform poorly in the web page cluster-

ing domain, they often provide the under-pinning of more effective algorithms including Query Directed Clustering.

5.2.2 Similarity Measures

The standard clustering algorithms typically use data similarity measures [169, 14, 207, 167] to construct clusters; typical measures for the similarity[1] between documents include Minkowski distance[2], Cosine similarity, Pearson correlation[3], and extended Jaccard similarity.

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

$$\text{Pearson Correlation} = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \overline{y})^2}}$$

$$\text{Extended Jaccard Similarity} = \frac{\sum_{i=1}^{n} x_i y_i}{|\sum_{i=1}^{n} x_i^2| + |\sum_{i=1}^{n} y_i^2| - \sum_{i=1}^{n} x_i y_i}$$

where x and y are n-dimensional document vectors, p is any real number greater than or equal to one[4], and $\overline{x}$ and $\overline{y}$ are the mean value over all dimensions of x and y.

Unfortunately, the standard algorithms using the standard similarity measures are not effective at producing semantically meaningful clusters. Metric distances such as Minkowski are particularly ineffective due to the high dimensional and sparse nature of the textual domain [169]. While the other standard similarity measures are dramatically more effective than

[1] Distance and similarity are equivalent since similarity is the inverse of distance.
[2] Manhattan distance and Euclidean distance are special cases of Minkowski distance.
[3] Pearson Correlation is typically normalized to the range [0,1].
[4] p is typically 1 (Manhattan distance) or 2 (Euclidean distance).

Metric distances [169], they too are ineffective as the most similar documents are not necessarily semantically similar, because the semantic content of terms within a document is not homogeneous. QDC accounts for this non-homogeneity by using Google Distance to measure the semantic content of terms.

5.2.3 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering is a bottom-up clustering approach that treats each document as a singleton clustering and forms a hierarchical clustering by sequentially merging the most similar clusters according to one of three linkage metrics [120, 14, 179]: single linkage, group average (average-link), or complete linkage. Table 5.1 shows the merging criterion for each linkage metric.

Table 5.1: Linkage metrics for Hierarchical Agglomerative Clustering

Metric	Merging Criterion	Complexity
Single-link	smallest minimum pairwise distance	$O(n^2)$
Average-link	smallest average pairwise distance	$O(n^2 \log n)$
Complete-link	smallest maximum pairwise distance	$O(n^2 \log n)$

The hierarchical clustering can be represented by a dendrogram as shown in figure 5.1. Different clusterings may be found by cutting the dendrogram, stopping the algorithm when the distance exceeds some threshold.

Interestingly, single-link clustering with a threshold stopping criteria is equivalent to finding the connected components of a graph, where the nodes represent the documents and there is an edge between any two documents with similarity exceeding the stopping criteria.

Some clustering algorithms, including single-link clustering, are susceptible to cluster chaining (cluster drift) [212]. In a sequence of clus-

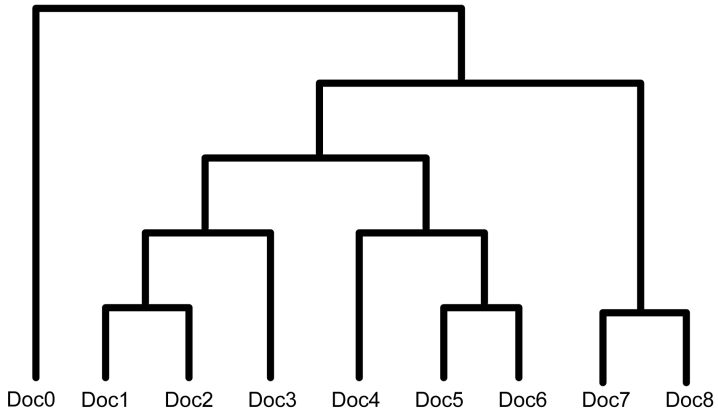

Figure 5.1: A dendrogram showing one possible hierarchical clustering of 9 documents. The vertical dimension corresponds to the distance between items/clusters.

ters, each cluster may be similar to its immediate neighbours, but completely dissimilar from clusters further away in the sequence. While a desirable property for some domains, it is undesirable in the textual domain: clusters obtained by merging such sequences are often of low quality and are not semantically meaningful. In contrast to single-link clustering, complete-link clustering is sensitive to outliers, while average-link clustering is a compromise between the two [120].

QDC uses an improved similarity measure for merging that limits cluster chaining significantly, but does not stop it entirely. To solve the cluster chain problem (without the alternate limitations of average-link or complete-link clustering) QDC makes a second pass over the merged clusters and splits those that have been joined inappropriately.

5.2.4 Partitional Clustering

Partitional clustering maximizes an objective function by iteratively relocating points between clusters [14]. There are several common approaches to partitional clustering. The probabilistic approach assumes the data was

sampled from a mixture model of several probability distributions. The log likelihood of the data provides the objective function for a two-step iterative EM (expectation-maximization) algorithm [14, 16]. The k-Medoids approach represents each cluster by one of its points and typical objective functions measure the similarity using the distance between a point and the medoid [14, 211, 167]. The k-Means approach [14, 31] represents each cluster by a centroid, which is the mean of the document vectors assigned to that cluster.

k-Means starts with an arbitrary assignment of documents to k clusters, then iteratively shifts documents to the nearest cluster. The algorithm iterates until convergence (typically when there were no changes in the last iteration) using a two-step process:

1. calculate the vector-space centroid of each cluster using its current members

2. assign each document to the cluster with the nearest centroid

Partitional clustering algorithms find a local optima based on the initial seed (the initial assignment of documents to clusters), which makes the selection of this seed critical. A commonly used and simple approach is to randomly assign documents to clusters. There are many alternative seed selections methods [120] that include

1. creating several random seeds and running the algorithm on each seed, then choose the clustering that minimizes the objective function

2. excluding outliers (documents with low similarity to all other documents) from the seed

3. creating the seed using another clustering algorithm such as hierarchical clustering

Both k-Means and hierarchical agglomerative clustering use hard assignments where each document belongs to exactly one cluster. Documents often reflect multiple concepts and ought to belong in multiple clusters; using hard assignment limits the potential performance by needlessly restricting the clusterings an algorithm can construct. An alternative is soft assignment, where each document has fractional membership to each cluster and may therefore exist in multiple clusters. Fuzzy c-Means is an adaptation of k-Means that uses soft assignment. Query Directed Clustering, like most web page clustering algorithms, uses soft assignment.

5.2.5 Web Page Clustering Algorithms

Web pages are a rich clustering domain with many sources of insight beyond the pages themselves. Many algorithms build on the standard methods by using web or document specific characteristics to assist clustering: Suffix Tree Clustering (STC) [201] and Lingo [137, 136] use phrases. Other algorithms exploit further sources of information including hyperlinks [188, 123], web logs [171], the structure of pages [8], the colours used on pages [132], and the URL [132]. Query Directed Clustering exploits the query and term relationships learned from global document analysis to aid clustering.

A wide variety of algorithms have been considered and applied to web clustering, with the most successful being Suffix Tree Clustering and Lingo. Web page clustering algorithms include

- Combining link analysis and contents [188, 86, 122, 197, 191, 123, 34, 180, 87]
- Density based [171]
- Divide and merge clustering — combining divisive and agglomerative hierarchical clustering approaches [33]
- Grouper [202]

- Hierarchical agglomerative [166, 118, 168, 77]
- Hierarchical clustering [64, 154, 72, 191, 36, 37, 34, 205, 87]
- Hierarchical divisive [166, 86]
- Hierarchical monothetic clustering [105]
- Hierarchical fuzzy c-means clustering [124]
- Highly connected subgraphs [79]
- Lingo: phrase discovery using modified version of SHOC and singular value decomposition [136, 137, 135, 44]
- k-Means [166, 155, 168, 190]
- Nearest Neighbour [198, 77]
- Mapuccino [15]
- Robust fuzzy k-medoids [174]
- Scatter Gather [15]
- SHOC [205, 15]
- Similarity based graph partitioning [168]
- Suffix Tree Clustering: phrase based single link hierarchical agglomerative method [203, 174, 164, 15, 28, 201]
- Supervised clustering [62]
- Unsupervised Bayes [168]
- Vivisimo [15]

5.2.6 Suffix Tree Clustering

Suffix Tree Clustering (STC) [203] is a four-stage algorithm. The first stage prepares the pages for clustering. This stage splits the pages into phrases using standard sentence terminators such as full stops or exclamation points, and the location of surrounding HTML tags. It then cleans and stems the pages in the standard manner described in section 2.4.3.

The second stage efficiently constructs base clusters using a suffix tree. Base clusters are sets of documents that contain at least one phrase in common. The algorithm constructs a suffix tree using the phrases in the documents (an example is shown in figure 5.2). Then it extracts the base clusters from the suffix tree. Each non-root node in the suffix tree corresponds to a phrase, and each base cluster corresponds to a node on the suffix tree. For example, node A in figure 1 is the base cluster that represents documents containing the common phrase "the dog" and this base cluster contains documents 1 and 2.

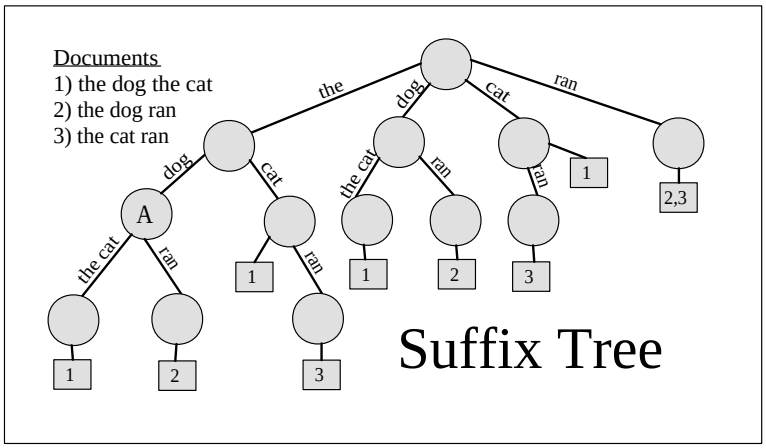

Figure 5.2: A suffix tree for three documents

The base cluster score combines the length of the common phrase and the number of documents in the base cluster. The score is defined as $s = |D| \cdot f(|P|)$ where $|D|$ is the number of documents in the base cluster, $|P|$

is the length of the common phrase, and $f(|P|)$ is defined as follows:

$$f(|P|) = \begin{cases} 0.5 & \text{if } |P| = 1 \\ |P| & \text{if } 1 < |P| \leq 5 \\ 6 & \text{if } |P| > 5 \end{cases}$$

The third stage combines base clusters to form the output clusters. STC combines the base clusters by clustering them using a single-link clustering algorithm that stops when the similarity between base clusters is less than a constant threshold (the similarity constant). STC defines the similarity between base clusters as follows:

$$similarity(b_1, b_2) = \frac{|b_1 \cap b_2|}{\max(|b_1|, |b_2|)}$$

where b_1 and b_2 are sets containing the documents in a base cluster. Each cluster found using this algorithm is a set of base clusters. For each set, STC constructs an output cluster that contains the documents in the union of the set's base clusters.

The fourth stage scores each output cluster and shows the highest scoring clusters to the user. The score of an output cluster is equal to the sum of its base cluster scores as shown in figure 5.3.

Researchers have made many improvements to STC. The suffix tree structure can be replaced by a suffix array [116], which is similar to a suffix tree and performs the same function, but has significantly lower memory requirements. SHOC (Semantic, Hierarchical, Online Clustering) [205] extends STC to handle oriental languages such as Chinese.

My earlier work on Extended Suffix Tree Clustering (ESTC) [44] significantly improved the clustering quality of STC and made the algorithm significantly more stable to variation of the similarity constant. STC scores clusters for some topics much higher than other clusters, even when the underlying cluster quality is identical. This is because some topics may have a disproportionately large number of phrases and therefore have a large number of overlapping base clusters as shown in figure 5.4. ESTC

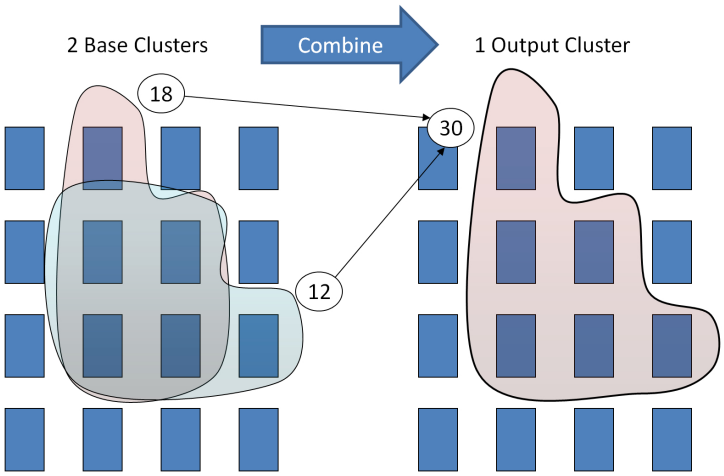

Figure 5.3: STC scores output clusters by combining the scores of base clusters

improved the scoring function of STC to eliminate the over-counting of overlapping base clusters. ESTC also improved the clustering coverage by using a particularly efficient hill climbing search with look-ahead to select clusters that cover as many documents as possible.

5.2.7 Google Distance

One way of improving web page clustering algorithms is to exploit the textual properties of web pages to measure document similarity and cluster coherence. The semantic relationships between words, for example, synonyms, hyponyms, meronyms, etc., is very useful information for such a measure [39]. WordNet [39] is a lexical reference system and is one source of such information. However, the data in WordNet is incomplete, particularly for commercial, technical, and popular culture word usage.

An alternate source, although less accurate and less informative, is to use global document analysis and term co-occurrence statistics to identify whether terms are related or unrelated. It is possible to compute approximate measures of term co-occurrence using the Web Frequency (the

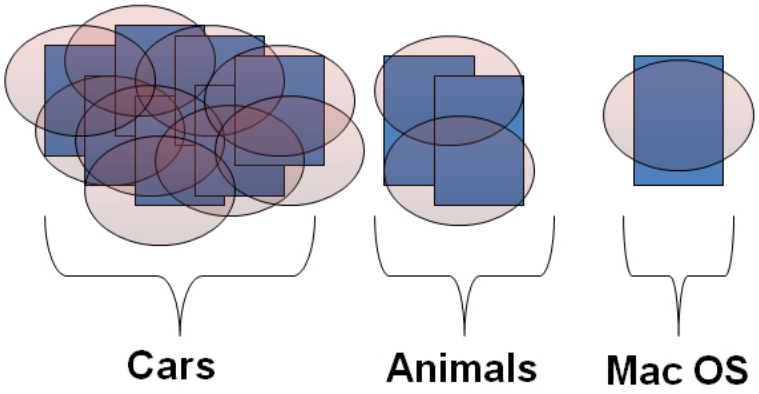

Figure 5.4: Some topics will have many overlapping base clusters because the distribution of phrases between topics is not uniform, but is dependent on the number of pages related to that topic in the result set

number of pages retrieved by a web search) of the terms under comparison and by a multi-term search for their conjunction. For example, computing the term co-occurrence between the terms Ford and car would involve the number of results for the queries "Ford", "car", and "Ford car". Google Distance [39] and the Rough Set based Graded Thesaurus [41] are two techniques that use these statistics to determine term similarity and both have been shown to be effective on various tasks, such as hierarchical word clustering [39] and query expansion [41].

QDC uses term relationships to provide a dramatic improvement in clustering performance. Specifically, QDC uses the original normalized Google distance (NGD) [39]:

$$NGD(i,j) = \frac{\max(\ln(f(i)), \ln(f(j))) - \ln(f(i \wedge j))}{\ln(M) - \max(\ln(f(i)), \ln(f(j)))}$$

where i and j are terms, $f(t)$ is the Web Frequency of some term or terms t, and M is the total number of pages. Note, after the development of QDC [46], researchers revised NGD [40] to use the minimum, rather than the maximum in the denominator:

$$NGD'(i,j) = \frac{\max(\ln(f(i)), \ln(f(j))) - \ln(f(i \wedge j))}{\ln(M) - \min(\ln(f(i)), \ln(f(j)))}$$

In practice, the original NGD provides a greater spread of values and works well with QDC. QDC's parameters could be re-tuned for the revised NGD, however, performance is unlikely to improve significantly. The remainder of this thesis exclusively uses the original NGD.

5.3 Analysis of Web Page Clustering Algorithms

This section decomposes the general model used for clustering web pages and analyzes the cause behind the failures of current approaches. The design of Query Directed Clustering uses this model and addresses each cause of failure.

5.3.1 General Model for Clustering Web Pages

The diversity of web page clustering algorithms is large, but all follow roughly the same model. My 7 stage model is shown in figure 5.5, and, like almost all algorithms, starts by pre-processing the pages before clustering. This involves removing stop words, HTML tags, punctuation, and applying stemming. The result of the pre-processing stage is an association between pages and terms, which, depending on the algorithm, might be words or phrases.

After pre-processing, clustering algorithms diverge somewhat, but largely follow the same process. For example, Suffix Tree Clustering [201], Link-based Clustering [190], and Lingo [137] map quite cleanly onto the model as shown in figures 5.6, 5.7, and 5.8 respectively. My new algorithm, Query Directed Clustering, follows this model almost exactly as shown in figure 5.9.

The first stage identifies the topics: an imprecise description of a superset of the final clusters. The description can be either concrete (an extension of a cluster: a set of documents) or abstract (an intension of a cluster: a set of terms describing the cluster). Standard clustering al-

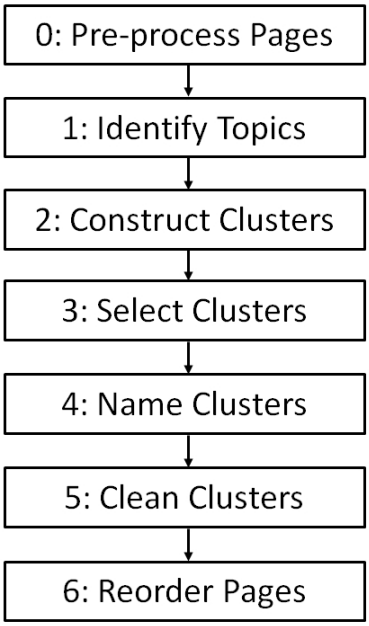

Figure 5.5: General model for web page clustering algorithms

gorithms typically skip this step and go straight to constructing the final clusters. However, many good web page clustering algorithms first restrict the set of possible clusters. Suffix Tree Clustering and Link-based Clustering achieve this by building base clusters, which are concrete descriptions, while Lingo chooses the final names of the clusters (usually stage four), which are abstract descriptions.

The second stage constructs possible clusters: finds the extensions for a superset of the final clusters. All clustering algorithms do this, although it might involve multiple sub-stages. For example, Query Directed Clustering first merges the base clusters and then splits them.

The third stage selects the set of output clusters to show the user. Standard clustering algorithms typically skip this stage and retain all the output clusters from stage two, as do other algorithms like Link-based Clustering.

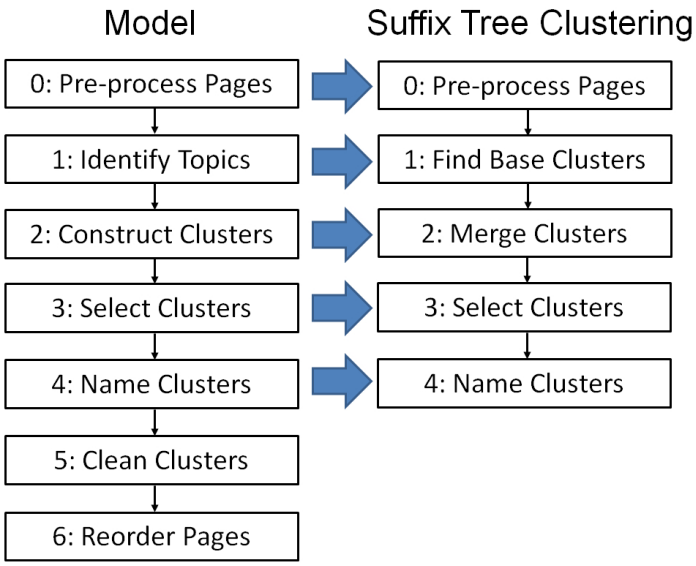

Figure 5.6: Suffix Tree Clustering fits the general model when some steps are left out

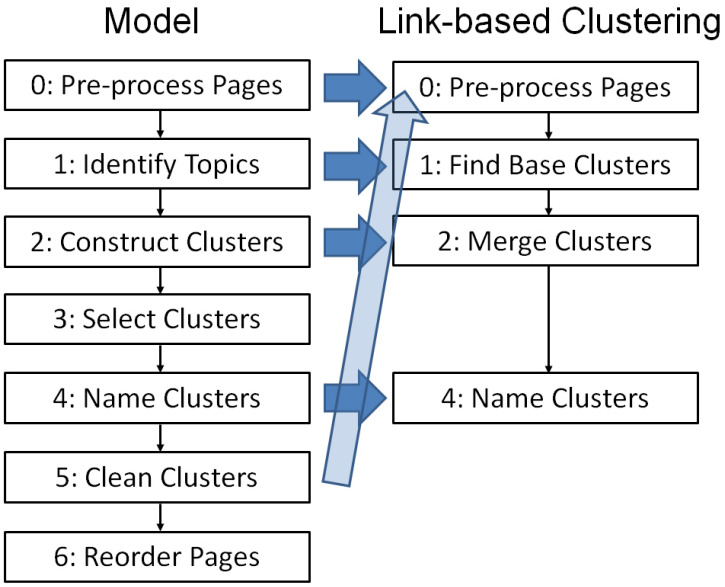

Figure 5.7: Link-based Clustering fits the general model when some steps are skipped

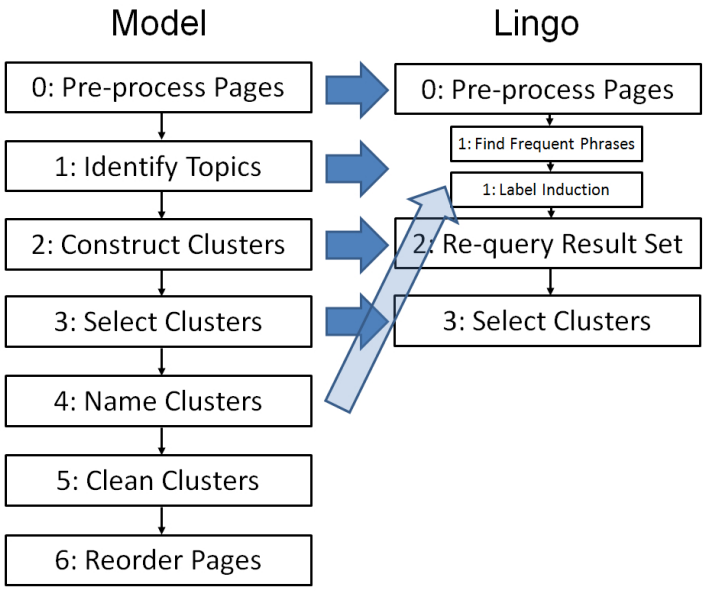

Figure 5.8: Lingo fits the general model when some steps are re-ordered

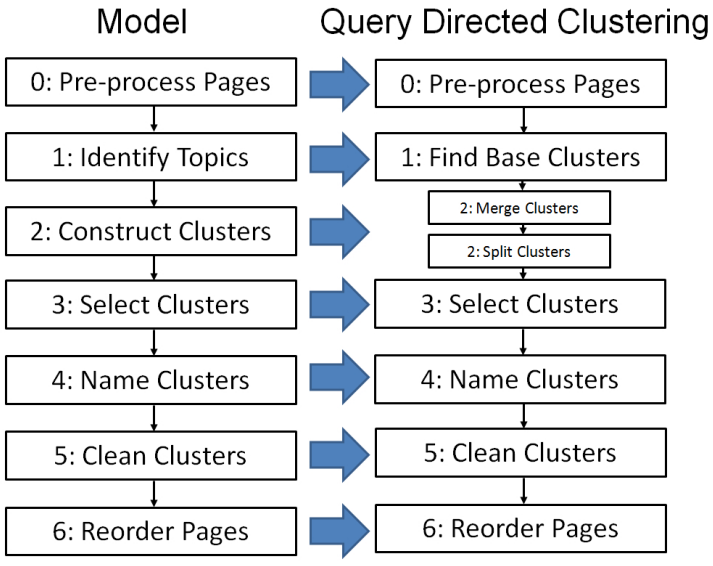

Figure 5.9: Query Directed Clustering fits the general model

The fourth stage names the output clusters. All web page clustering algorithms name the clusters to allow users to identify relevant clusters. Most algorithms use a simple naming convention, for example, STC uses the union of the phrases on the base clusters that form an output cluster. Other algorithms like Lingo use a complex naming scheme and may name the output clusters as part of an earlier stage, for example, Lingo names the output clusters as part of identifying the topics in the first stage.

The fifth stage involves cleaning the output clusters (removing irrelevant pages or outliers). Most algorithms skip this stage and instead assume all pages in the output cluster are relevant. Like naming, algorithms can clean clusters during an earlier stage. For example, Link-based Clustering identifies outliers during pre-processing. This has the benefit of excluding the outliers from the clustering process and therefore avoids biasing the constructed clusters. However, when performed later in the process, cleaning is more accurate, because the final clusters can be analyzed when determining relevancy and finding outliers. For this reason, Query Directed Clustering performs cleaning near the end of the process.

The sixth stage involves re-ordering the pages within each cluster, according to their relevancy to the cluster. Most algorithms skip this step and instead use the default behaviour: maintain the original ordering from the search results. Query Directed Clustering reuses its measure of cluster relevancy from the fifth stage to re-order the pages and improve the relevancy of the pages the user is most likely to view.

Deviations from the model are relatively minor: skipping stages (e.g. most direct implementations of standard clustering algorithms, Suffix Tree Clustering, Lingo, and Link-based Clustering), splitting stages(e.g. Lingo, Link-based Clustering, and Query Directed Clustering), and rearranging stages (e.g. Lingo and Link-based Clustering). Skipping occurs due to the use of default behaviour, for example, maintaining the search result order of pages within the clusters as opposed to reordering the pages. Splitting occurs when algorithms invoke multiple discrete steps within one stage.

Rearrangements typically involve shifting later stages to occur earlier and most commonly involve stages 4 and 5, naming clusters and cleaning clusters.

5.3.2 Problem 1: Low quality clusters

While Suffix Tree Clustering (STC), Lingo, and other algorithms find many good clusters, they also find many clusters that are ambiguous, overly specific, low value, or incomprehensible (semantically meaningless). For example, consider the following sample of clusters produced by two of the best clustering algorithms (ESTC and Lingo) for the query "Jaguar".

1. Mac OS Jaguar - Good Cluster
2. Cat the Jaguar - Good Cluster
3. Safety Information - Ambiguous (Car Safety, Mac OS Safety, Cat Safety)
4. Locate a Used Car - Overly Specific
5. Home Page - Low Value
6. Official Web - Low Value
7. Pictures - Low Value
8. Online - Incomprehensible
9. System - Incomprehensible

Low value, incomprehensible, and semantically meaningless clusters occur because from a textual perspective the algorithms use only local[5] document properties to form clusters. While local term frequencies are informative, in isolation they are an inadequate guide for the clustering process, because the semantic content of terms within a result set is not

[5]from pages within the result set

homogeneous. For example, while "Home Page", "Official Web", "Pictures", "Online", and "System" might occur frequently in the result set of "Jaguar", they also occur frequently on the web and therefore carry less semantic weight than other frequently occurring terms like "Mac OS" and "Used Car".

The problem arises as terms with low semantic weight form relationships between otherwise unrelated documents, which leads to the construction of ambiguous clusters. While the problem exists in many algorithms, including Lingo, the problem is most evident in algorithms like STC that construct output clusters from a union of base clusters. Since base clusters constructed from terms with low semantic weight are ambiguous, the output clusters containing them will ultimately be ambiguous too. The solution is to ignore terms with low semantic weight, because this avoids ambiguity inducing relationships from contaminating the clustering process. Query Directed Clustering achieves this by pruning such terms during the first stage of the clustering process.

Search engines use inverse document frequency to weight the semantic importance of different query terms. Even this simple global information could improve the clustering. For example, one extension improves STC [203] by incorporating inverse document frequency into the cluster score and Lingo uses inverse document frequency when identifying topics. However, global document analysis such as Google Distance and other co-occurrence based measures can produce even richer information that could further improve the clustering. Query Directed Clustering uses richer global information to improve several stages of the clustering process and to reduce significantly the number of low quality output clusters.

5.3.3 Problem 2: Low coverage clusters

Web Page Clustering algorithms face a tradeoff between quality and coverage. Constructing high quality clusters is relatively easy: most terms with

high semantic weight lead to pure non-ambiguous high-quality clusters.[6] However, such clusters typically have low coverage of the topic they represent. The solution is to combine the relationships inferred from multiple terms to form larger, high coverage clusters. Unfortunately, algorithms often combine unrelated terms to produce low quality clusters.

This problem exists in many algorithms, including Lingo, which hides term combination as part of Singular Value Decomposition, and is most evident in algorithms like STC, that explicitly merge base clusters. STC addresses low coverage clusters by merging base clusters using a single-link clustering algorithm [92] with cluster overlap as the similarity measure. The problem is that cluster overlap is a poor measure of semantic relatedness and it may merge semantically unrelated clusters, which lowers cluster quality, unless the overlap threshold is set very high. However, this leaves many related clusters separate, which limits cluster coverage.

Similar to the solution for low quality clusters, the solution for low coverage clusters is a better understanding of semantics and again the source of that understanding is global document analysis. With an improved measure of semantic relatedness, the merging threshold can simultaneously be lower for related clusters and higher for unrelated clusters, which improves both quality and coverage. QDC measures the semantic similarity of cluster intensions (cluster descriptions) in addition to using cluster overlap to provide a more effective similarity measure for merging clusters that boosts both cluster quality and coverage.

5.3.4 Problem 3: Poor output clusters

Algorithms must select some number of clusters to show the user. The number of clusters shown to the user should be at least the number of query interpretations (ideally exactly equal), and not too large for the user to comprehend (no more than 10). Many algorithms, like k-Means, always

[6] where the cluster contains exactly the pages that contain the term

construct a fixed number of clusters. Others, like STC, construct thousands of clusters (when using full text as opposed to snippets) and then select the best subset of fixed size.

Presenting a fixed number of clusters prevents the user being overwhelmed with too many clusters. However, it fails to simplify their decision process when the number of query interpretations is less than the number of clusters. At best, the additional clusters are semantically relevant (which is not usually the case) and represent sub-topics of the main query interpretations. For example, reconsidering the "Jaguar" query: "Car" represents a main query interpretation, while "Locate a Used Car" and "Jaguar Auto Parts" represent sub-topics of the broader "Car" topic. The problem with showing sub-topics is the increased probability of multiple relevant choices, which hinders the user's decision. Even worse, algorithms often select sub-topics in preference to those representing the main query interpretations.

One solution is a hierarchical display of clusters [68, 105]. However, hierarchical algorithms still face the issue of correctly constructing the hierarchy. Hierarchical algorithms can incorrectly promote sub-topics to top-level topics and miss entire top-level topics. One approach for generating a hierarchical clustering is the recursive application of the same clustering algorithm. If the algorithm finds clusters representing exactly the top-level topics, this will generate a good hierarchical clustering. Unfortunately, even when the original query was an easy ambiguous query, there is no guarantee that the sub-topics of a top-level cluster have distinct vocabulary and therefore further sub-clustering may fail.

QDC builds the top-level of a hierarchical clustering by iteratively selecting the highest quality cluster that improves coverage, ensuring the number of clusters matches the number of topics. QDC measures quality using the cluster's semantic distance from the query. The semantic distance is designed to be negative for sub-topics, which ensures only top-level topics are selected. If users need further refinement of a cluster's

results, it is treated as a new and separate refinement task, and solved using the most appropriate method (which might be QDC or one of the methods described later in this thesis).

5.3.5 Problem 4: Irrelevant page ordering within clusters

After selecting a cluster, the user sees a result set consisting of the cluster's pages. Providing the most relevant pages earlier in the results can reduce the time users spend searching [10]. Most clustering algorithms order the pages in the clusters by their position in the search results [32]. Such an ordering fails to use the additional information about the user's search goal, provided by the user selecting the cluster, so the most relevant pages may not be shown first. Query Directed Clustering orders the pages within each cluster according to their relevance to the cluster.

5.4 Algorithm - QDC

The next few sections describe Query Directed Clustering (QDC), a new web page clustering algorithm with five key innovations. The algorithm is explained sequentially in five parts, which cover the stages of the algorithm outlined in figure 5.9. The pre-processing in QDC is standard and is not described further, section 5.5 covers topic identification, together sections 5.6 and 5.7 cover the construction of clusters, section 5.8 covers the selection of clusters, and finally section 5.9 covers the naming, cleaning, and page reordering of clusters.

Base Cluster Identification (section 5.5) describes QDC's query guided quality measure and explains how it improves cluster quality by reducing cluster contamination. Cluster Merging (section 5.6) then explains how QDC uses the semantic relationship between cluster intensions to improve cluster coverage without impeding cluster quality. Cluster Splitting (section 5.7) then explains how QDC improves cluster quality by resolving the

single-link clustering problem of cluster chaining (cluster drift) using a hierarchical agglomerative clustering algorithm. Cluster Selection (section 5.8) then explains how QDC uses the relationship of terms to the query to gauge the specificity of clusters and thus select the highest quality clusters that both maximize coverage and represent the top-level clusters of a hierarchical clustering. Finally, Cluster Naming, Cleaning, and Reordering (section 5.9) explains how QDC improves clusters by ranking the pages by relevance to the cluster.

5.5 Base Cluster Identification

QDC identifies the search result topics by establishing a set of unambiguous base relationships between the pages. QDC represents this set of relationships by a set of base clusters, which it constructs after standard page pre-processing.

5.5.1 Initial Base Clusters

Initially, QDC finds a superset of the final base clusters. A base cluster (b) is described by a single word[7] ($D(b)$) and consists of all the pages containing that word. Equivalently, base clusters are single word search refinements based on the current search results. QDC constructs a collection of base clusters, one for every word that is in at least 4% of the pages.

In general, lowering the threshold will increase clustering performance at the cost of cpu time. However, if the smallest base clusters are too small, they introduce noise and reduce clustering performance. 4% provided good performance during my experiments, which clustered 200 pages; 200

[7]As shown by STC and Lingo, using phrases might improve performance, however QDC aims to establish the significance of exploiting the query and global document analysis independently of phrases.

pages is typical of recent web page clustering methods and proved sufficient[8] for covering the range of topics in the result set.

Many of the initial base clusters are useless and only serve to contaminate the final clusters as discussed in section 5.3.2. Removing these useless clusters would improve the clustering, but selecting the right clusters to prune requires some guide to cluster quality.

5.5.2 Estimating Cluster Quality using Query Distance

The user's query is the best, and often the only, specification of the information desired by the user. QDC uses the relationship between query terms and cluster descriptions as one part of a cluster quality guide. The Query Distance ($QD(b)$) of each base cluster is the minimum of the NGD[9] between the word specifying the base cluster and any query term.

$$QD(b) = \min_{term \in query} NGD(D(b), term)$$

Query Distance is a useful guide to cluster quality because it distinguishes terms according to their ability to resolve the ambiguity in the query. Words with a high query distance tend to be semantically unrelated to all the query terms, making them unlikely to be related to any query interpretation and therefore unlikely to contribute to the disambiguation of interpretations. Whereas words with a low query distance tend to be very specific, making them more likely to be related to a single query interpretation and therefore better at disambiguating the different query interpretations. Figure 5.10 shows the Query Distance for various terms relative to the query "Jaguar". As expected, terms that relate closely to a single query interpretation (Toyota, Ford, Seat, Tiger, Vehicle) have a small query distance, while terms that are semantically unrelated to all interpretations

[8] My experiments found that performance was relatively poor on 50 pages, acceptable on 100 pages, and good on 200 pages and above.

[9] Using Normalized Google Distance (NGD) as defined in section 5.2.7.

(Professional, Technology, Location, Buy, Homepage) have a large query distance.

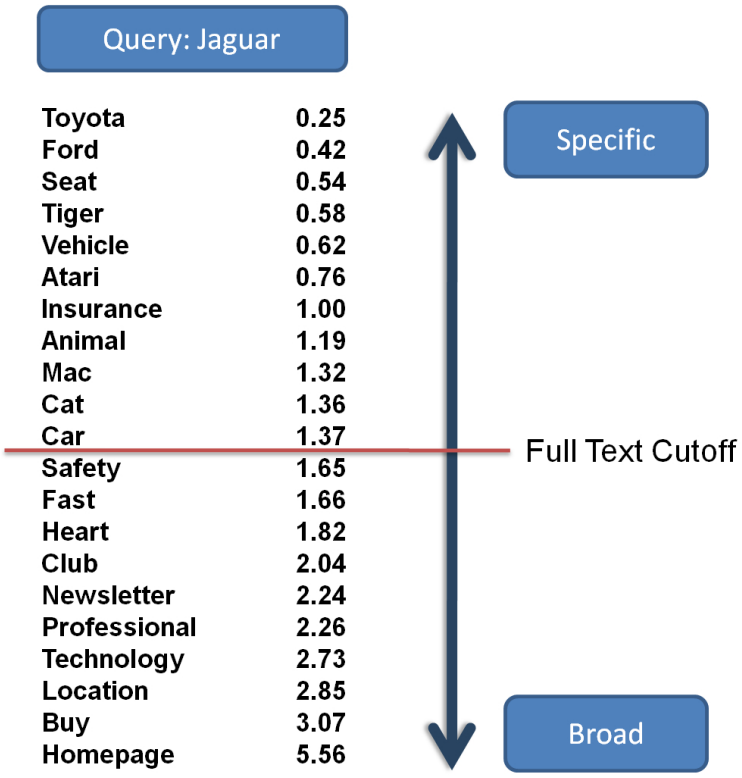

Figure 5.10: The query distance for various terms relative to the query "Jaguar"

5.5.3 Pruning Ambiguous Base Clusters

Ambiguous clusters are often of poor quality because they combine multiple distinct ideas of which only one is normally of interest to a given user. QDC removes these low quality clusters by removing clusters whose Query Distance is too large. The experiments use cutoffs of 1.5 when using full text data and 2.5 when using snippet data.[10] This removes most low

[10]The higher cutoff is necessary because snippets contain dramatically less content than full text.

quality clusters, but if the cutoff is too low (for example, 1.0), high quality clusters may be removed as well (for example, Mac, Cat, and Car); using a higher cutoff removes fewer clusters.

After pruning using Query Distance, there are still many low quality clusters (for example, Symbol, Weight, and Distribution). The dilemma is that low quality clusters persist because the cutoff is too high, but lowering the cutoff would remove high quality clusters. The solution is to combine local information (document frequency) with the global information (Query Distance) used already.

The distribution of web pages tends to follow the frequency of user interest in the page topics. Therefore, larger clusters have a greater probability of being useful refinements and cluster size is an indication of cluster quality. QDC removes the worst clusters according to a measure ($baseQuality(b)$) proportional to cluster size[11] and inversely proportional to Query Distance.

$$baseQuality(b) = \frac{|b|}{QD(b)}$$

This measure of base cluster quality is very effective as shown in figure 5.11.[12]

The number of clusters kept is proportional (1:1) to the total number of pages being clustered (200 during this chapter's experiments). Keeping fewer clusters will increase algorithm speed but lower clustering performance; however, if too many clusters are kept, low quality clusters are not pruned and may contaminate the merging process.

Pruning reduces the number of clusters by an order of magnitude (10 – 25 fold on full text data), which dramatically reduces the cpu time cost relative to other algorithms that generate base clusters like Suffix Tree Clus-

[11] the document frequency of the cluster's description amongst the pages to be clustered

[12] Figures 5.10 and 5.11 show different subsets of the full set of base clusters. Specifically, figure 5.11 includes Symbol, Weight, and Distribution even though these were not shown in figure 5.10.

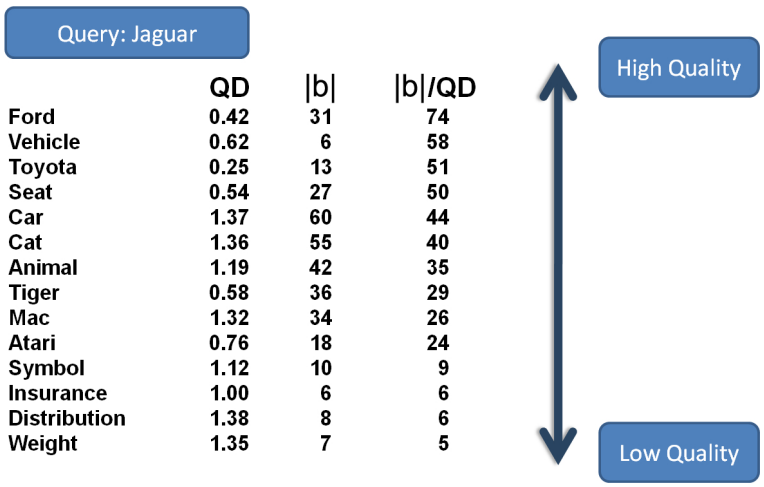

Figure 5.11: Base Cluster quality is estimated using a combination of Query Distance (QD) and cluster size ($|b|$)

tering. Removing this many clusters would normally have a negative effect on clustering performance, but because the query directed heuristics give a reliable guide to cluster quality, the low quality clusters that would later contaminate the merging process are removed, and the performance actually improves.

5.6 Cluster Merging

QDC constructs larger clusters with improved coverage by merging clusters together using a single-link clustering algorithm. Each cluster (c) is constructed from a set of base clusters ($base(c)$), and the description of a cluster ($D(c)$) is the term that describes the cluster's largest base cluster.

5.6.1 Base Cluster Similarity

Previous methods often merge unrelated clusters, which decreases cluster quality by introducing irrelevant pages as discussed in section 5.3.3. QDC

addresses this by using an improved measure of base cluster similarity that considers both the cluster's contents (page members) and descriptions.

The problem is normally exacerbated when using single-link clustering by cluster chaining (cluster drift): clusters that are closely related to one of the unrelated clusters but not the others are often merged in too, bringing further irrelevant pages with them. While the improved similarity measure helps resolve cluster chaining, it does not solve the problem completely; for that, QDC splits clusters as described in section 5.7.

QDC defines two clusters to be sufficiently similar only if both the cluster contents and cluster descriptions are sufficiently similar. Requiring the cluster descriptions to match in addition to the contents dramatically reduces the merging of semantically unrelated clusters and increases cluster quality. Additionally, by using two independent conditions, the thresholds for both can be significantly reduced, which allows more semantically related clusters to merge (increasing cluster coverage), while not hurting cluster quality. The thresholds are independent because one is based on local information (cluster contents) and the other is based on global information (cluster descriptions).

The cluster contents are sufficiently similar if enough of the pages in one cluster are also in the other cluster (*i.e.*, if there is enough overlap between the clusters):

$$\frac{|c_1 \cap c_2|}{\min(|c_1|, |c_2|)} > \gamma$$

The cluster descriptions are sufficiently similar if the pair of cluster descriptions occur together on the web significantly more frequently than would be expected if the pair were unrelated (*i.e.*, if their appearances were

independent):

$$\begin{aligned} & P(d_1 \wedge d_2) && > \delta P(d_1)P(d_2) \\ \equiv\ & \tfrac{f(d_1 \wedge d_2)}{M} && > \delta \tfrac{f(d_1)}{M} \tfrac{f(d_2)}{M} \\ \equiv\ & \tfrac{M f(d_1 \wedge d_2)}{f(d_1) f(d_2)} && > \delta \end{aligned}$$

where d_1 and d_2 are the cluster descriptions, and $f(t)$ and M are as per NGD in section 5.2.7.

My experiments use a cluster content threshold of $\gamma = 0.6$ and a cluster description threshold of $\delta = 4$, which means that only clusters that share at least 60% of their documents and whose descriptive terms co-occur on at least four times more web pages than they would if they were independent. Decreasing either the cluster content or the cluster description thresholds will increase cluster coverage at the cost of greater cluster overlap.

5.6.2 Merging Clusters with Single-link Clustering

Initially there is a singleton cluster for each base cluster. QDC merges clusters using single-link clustering over a relatedness graph. Single-link clustering merges all clusters that are part of the same connected component on the graph. The relatedness graph has the clusters as vertices and has an edge between any two clusters whose content similarity and description similarity exceed the relevant thresholds.

While the set of base clusters defines the cluster, the base clusters may contain outliers: pages that contain the base cluster's term, but which do not reflect the topic of the merged cluster. QDC removes outliers, by selectively including pages from the base clusters in the final cluster. A cluster only includes a page if it is present in enough of the cluster's base clusters. The more base clusters containing a page, the greater the evidence that the page is not an outlier. However, the increase in evidence with each additional base cluster decreases. Therefore, the threshold should be a function

with a monotonically decreasing first derivative; QDC uses the logarithm. The cluster's pages comprise pages that are in at least $\log_2(|base(c)|+1)$ of the cluster's base clusters. As shown in figure 5.12, this means pages must be in at least 2 base clusters when the cluster has 3 base clusters, 4 when the cluster has 15, 6 when the cluster has 63, etc.

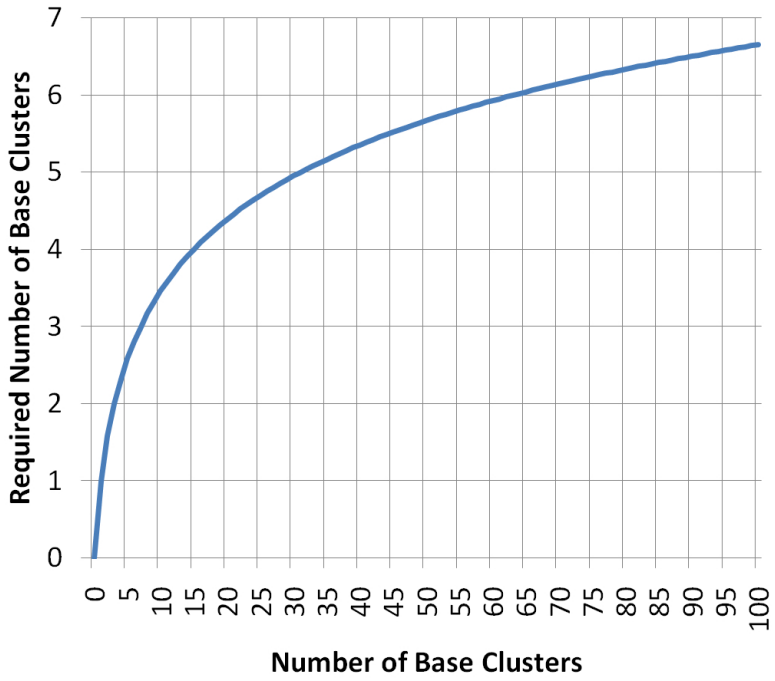

Figure 5.12: The threshold for cluster page membership depends on the number of base clusters

5.7 Cluster Splitting

QDC further improves the construction of clusters by splitting ambiguous clusters that cover multiple topics into their constituent parts using a hierarchical agglomerative algorithm.

5.7.1 Ambiguity of Maximally Merged Clusters

After merging, the clusters are maximal, in that each cluster now contains at least all the base clusters that relate to one idea; single-link clustering assures this because it merges all related clusters. However, single-link clustering, even with the improved similarity function, can produce unfocused/ambiguous clusters containing multiple topics and irrelevant base clusters due to cluster chaining (cluster drift). Such clusters need to be split.

Ideally, the merging algorithm would prevent the formation of these clusters; however, these restrictions often prevent desirable merging as well. Interestingly, it is easier to split compound clusters than to prevent their formation in the first place because the splitting can take into account the final cluster, whereas the merging process cannot. Researchers [166] observed a similar phenomenon where k-Means' non-committal strategy could compensate for early mistakes caused by an unreliable measure of nearest neighbours better than hierarchical agglomerative clustering, because k-Means could observe the overall clustering.

5.7.2 Finding Sub-clusters using Connectivity

During the merging process, the single-link clustering algorithm constructs a graph of the relationships between each cluster's constituent base clusters. The splitting algorithm can exploit this graph to identify groups of base clusters (sub-graphs) that probably represent different topics.

The relatedness graph connects base clusters that are semantically similar and their structure can vary dramatically. However, all relatedness graphs share the primitive sub-structures shown in figure 5.13: densely connected regions, sparsely connected regions, and long chains. These common sub-structures simplify the analysis.

The assumption underlying the merging of base clusters is that semantically similar terms represent the same topic. While generally this is a

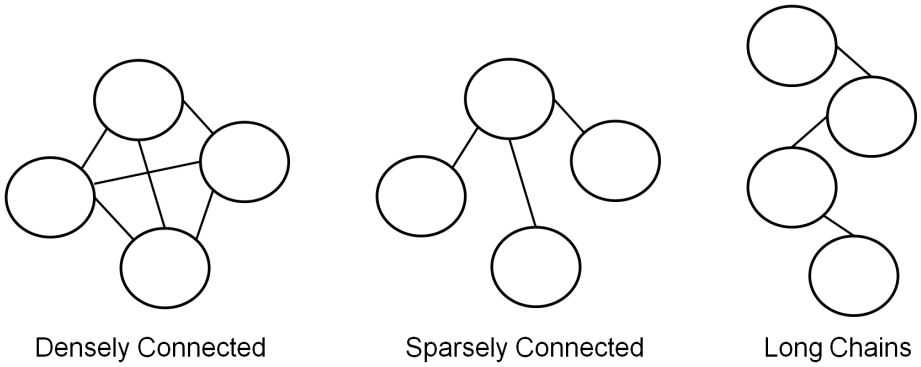

Figure 5.13: Single-link clustering creates a broad spectrum of merged clusters consisting three kinds of primitive sub-structure

reasonable assumption, it can be wrong, because individual terms can be polysemous (have multiple meanings). Additionally, the measure of semantic similarity may make mistakes. Identifying these problems during merging is difficult. However, since mostly the assumption is valid, when problematic relationships occur, they appear as anomalies in the relatedness graph.

Densely connected regions are most likely to represent a single topic, because they occur when many base clusters are mutually semantically similar. Since problematic relationships are rare, it is unlikely that all those relationships were wrong.

The most obvious anomaly is the long chain where a single string of clusters is the only connection between two groups of clusters. Long chains occur when a series of clusters are semantically similar to their neighbours, but not to clusters further away in the chain. While it is reasonable to assume two directly connected clusters represent the same topic, this assumption grows less reasonable as the number of intermediaries grows, because it requires that all intermediaries have failed to meet the relatedness criteria during merging (which is unlikely if they are truly related), otherwise there would be a shorter path between them. Therefore, the

probability of relatedness between groups of clusters connected by long chains is inversely proportional to the length of the chain.

Between these two extremes lie the sparsely connected regions, which can contain anomalies that are difficult to identify. For example, when two densely connected regions are sparsely connected to each other it is most likely they represent different topics, since the few connections between them are more likely to be anomalous than is the absence of the many expected connections if they represent the same topic. In contrast to long chains that are readily identified by a series of vertices of low degree, anomalous connections in sparsely connected regions are difficult to identify because the relevant edges and vertices are indistinguishable at the micro-level (individual edges and vertices). Identifying anomalous connections in sparsely connected regions requires analysis at the macro-level (groups of edges and vertices), but enumerating all groups is computationally impractical because the cost is exponential in the number of vertices. QDC uses a greedy approach with a good heuristic, so that the anomalous connections are identified, but only the relevant groups are considered.

In general, densely connected regions are more likely to represent a single topic than sparsely connected regions, and regions connected by long chains are the least likely to represent a single topic. This makes the path length and the number of independent paths between regions important in measuring how likely they are to represent the same topic. Given two regions with an equal number of clusters, the denser region will typically have both a shorter path length between clusters and a greater number of independent paths. Two regions connected by a long chain will have both a long path length and only one path. These properties are the basis of the similarity method QDC uses to split clusters.

5.7.3 Splitting Clusters with Hierarchical Agglomerative Clustering

QDC uses a hierarchical agglomerative clustering algorithm to identify the sub-cluster structure within each cluster. The algorithm uses a similarity measure based on the relatedness graph to build a dendrogram for each cluster starting from the base clusters in the cluster. QDC splits each cluster by cutting its dendrogram at an appropriate point — when the similarity between the closest pair of sub-clusters falls below a threshold (the experiments use -2).

A good similarity measure should ensure sub-graphs that represent the same topic have high similarity and vice-versa. Namely, the similarity measure should weight strongly the number of short paths between sub-graphs and discount regions separated by long chains and those that are sparsely connected.

QDC uses a similarity measure with three components: the number of length one paths between the two sub-clusters in the relatedness graph ($onelinks$), the number of length two paths ($twolinks$), and the average distance from base clusters in one sub-cluster to base clusters in the other sub-cluster ($avgdist$).

$$sim(c_1, c_2) = onelinks(c_1, c_2) + 0.5\, twolinks(c_1, c_2) - avgdist(c_1, c_2)$$

$$avgdist(c_1, c_2) = \frac{\sum_{b_1 \in base(c_1)} \sum_{b_2 \in base(c_2)} len(b_1, b_2)}{|base(c_1)||base(c_2)|}$$

where $len(b_1, b_2)$ is the length of the shortest path between two base clusters in the relatedness graph.

The first two components ($onelinks$ and $twolinks$) capture the importance of short paths and their relative importance is incorporated by discounting $twolinks$. While larger paths could be similarly incorporated, their significance diminishes fast — by analogy, consider a friendship network, while you may know the friends of your friends ($twolinks$), it is very unlikely that you know any of their friends ($threelinks$). However, it

is very important to incorporate at least $twolinks$, because it captures the notion of connectivity. This enables densely connected regions to be distinguished from sparsely connected regions, even at the lowest levels of the dendrogram when computing the similarity between individual base clusters.

The third component ($avgdist$) discounts sparsely connected regions and those separated by long chains. One interpretation is that $avgdist$ corresponds to the distance between the virtual[13] centers of each cluster.

The final part of the algorithm is the splitting threshold. The threshold of -2 means that any groups of base clusters that are not tightly interconnected with each other will be split. Using a higher threshold will lower the split point and increase the splitting frequency. Generally, the similarity between two sub-clusters will exceed a threshold of $-k$ when the distance between their virtual centers is less than $2k$. In the specific case that there is an articulation node (whose removal would disconnect the two sub-clusters), the distance between the virtual centers of two sub-clusters c_1 and c_2 is $ampl(c_1) + ampl(c_2)$, where $ampl(c)$ is the average of the minimum path lengths from the nodes in c to the articulation node. In general, the distance depends on the location of the connections between the sub-clusters and the internal structure of the sub-clusters.

5.7.4 Worked Examples

This section applies the splitting algorithm to two different clusters in figures 5.14 (cluster 1) and 5.16 (cluster 2). When the algorithm is applied, ties are broken by merging the two clusters that contain the lowest numbered clusters. The resulting dendrograms and application of the threshold cutting is shown in figures 5.15 and 5.17 respectively. The merge order and maximum similarity at each step are shown in tables 5.2 and 5.3 respectively. Cluster 1 shows that the algorithm correctly splits clusters

[13] virtual because it may not correspond to a specific base cluster

Table 5.2: Merge order and maximum similarity for cluster 1

Clusters Merged		Similarity
1	4	1
3	5	1
8	10	1
12	14	1
1,4	6	0.5
12,14	13	0.5
15	16	0.5
2	3,5	0
8,10	9	0
8,9,10	11	-0.333
12,13,14	15,16	-0.333
1,4,6	2,3,5	-0.5
7	8,9,10,11	-0.7
7,8,9,10,11	12,13,14,15,16	-2.2
1,2,3,4,5,6	7,8,9,10,11,12,13,14,15,16	-3.866

separated by isthmi and cluster 2 shows that the algorithm correctly splits densely connected clusters that are sparsely connected.

5.8 Cluster Selection

At this stage, QDC has a small set of coherent clusters. However, there will still be more clusters than can be presented to the user. QDC needs to select the best subset of the clusters to present to the user. Ideally, these clusters should be high quality clusters that represent exactly the top-level clusters of a hierarchical clustering as discussed in section 5.3.4.

Section 5.2.6 discussed ESTC, a clustering algorithm developed dur-

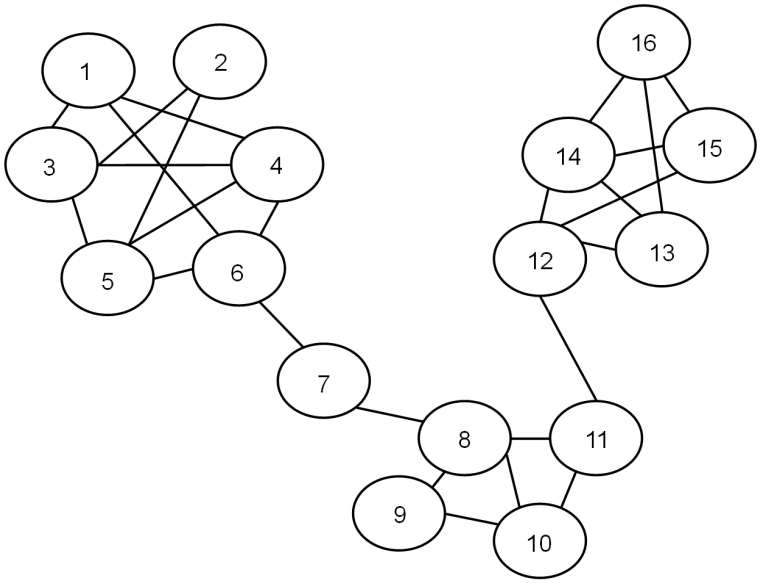

Figure 5.14: Merged cluster 1 containing three sub-clusters separated by isthmi

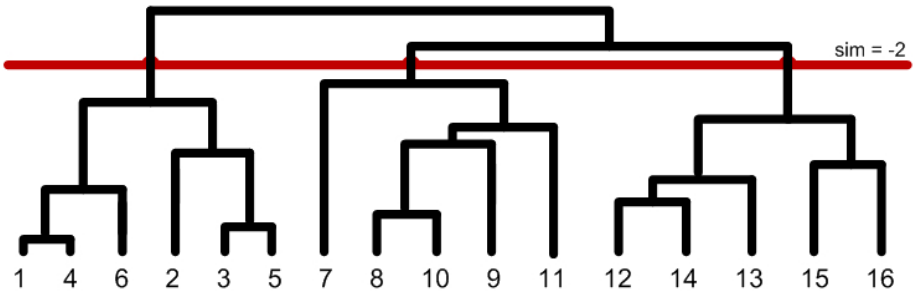

Figure 5.15: Dendrogram showing process of splitting cluster 1 into three sub-clusters

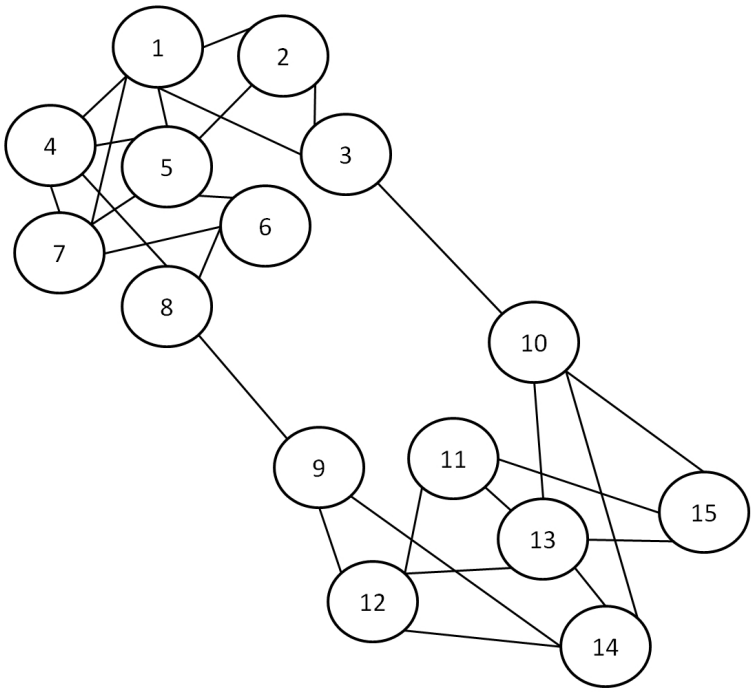

Figure 5.16: Merged cluster 2 containing two multiply connected sub-clusters

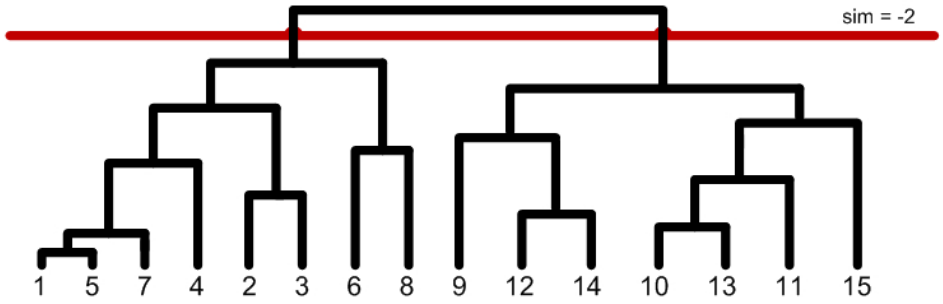

Figure 5.17: Dendrogram showing process of splitting cluster 2 into two sub-clusters

Table 5.3: Merge order and maximum similarity for cluster 2

Clusters Merged		Similarity
1	5	1.5
1,5	7	1
10	13	1
12	14	1
2	3	0.5
10,13	11	0.5
1,5,7	4	0
6	8	0.5
9	12,14	0
10,11,13	15	0
1,4,5,7	2,3	-0.6
9,12,14	10,11,13,15	-0.7
1,2,3,4,5,7	6,8	-0.5
1,2,3,4,5,6,7,8	9,10,11,12,13,14,15	-2.2

ing my earlier work [44]. One component of ESTC is an efficient search method that incorporates look-ahead to select a near optimal set of clusters efficiently, given a heuristic that evaluates a set of clusters (a clustering). QDC uses ESTC's search method with an improved heuristic ($H(C)$) to select a set of clusters to show the user. The remainder of this section describes QDC's heuristic.

5.8.1 Selecting High Quality Clusters using Query Distance

Section 5.5 estimated base cluster quality ($baseQuality(b)$) using two factors: the size of the cluster ($|b|$) and Query Distance ($QD(b)$). For base clusters, cluster size was useful because it predicted the probability of user interest and Query Distance was useful because it predicted the ambiguity of terms. For the same reasons, these factors are also useful for estimating cluster quality. However, there is a subtle difference between the desired properties of base cluster quality and cluster quality.

Query Distance distinguishes terms according to their ability to resolve the ambiguity in the query as discussed earlier in section 5.5.2. While ideal for determining the pure quality of a cluster, it acts in opposition to selecting only top-level clusters: terms with the lowest query distance are typically the most specific terms and therefore generally reflect sub-topics rather than top-level topics, which are typically represented by broader terms.

Cluster size partially compensates for the sub-topic preference of Query Distance, and ranks high quality top-level topics level with high quality sub-topics as shown earlier in figure 5.11 because top-level topics tend to be larger. To find the broader top-level clusters, QDC uses an additional size factor to give additional weight to the broader top-level clusters.

QDC uses three factors to estimate cluster quality ($quality(c)$):

1. Cluster Size = $|c|$
2. Average Base Cluster Query Distance = $QD(c)$

3. Number of Base Clusters = $\log_2(|base(c)| + 1)$

$$quality(c) = \log_2(|base(c)| + 1)\frac{|c|}{QD(c)}$$

The first factor, cluster size, mirrors the definition for base cluster quality. The second factor, Query Distance, was modified to handle the composite nature of clusters by finding the average Query Distance of its base clusters.

$$QD(c) = \frac{\sum_{b \in base(c)} QD(b)}{|base(c)|}$$

The third factor is new[14] and exploits the information gained from the composite nature of clusters by using the number of base clusters.

The more base clusters in a cluster, the greater the evidence that the cluster represents a semantically meaningful group of pages. However, the increase in evidence with each additional base cluster decreases. Therefore, the requirement is a function with a monotonically decreasing first derivative; QDC uses the logarithm.

The addition of the third factor not only improves accuracy, it also prefers top-level topics to sub-topics because top-level topics are broader and therefore generally contain more base clusters. Table 5.4 confirms this by calculating the quality for clusters constructed for the query "Jaguar". The top-level topics "Animal" and "Car" have dramatically higher quality than the clusters for their respective sub-topics "Habitat", "Hunting", "Mustang", and "Automatic", while all clusters have higher quality than the ambiguous[15] "Trademark" cluster. Additionally, table 5.4 shows that while the combination of Query Distance and cluster size do not distinguish clusters representing top-level topics and sub-topics, the third factor clearly separates the top-level topics from their respective sub-topics.

[14]The third factor is newly applied to measuring cluster quality; the same formula was also used in section 5.6 to eliminate outliers. This occurs because both assess the value of information from the same variable.

[15]Notably, by the cluster selection stage of QDC, very few low quality or ambiguous clusters remain at all.

Table 5.4: Cluster quality separates top-level topics (*) and sub-topics for the "Jaguar" query

Cluster	$QD(c)$	$\|c\|$	$\frac{\|c\|}{QD(c)}$	$\|base(c)\|$	$quality(c)$
Animal*	1.20	48	40	16	163
Habitat	0.67	15	22	2	35
Hunting	0.75	33	44	3	88
Car*	1.24	109	88	107	594
Mustang	0.45	10	22	1	22
Automatic	0.89	19	21	1	21
Atari*	0.66	18	27	1	27
Trademark	1.40	10	7	2	11

5.8.2 Selecting Top-level Clusters using Coverage

A good clustering should not only consist of high quality clusters, but should also have high coverage — it should contain clusters representing each of the topics. Assuming the clusters all have high quality, a surrogate for coverage is to measure the fraction of pages from the result set that are covered by at least one cluster in the clustering. When most of the pages are covered, it is reasonable to believe that most topics are also covered.

While quality distinguishes good clusters from bad clusters (ambiguous / semantically meaningless) and top-level topics from their respective sub-topics, it does not necessarily distinguish top-level topics from the sub-topics of other top-level topics. For example, table 5.4 shows that the cluster for top-level topic "Atari" has a lower quality score than the cluster for the "Animal" sub-topic "Hunting". Fortunately, the coverage surrogate addresses this case, as the "Atari" topic will be included because it improves coverage, but the "Hunting" sub-topic will be excluded because it will fail to increase coverage after the higher quality "Animal" cluster is added.

There is yet another case to consider; the "Hunting" cluster may contain pages that are not in the parent "Animal" cluster. This suggests that overlap is an important factor for the coverage surrogate. Using only the fraction of covered pages would suggest that the "Hunting" cluster should be added, but by considering overlap as well, it is easy to see that it adds little value.

QDC evaluates a candidate set of clusters (C) using a new heuristic ($H(C)$) that combines three factors:

1. Cluster Quality ($q(c)$)
2. Page Coverage (C_O)
3. Overlap ($C_P - C_D$)

$$H(C) = \left(\sum_{c \in C} q(c) \right) - \alpha C_O - \beta (C_P - C_D)$$

where C_O is the number of pages not covered by any of the clusters, C_P is the number of pages covered by the clusters, C_D is the number of distinct pages covered by the clusters, and $q(c)$ is a measure of cluster quality based on $quality(c)$.

The new heuristic has two parameters that enable control of clustering characteristics: α (the experiments use 0.2) and β (the experiments use 0.3). α controls coverage and increasing α will generate clusterings with greater coverage at the cost of cluster quality. β controls overlap and increasing β will lead to clusterings with fewer pages in multiple clusters at the cost of page coverage. In concert, α and β also control the relative importance of cluster quality and thus, as shown in the next section, influence the number of clusters shown to the user.

5.8.3 Number of Clusters

Ideally, regardless of any parameters, the user will inevitably see the highest quality top-level clusters. The parameters should instead control the

relative number of boundary cases shown to the user. Given the heuristic ($H(C)$) for clustering quality, the clusters themselves principally determine the number of clusters shown to the user through their quality ($q(c)$). The challenge is to ensure $q(c)$ entails these conditions.

If $q(c)$ were set equal to $quality(c)$, $H(C)$ dictates that clusters be added until no cluster has quality exceeding its overlap with existing clusters, subject to the number of new pages covered by the cluster. Usually $quality(c) > |c|$, so the quality will usually exceed the overlap, which means that almost every cluster is added — not a desirable outcome. Clearly, $q(c)$ must be substantially less than $quality(c)$.

QDC defines $q(c)$ as the logarithm of the ratio between $quality(c)$ and the mean cluster quality of all clusters.

$$q(c) = \log_2(\frac{quality(c)}{\text{mean cluster quality}})$$

High quality clusters have above average quality and therefore, because of the logarithm, positive quality values, whereas low quality clusters have below average quality and therefore negative quality values. Using this ratio also serves to normalize cluster quality and ensures consistency across different clusterings.

As $quality(c)$ assigns significantly higher scores to the top-level clusters, the mean cluster quality is significantly higher than the median and therefore, very few clusters have above average quality. This ensures that QDC almost inevitably selects the highest quality top-level clusters (as they are the only ones with positive quality values) and only selects lower quality clusters when they contribute enough coverage to overcome their negative quality. In concert, α and β control the exact amount they must contribute, because together they control the relative importance of cluster quality to page coverage and overlap.

5.9 Cluster Naming, Cleaning, and Reordering

Having selected the clusters to show the user, clustering is complete. However, the clusters need naming and the clusters can be cleaned and reordered to improve their usefulness to the user as described in section 5.3.5. QDC defines a measure of cluster-page relevance that it uses to clean clusters and to reorder the pages in the clusters.

5.9.1 Cluster Naming

QDC names a cluster using the term that describes the cluster's largest base cluster.

$$name(c) = D(\underset{b \in base(c)}{\operatorname{argmax}} (|base(b)|))$$

Each base cluster term usually represents either a good name for the topic or for one of its sub-topics. Since the largest base cluster is typically described by the broadest term, which is most likely to reflect the topic, rather than a sub-topic, this is a good choice for the name of the cluster.

5.9.2 Cluster Cleaning

Even after all the previous steps and attempts to remove outliers such as those in section 5.6, the clusters still contain some outliers.

Since the clusters should relate to only one topic, pages that are in multiple clusters might be irrelevant in some of the clusters. QDC computes the relevance of each page in each cluster and removes irrelevant pages from clusters where two requirements are met: the page has relevance below a threshold (the experiments use 0.1) and the page has higher relevance in another cluster. A higher threshold will remove additional irrelevant pages but will also remove relevant pages, but the threshold is not very sensitive because the second requirement limits the pages that can be removed.

Cluster-page relevance can be viewed as the probability that a page is a member of the cluster. It varies between 0 and 1, with 0 being a page that is completely irrelevant to the cluster. QDC defines the relevance of a page (p) to a cluster (c) based on two factors:

1. the number of base clusters of which the page is a member
2. the size of the base clusters of which the page is a member

Page relevance is computed as the sum of the sizes of the cluster's base clusters of which it is a member, divided by the sum of the sizes of all of the cluster's base clusters.

$$relevance(p, c) = \frac{\sum_{\{b|b \in base(c) \wedge p \in b\}} |b|}{\sum_{b \in base(c)} |b|}$$

5.9.3 Cluster Reordering

As discussed in section 5.3.5, it is preferable to show the most relevant results earlier. While most algorithms use the original search result order, QDC improves on this by ranking and displaying the pages in each cluster according to their relevance to the cluster. This improves cluster quality from the user's perspective because any remaining irrelevant pages are frequently near the bottom of clusters and so users rarely see them. Because of this, in practice, it may be preferable to skip the step of cleaning clusters and use only reordering.

5.10 Evaluation

The aim of Query Directed Clustering (QDC) was to improve clustering performance for easy ambiguous queries, without being less efficient than existing approaches. This section evaluates QDC both quantitatively and qualitatively on these objectives and finds that QDC significantly improves clustering performance for easy ambiguous queries, while being substantially more efficient than existing approaches.

5.10.1 Algorithm Efficiency

Suffix Tree Clustering (STC) is considered to be very efficient [201]. Even so, QDC is on the order of ten times faster than STC and on the order of one hundred times faster than K-means. This speed-up is achieved even though the later stages of QDC are substantially more complex than STC, because QDC significantly reduces execution time by pruning many base clusters during base cluster construction using the new query directed cluster quality measure.

5.10.2 Quantitative Evaluation of Clustering Performance

The evaluation used 13 measurements to compare the clustering performance of QDC against four other web page clustering algorithms (STC, ESTC, K-means, and Random Clustering) on eight data sets: search results of four different queries ("salsa", "jaguar", "gp", and "victoria university") using both full-page and snippet data. The queries are of varying clustering difficulty. The first three queries are easy ambiguous queries, while the fourth ("victoria university") is a hard ambiguous query. The simplest is "salsa", which has two distinct clusters (both large) and few outliers. "jaguar" is more challenging with five distinct clusters (one large, three medium, and one small) and some outliers. "gp" is harder with five distinct clusters (two large, and three small) and many outliers. "victoria university" is the hardest with five very similar clusters (two large, one medium, and two small) and few outliers.

I compared the algorithms under an external evaluation methodology using a gold standard method [167, 45] with a rich ideal clustering structure and QC4 measurements (quality and coverage) as discussed in chapter 4, because this is well suited for web page clustering evaluation. In addition to QC4 measurements, the standard[16] Precision, Recall, F-measure,

[16]Specifically, those defined in section 4.3 with Recall corresponding to Total Recall and F-measure corresponding to Cluster F-measure.

Entropy, and Mutual Information measurements [167, 45] provide further evidence for the results. All measurements come in both average and cluster-size weighted varieties (except mutual information for which averaging is not applicable), providing 13 measurements in total. Average measurements treat all clusters as equally important, while weighted measurements treat larger clusters as more important. Note that there is a tradeoff between different measurement pairs: quality *vs* coverage, precision *vs* recall, and entropy *vs* recall. F-measure and Mutual information provide single measures that combine both aspects. For all measurements, higher is better, except entropy, for which lower is better.

On average QDC performs substantially better than the other algorithms. Figures 5.18, 5.19, 5.20, and 5.21 show that for full text data, on average, QDC outperforms all of the other algorithms on all measurements by convincing margins. Figure 5.22 shows the overall percentage improvement each algorithm makes over the random clustering using the combined QC4 measure $\frac{1}{2}(H(AQ, AC) + H(WQ, WC))$, where H is the Harmonic Mean, AQ is Average Quality, AC is Average Coverage, WQ is Weighted Quality, and WC is Weighted Coverage.

On the full text data, QDC performed significantly better than the Random Clustering ($p < 0.05$) on all measurements. It performed significantly better than K-means ($p < 0.05$) on all measurements, except Average Coverage and Average Precision, where it was almost significantly better ($p < 0.10$). It performed significantly better than STC ($p < 0.05$) on Average and Weighted Quality, and Average and Weighted F-measure, and was almost significantly better than STC ($p < 0.10$) on Average and Weighted Coverage, Weighted Precision, Average Recall, and Mutual Information. It performed significantly better than ESTC ($p < 0.05$) on all measurements, except Average and Weighted Recall, where it was almost significantly better ($p < 0.10$).

A more detailed investigation of all test cases shows that QDC was almost universally better than the other algorithms. In 48 of the 52 full text

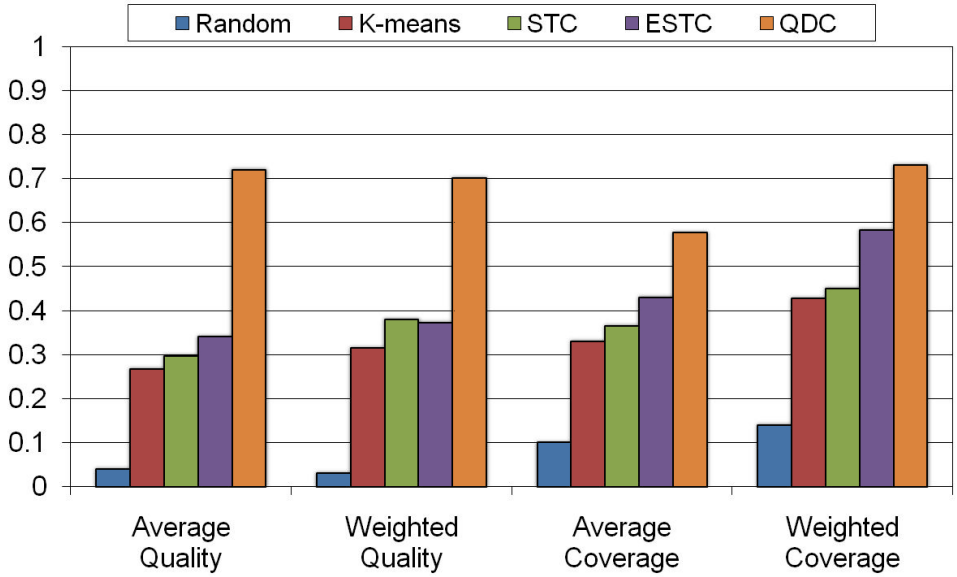

Figure 5.18: Individual QC4 measures averaged over all full-text queries

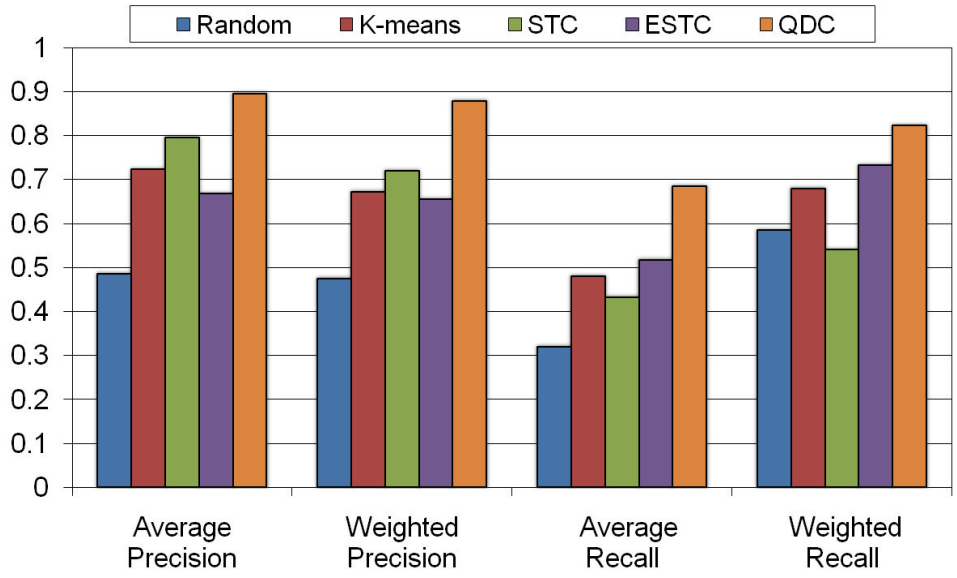

Figure 5.19: Precision and Recall averaged over all full-text queries

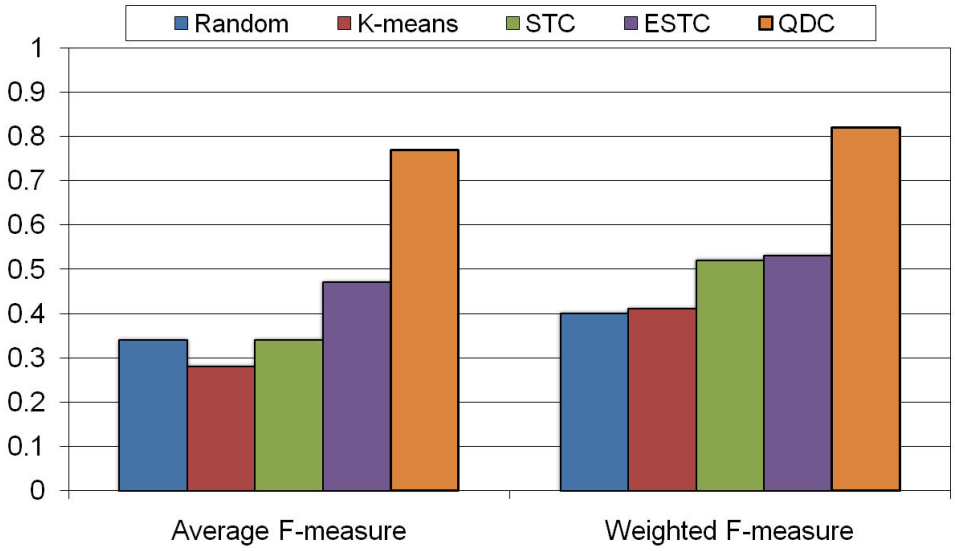

Figure 5.20: F-measure averaged over all full-text queries

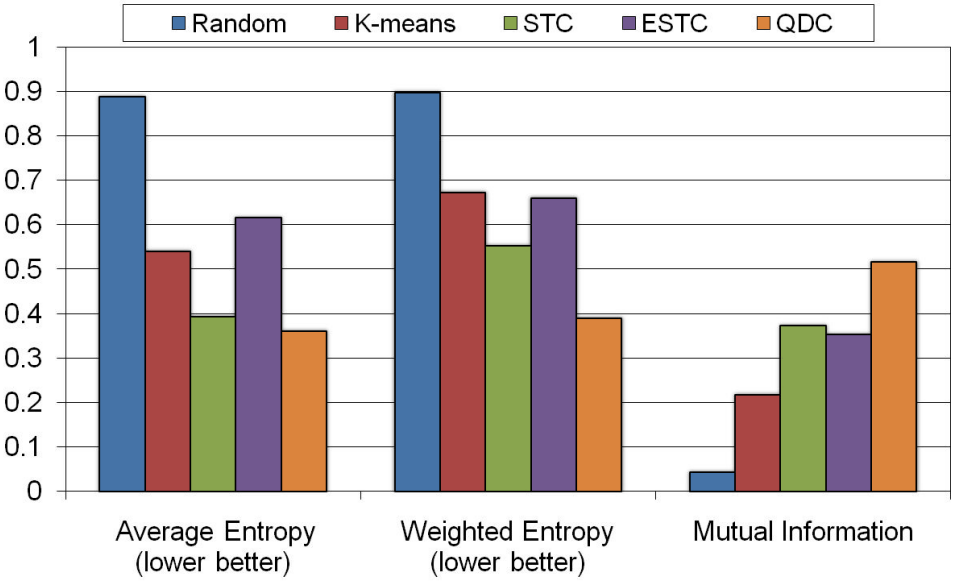

Figure 5.21: Entropy and Mutual Information averaged over all full-text queries

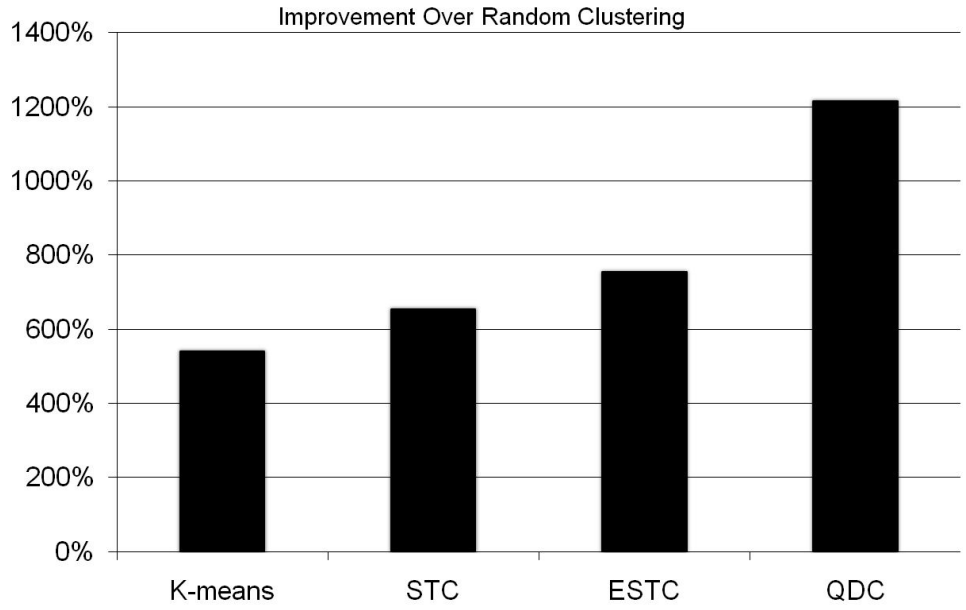

Figure 5.22: Average improvement over a random clustering, measured using a combination QC4 measure over all full-text queries

test cases (13 measurements on each of 4 data sets), QDC was better than all the other algorithms. In the four cases where QDC was worse, QDC had second best performance. The four cases were for the "salsa" data set, which was the easiest search because all algorithms performed comparatively well on this data set. In all cases where QDC performed worse, the advantage of the other algorithms was very marginal (typically a few percent). Furthermore, when considering the tradeoffs, it was clear that QDC performed better overall. When QDC had slightly worse average and weighted precision and entropy than STC, it had much better average and weighted recall and would be better on a combination score that balanced both factors in the tradeoff.

I also evaluated the performance of QDC against the other algorithms at clustering just snippet data. Figures 5.23, 5.24, 5.25, and 5.26 show the 13 measurements averaged across the four data sets and figure 5.27 shows the percentage improvement each algorithm makes over the random clus-

tering. The results show that QDC offers a very large improvement in performance over other clustering algorithms. QDC has better performance in all but the unreliable[17] recall measurements (where QDC is slightly outperformed by K-means[18]), but QDC does better on precision and entropy and would be better on a combination score that balanced both factors in the tradeoff.

On the snippet data, QDC performed significantly better than the Random Clustering ($p < 0.05$) on all measurements. It performed significantly better than K-means ($p < 0.05$) on Average and Weighted Quality, Average and Weighted Precision, and Average and Weighted Entropy. It performed significantly better than STC ($p < 0.05$) on Average and Weighted Quality, Average and Weighted Precision, and Average and Weighted Entropy, and was almost significantly better than STC ($p < 0.10$) on Weighted Recall and Weighted F-measure. It performed significantly better than ESTC ($p < 0.05$) on Average and Weighted Quality, Average and Weighted Precision, Weighted Recall, Weighted F-measure, and Average and Weighted Entropy, and was almost significantly better than ESTC ($p < 0.10$) on Average Recall and Average F-measure.

As with the full text, QDC was almost universally better than the other algorithms on the snippet data sets. In 44 of the 52 snippet test cases, QDC was better than all the other algorithms. In five of the eight cases where QDC was worse, QDC had second best performance. Four of the cases were for weighted recall, a particularly unreliable measure that often gave better performance to the random clustering than to other algorithms. Two of the cases were for F-measure, which frequently gave better performance to the random clustering. The other two cases were the coverage for the "salsa" data set. In all eight cases the coverage or recall were only slightly worse (a few percent), but the quality, precision, and entropy were much

[17] The recall measurements fail here because they overvalue the mere inclusion of pages in the clustering: the random assignment of otherwise excluded pages to new clusters will increase recall.

[18] As k-Means ensures all pages are assigned to a cluster.

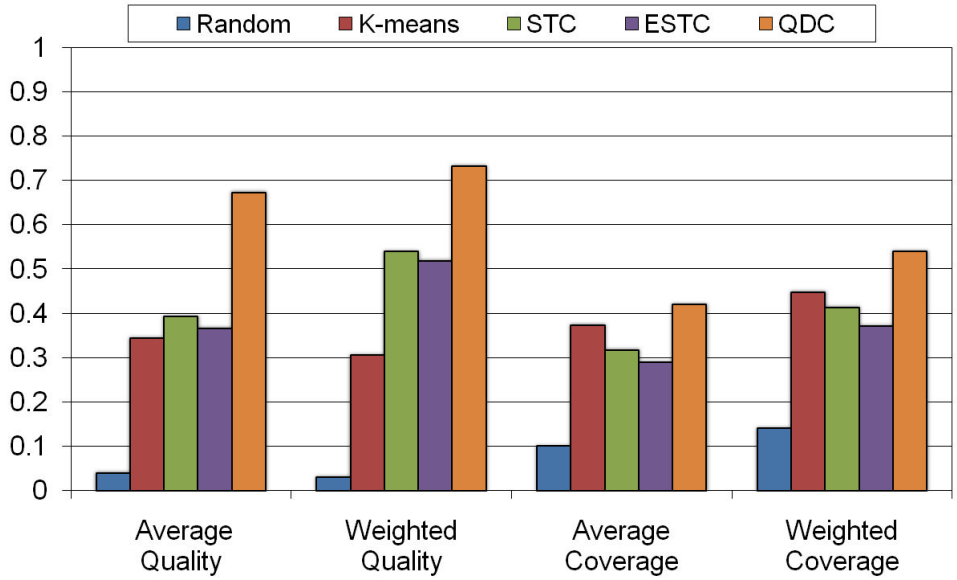

Figure 5.23: Individual QC4 measures averaged over all snippet queries

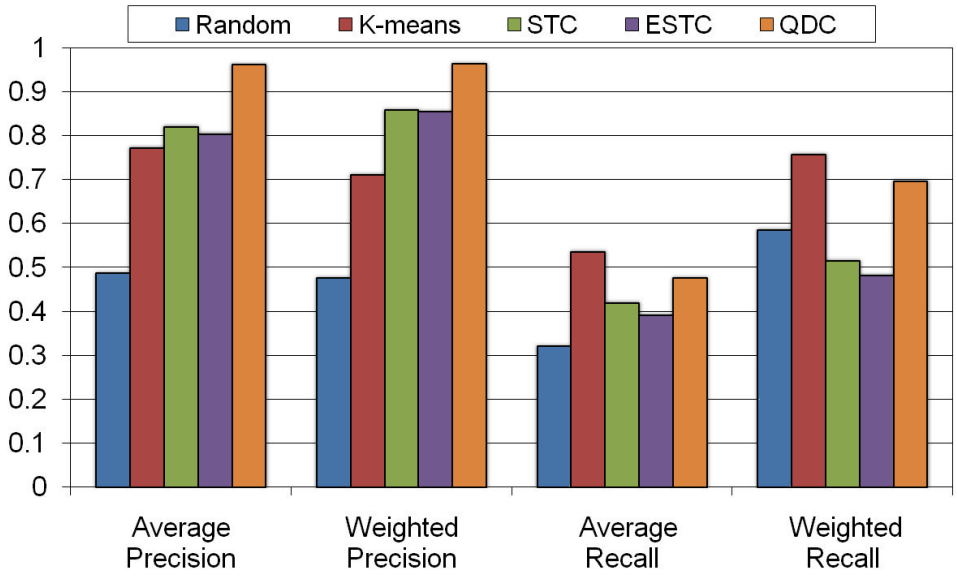

Figure 5.24: Precision and Recall averaged over all snippet queries

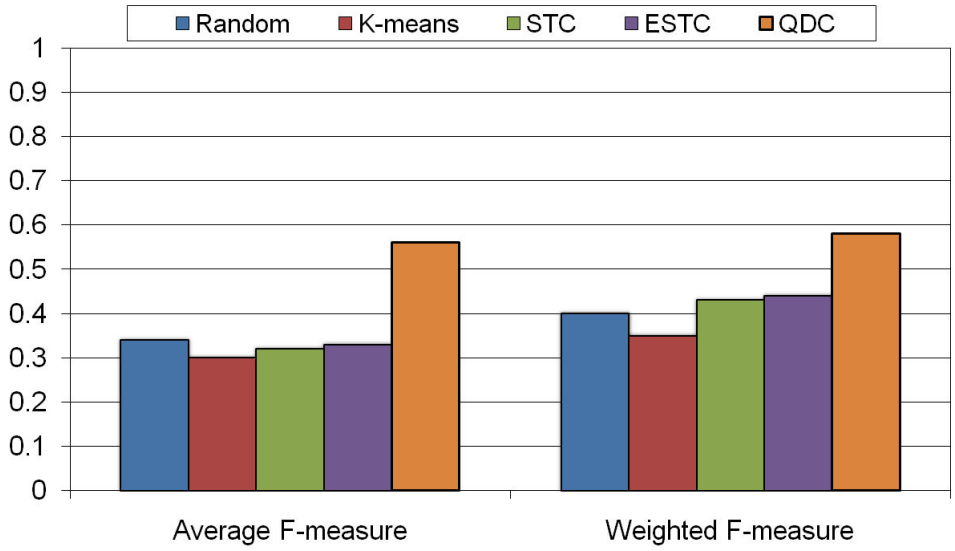

Figure 5.25: F-measure averaged over all snippet queries

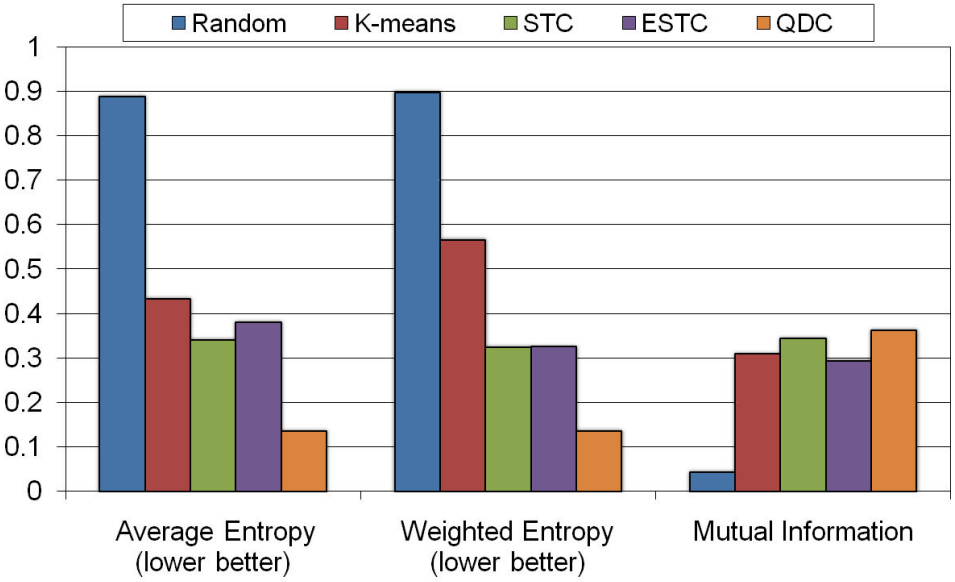

Figure 5.26: Entropy and Mutual Information averaged over all snippet queries

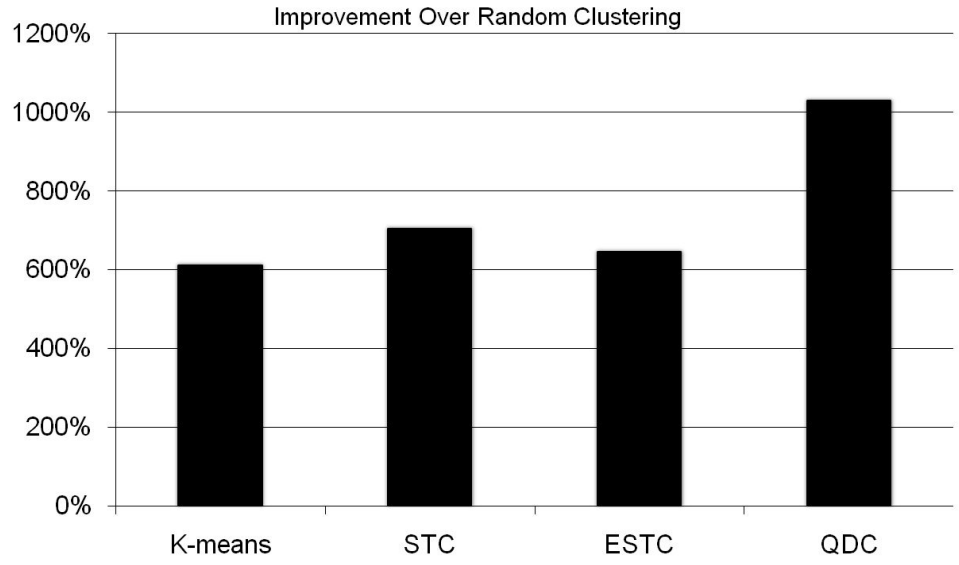

Figure 5.27: Average improvement over a random clustering, measured using a combination QC4 measure over all snippet queries

better (twice as good in seven of the eight cases).

5.10.3 Qualitative Analysis of Clustering Usability

The results of the quantitative external evaluation are impressive, but the real test of a web page clustering algorithm is end user usability. While a formal user study would best confirm the results from the quantitative evaluation, this is costly and time consuming to perform, so instead I performed an informal qualitative analysis and comparison with other clustering algorithms. The analysis used the same four queries as the quantitative evaluation ("salsa", "jaguar", "gp", and "victoria university") and indicates the results from the quantitative evaluation may have underestimated the real world usability and performance of QDC. In addition to the algorithms used during the quantitative analysis, the qualitative analysis also evaluates Lingo and Vivisimo [162]. This section primarily presents the results from the "jaguar" query (the results were similar for the other

queries), but examines all results for QDC. Additionally STC (due to result similarity with ESTC) and Random clustering (due to its obviously poor performance) are excluded here, but were included in the analysis.

Table 5.5 shows the cluster names and number of pages in each cluster produced by QDC, K-means, ESTC, Lingo[19], and Vivisimo[20] [162] for the search "jaguar", sorted by size. Unlike the other algorithms, Lingo and Vivisimo clustered snippets instead of full-page data and used different data sets of 200 and 228 pages respectively. To account for those differences, the Lingo and Vivisimo clusters were adjusted: normalizing cluster sizes to account for the different data set sizes, and truncating overly long cluster names. For Lingo, table 5.5 displays only the ten largest clusters of twenty-five. Table 5.6 shows the names, QC4 Quality, and QC4 Coverage for each cluster produced by QDC for each of the 4 full-text queries.

The qualitative analysis achieves three goals:

1. identifies the cause of the improvement shown by QDC in the quantitative analysis as a reduction in the number of semantically meaningless clusters

2. evaluates additional usability factors ignored by the quantitative evaluation

3. discovers the limitations of QDC (which are also shared by the other algorithms)

Qualitative analysis indicates that the usability and performance of QDC is even better than is shown by the quantitative evaluation, because the quantitative evaluation did not penalize the creation of overly specific clusters since the gold standard included them. Instead, the quantitative evaluation shows QDC produces fewer semantically meaningless clusters, and QDC's clusters had fewer irrelevant pages and covered additional relevant pages.

[19]Lingo results are from http://carrot.cs.put.poznan.pl.
[20]Vivisimo results are from http://www.vivisimo.com.

Table 5.5: Clusters for "jaguar"

QDC		k-Means		ESTC	
Car	109	Include	115	Car	56
Cat	48	Car	22	OS 10	33
Apple	35	OS	17	Panthera onca	21
Atari	18	Free	16	Online	9
		Largest	14	Pictures	9
		Type	13	System	8
		Atari	12	Racing	7
		Service	12	Prices	7
		Panthera	9	Auto	7
				Wildlife	7

Lingo		Vivisimo	
Other	68	Club	48
Locate a Used Car	29	Parts	46
Mac OS Jaguar	24	Jaguar Cars	41
Cat the Jaguar	20	Photos	32
Jaguar Auto Parts	18	Classic	16
Safety Information	16	Animals	7
Jaguar Club	15	Mark Webber	7
Home Page	13	Maya	5
Official Web	13	Enthusiast	4
Amazon.com Books	11	Panthera onca	4

Table 5.6: QC4 Quality and QC4 Coverage of QDC Clusters

Query	Clusters	QC4 Quality	QC4 Coverage
Salsa			
	Dance	0.87	0.88
	Food	0.63	0.87
Jaguar			
	Car	0.70	0.81
	Cat	0.68	0.85
	Apple	0.70	0.79
	Atari	0.96	0.78
GP			
	Racing	0.50	0.67
	Doctor	0.90	0.76
	Games	0.60	0.71
Victoria University			
	Melbourne	0.41	0.49
	Wellington	0.85	0.81
	UVic	0.86	0.46

QDC finds fewer semantically meaningless clusters compared with the other algorithms. For instance, table 5.5 shows that QDC found none when clustering "jaguar", whereas K-means found three ("include", "free", and "service"), ESTC and Lingo each found two, and Vivisimo found one. Furthermore, QDC found no semantically meaningless clusters for the other queries either as shown in table 5.6. This explains much of the QDC's quality advantage in the quantitative results, the remainder is accounted for by the fact that QDC's semantically meaningful clusters contain fewer irrelevant pages. Table 5.5 also shows that QDC finds substantially larger clusters than the other algorithms. Considering these clusters are also of higher quality accounts for QDC's coverage advantage.

The quantitative evaluation overlooked an important usability factor identified in section 5.3.4: the user should see only top-level high quality clusters. Table 5.6 shows the clusters found by QDC for all queries, all are top-level high quality clusters. QDC finds larger, broader clusters such as "Car" that represent top-level topics, while the other algorithms intermix these with smaller more specific clusters that represent sub-topics such as "Locate a Used Car" and "Jaguar Auto Parts". Showing only top-level topics, as QDC does, is preferred. Firstly, it maximizes the probability of successful refinement, because algorithms that select sub-topics often miss some top-level topics entirely because the sub-topics appear more relevant than the smallest top-level topics. For example, ESTC, Lingo, and Vivisimo all miss the top-level "Atari" topic for the "Jaguar" query. Secondly, the decision process is simpler because there is only one relevant choice. For example, QDC shows 4 clusters for the "Jaguar" query and each represents a distinct topic, while the other algorithms find additional clusters that sometimes represent exactly the same topic — for instance, ESTC finds a "Car" cluster and an "Auto" cluster. Additionally, since QDC finds exactly the top-level topics, it can be applied recursively to form an effective hierarchical clustering algorithm.

Close inspection of table 5.6 shows that QDC missed some of the small-

est topics. It missed one "Jaguar" topic and two topics in each of "GP" and "Victoria University". While not ideal, it should be noted that none of the other algorithms found any of these missed topics and indeed, the other algorithms missed additional topics that were much larger like Atari in "Jaguar". QDC missed these small topics, because they were smaller than QDC's minimum base cluster size. There are two reasonable strategies to address this problem. The first is to adjust the parameters so that the topics are larger. QDC can achieve this by simultaneously increasing the number of search results and lowering the base cluster threshold, but comes at the cost of increased runtime. A second strategy is to run a second query, which excludes the clusters from the first query, and to cluster that. For example, in the case of "Jaguar", cluster the search results of the query "Jaguar -Car -Cat -Apple -Atari" and incorporate those clusters with the clusters from the first query. My experiments indicate that the second approach is more efficient[21] and produces clusters of higher quality, although there is little quantitative improvement as only average coverage improves significantly[22], due to the small size of these topics.

5.10.4 Stage Performance and Sensitivity

I conducted experiments to discover the relative importance of each of the five innovations, which roughly correspond to the five stages of the algorithm. Each innovation and stage of the algorithm individually has a positive effect on clustering or algorithm performance, though not as much as the combination of all five.

The query directed cluster quality guide has a large impact on performance. The pruning it enables in the first stage of the algorithm dramatically improves algorithm efficiency and clustering performance. While it is difficult to directly measure the quantitative accuracy of the query distance measure, qualitative analysis (from which figures 5.10 and 5.11 were

[21] twice the cost, so still five times faster than STC

[22] up to approximately the level of weighted coverage

extracted) showed that it performs well at separating specific and broad terms and at identifying terms that would lead to high quality and non-ambiguous clusters.

Using the cluster description similarity in the second stage significantly improves both cluster quality and cluster coverage. Experiments confirmed that the threshold for cluster content similarity could be significantly reduced as a result, which lead to improved coverage.

The new cluster splitting method in the third stage solves the cluster chaining problem and improves cluster quality. This was independently verified by applying the cluster splitting method to synthetic test cases, including cases that exhibited cluster chaining such as those shown in section 5.7.4.

The improved heuristic of the fourth stage improves the efficiency of cluster selection significantly over ESTC and makes far better selections. Quantitative analysis (from which table 5.4 was extracted) confirmed that the heuristic assigns significantly higher quality to high quality top-level clusters than to sub-topics. Qualitative analysis (section 5.10.3) confirmed this enables QDC to select high quality top-level clusters and to choose an appropriate number of clusters to show the user. The improvement to the heuristic is an additional result of the query directed cluster quality guide.

The ranking method used in the final stage improves cluster quality, but does not contribute much to the quantitative evaluation discussed previously in section 5.10.2, because none of the measures used take the ordering into account. An independent analysis confirmed the benefits of the ranking method and found that approximately 90% of the irrelevant pages that were hurting cluster quality were placed in approximately the bottom 10% of the cluster rankings; when sorted by search position, they were distributed randomly throughout the cluster rankings.

The heuristics in QDC use quite a number of parameters. For the experiments, the parameters were set on the basis of what proved effective during the independent analysis of each stage of the algorithm. Further

experiments explored the sensitivity of the results to the parameter values and found that with one exception all parameters were able to be shifted in either direction by at least 20% without making more than a ±2% difference in average clustering performance. The query distance threshold in the first stage of the algorithm was more sensitive: shifting this by 20% could make up to a ±6% difference in clustering performance. This is a further indication of the importance and effect of query distance in QDC. It may be worth tuning this parameter.

Chapter 6

Query Aspect Approach

Although users often do not realize it, relatively few hard searches consume most of the time they spend searching. To improve performance substantially, search engines must make it easier to find relevant documents for these hard searches; either by helping users to refine their queries or by finding results that are more relevant for their existing queries. However, this is not trivial, because even experienced searchers can struggle to find effective queries for hard searches.

When a search goal has multiple components or aspects, documents that represent all the aspects are likely to be more relevant than those that only represent some aspects. Current web search engines often produce result sets whose top ranking documents represent only a subset of the query aspects. By expanding the query using the right keywords, the search engine can find documents that represent more query aspects and performance improves.

This chapter presents the Query Aspect Approach (QAA), a novel method that helps search engines find good query expansion terms. The method uses a simple technique that helps search engines interpret queries in a way that is closer to how users interpret them. The QAA has a number of applications and can help improve performance for many different types of hard search. This chapter explores two of these applications. AbraQ

uses the QAA to improve queries with underrepresented aspects automatically. Qasp uses the QAA to suggest refinements for hard ambiguous queries. The chapter concludes by outlining several other applications including one that addresses queries for narrow search goals.

6.1 Multi-Aspect Queries

Search goals are frequently a composition of several aspects. For example, a search goal of finding possible holidays has one aspect, but searching for travel agents in Los Angeles who deal with cruises involves three different aspects. To express their search goal as a query, users typically select one term (of one or more words) for each aspect from their search goal. Queries for these search goals might be "holidays" and "Los Angeles travel agents cruises". Query aspects are the aspects of the search goal that the query represents explicitly.

Many queries have multiple aspects and queries for hard searches are even more likely to have multiple aspects. I confirmed this by classifying whether 150 random queries from the AOL query logs [139] contained multiple aspects, 50 each from easy searches (those taking less than 5 minutes), hard searches (those taking more than 5 minutes), and very hard searches (those taking more than 30 minutes). Table 6.1 shows the results of this analysis.

Table 6.1: Percentage of queries with multiple aspects
(95% confidence interval)

Query Type	Percentage
Easy	$62\% \pm 13\%$
Hard	$76\% \pm 12\%$
Very Hard	$84\% \pm 10\%$

Since many queries have multiple aspects and a majority of the queries

for hard searches have multiple aspects, users would benefit significantly from an improvement in performance for multi-aspect queries. The QAA helps improve performance for multi-aspect queries.

6.2 Related Work - Query Expansion

6.2.1 Retrieval Model and Term Weighting

Query expansion methods (both AQE and IQE) have three components: the retrieval model, term weighting, and term selection. The retrieval model used for query expansion is determined by the underlying search engine and is usually the Boolean model, the vector space model, or the probabilistic model [150].

Term weighting relates to the importance placed on the different terms used to expand the query and is dependent on the retrieval model. For the Boolean model, the selected terms simply extend the query [78] and in advanced variants may modify the Boolean connectives (*and* and *or*) between query terms [102]. For the vector space model, Rocchio's method [148, 25] is the most common weighting scheme, which increases the weight of relevant terms and decreases the weight of irrelevant terms, although there are also many other methods including Robertson Selection Value, CHI-squared, and Kullback-Leibler distance [30]. For the probabilistic model, a wide variety of approaches exist [183]: Robertson and Sparck Jones [146] introduced four different term weighting schemes and Vinay et al. [183] used a Bayesian approach.

All query expansion methods benefit equally from more sophisticated retrieval models and term weighting, but the manual tuning of parameters is critical to performance [175]. As the focus of the QAA is on term selection, the evaluation of QAA will use a simple Boolean retrieval model, because this removes any bias that the necessary tuning of retrieval model and term weighting parameters may introduce.

6.2.2 Term Selection

Term selection involves finding the terms used to expand the query. Methods that use local document analysis typically select terms from the documents that are assumed to be relevant. For Relevance Feedback [150], these are the documents marked relevant, for Pseudo-Relevance Feedback [156] these are the top N documents in the result set, and for Web Page Clustering (Chapter 5) these are the documents in the selected cluster. The terms are typically ranked using document frequency (df), term frequency (tf), or term frequency inverse document frequency (tf-idf) [181] and the highest ranking terms are used to expand the query. The problem is that these approaches only improve performance when there are at least some relevant documents. For hard searches, there are often no relevant documents near the top of the result set.

Thesauri Expansion [157] and Query Log Analysis [52] use a different approach to select terms. Thesauri Expansion selects terms semantically related to the query terms from a thesaurus such as Word-Net [88] and Query Log Analysis selects terms from similar queries performed previously by other users. Methods can also select terms using global document analysis by looking for words that frequently co-occur with query terms in the whole corpus [195]. The problem with these approaches is that they can increase query drift (changing the meaning of the query), hurting Precision.

While the query expansion methods differ in their source of expansion terms, Theobald et al. [175] found that all query expansion methods aim to select the terms that are the most semantically related to the query terms. This term selection strategy is sensible, because the addition of less closely related terms is more likely to change the meaning of the query, causing unwanted query drift. The QAA uses a different, counterintuitive approach to select terms not necessarily closely related to the query terms. Instead of attempting to avoid query drift, the QAA leverages the change in meaning to shift the search engine's interpretation of the query closer

to the user's interpretation, improving Precision in the process.

6.3 Related Work - Aspects

6.3.1 Query Aspect Identification

There are several methods for identifying query aspects. A simplistic approach is to split the query into single words [128, 199]; this does not sufficiently capture the aspects because many words change substantially from their individual meaning once placed into a sequence. For example, "New" and "New Zealand" have completely different meanings.

Two better approaches, the syntactic tagging approach and the dictionary approach, stem from methods of identifying common phrases in queries [49]. The syntactic tagging approach uses part of speech tagging with natural language queries, but most web search queries are just an unstructured sequence of words and not properly formed natural language (section 2.6.1). Dictionary approaches work with unstructured queries, but can have limited scope and therefore work best with domain specific corpora.

Another approach is the Query Splitting method [200], which uses clustering to separate the query. Although the Query Splitting method processes the query as individual words, the clustering process puts them together into multi-word aspects. However, because Query Splitting ignores the term order from the original query, the method may join disparate parts of the query to create non-existent aspects. For example, the query "Radio Waves and Brain Cancer" may be split into three aspects "Brain Waves", "Radio", and "Cancer", when the aspects are really "Radio Waves", "Brain", and "Cancer". In practice, the best performing implementation failed to identify multiple aspects in 26% of multi-aspect queries [200] and incorrectly split some queries [66]. For example, the query "Lyme disease arthritis" was separated into "Lyme" and "disease

arthritis", when the aspects are really "Lyme disease" and "arthritis".

The QAA introduces a new approach that uses global document analysis to identify multi-word query aspects from both unstructured queries and natural language queries on any corpus.

6.3.2 Underrepresented Query Aspects

Queries have underrepresented aspects when individual documents in the result set do not reflect all the different query aspects, and this has been termed the Aspect Coverage Problem [29]. Searches may fail because the retrieved documents focus on a subset of the aspects[1] or focus on an irrelevant aspect. For example, the query "disasters tunnels transportation" may find some documents that discuss disasters with no relation to tunnels and other documents that discuss tunnels but not disasters. The problem is significant because more than half the categories of search failure involve underrepresented aspects [29].

At the Reliable Information Access workshop [24], 1000 person hours were devoted to analyzing the cause of failure for queries across a range of state-of-the-art search systems. The result of this mammoth effort was the surprising result that higher order constraints and natural language understanding is not especially important for state-of-the-art search systems because they currently fail at a much more basic level: the top ranked documents often fail to reflect some aspects. Of the 44 failures they analyzed, two failed for simple technical reasons (e.g. stemming and tokenization), seven failed due to a lack of natural language understanding or understanding of relationships, and 35 failed due to underrepresented query aspects.[2]

My analysis of a random selection of 15 hard and 15 very hard multi-aspect queries from the AOL query logs [139] using Google reached a sim-

[1] Individual documents may focus on different subsets.

[2] Seven of the 35 involved aspects that would require natural language understanding to identify them.

ilar conclusion. Of the 21 that failed, 12 had underrepresented query aspects, four were ambiguous, and five failed because of higher order constraints and natural language. Note that it is unsurprising that some of the hard searches proved successful on a different search system, because different search systems generally retrieve very different documents. Even so, researchers have found that the major cause of failure for any particular query is usually the same across search systems [24].

The underrepresentation of query aspects is a significant problem because the utility of the result set depends strongly on the number of relevant aspects covered [29].

6.3.3 Using Aspects to Improve Search

Once identified, query aspects are useful for improving search. Researchers [195, 43] have found that aspects affect query expansion and that terms reflecting multiple aspects of the original query often perform better.

Although underrepresented query aspects are a very significant problem, few publications address them directly. Mitra et al. [128] address underrepresented query aspects using the Boolean Constraint method (BoolCon). BoolCon is based on the valid assumption that the more aspects covered by a document, the more likely it is to be relevant. However, BoolCon uses a very simplistic model of aspect coverage. Each non stop word is treated as an aspect and an aspect is considered covered in a document if it contains the word — this effectively moves BoolCon closer to an exact match model. BoolCon incorporates proximity constraints and term correlation to improve performance. The proximity constraints require that words occur together within a window (sequence of words) to account for long documents using terms in unrelated contexts. The term correlations down-weight related aspects, thereby increasing the importance of independent aspects.

While BoolCon improves P@20 by 5% over the baseline (the SMART IR

system with standard AQE [148]), it fails to solve the problem of underrepresented query aspects. The problem is that aspects are not necessarily single words and documents containing the aspect's individual words do not necessarily cover the aspect (in fact, even documents containing the aspect's term as a phrase do not necessarily cover the aspect). Note that Google demonstrably has problems with underrepresented query aspects, even though it finds only pages containing all query terms and probably has equivalents to proximity constraints and term correlations.

Researchers have also used aspects to address other search problems. Neves et al. [133] used aspects in the stepping stones and pathways method to address searches where no single document contains all the desired content, but the desired content can be found from a series of documents. Khan and Khor [101] used local document analysis to discover related aspects not expressed in the query that may be unknown to the user.

The QAA identifies multi-word aspects and then builds vocabulary models for each aspect, which it uses to identify whether documents cover each aspect. Extensions of the QAA address a range of different search problems including underrepresented query aspects and hard ambiguous queries.

6.4 Related Work - Query Difficulty

Query expansion is useful for some queries, but not others; most query expansion methods are only useful for easy queries that already have good performance (section 2.7). Yom-Tov et al. [199] propose using estimates of query difficulty to distinguish easy queries from hard queries and then to improve performance by the selective application of AQE to the easy queries. AbraQ (section 6.8), which uses the QAA, is the opposite of existing AQE methods in that it improves the performance of hard queries, but may decrease the performance of easy queries. Consequently, it also depends on estimates of query difficulty to distinguish the queries that it

is likely to improve.

6.4.1 Evaluation

To evaluate methods of estimating query difficulty, researchers compare the predictions of query difficulty with actual difficulty (measured by Precision) using standard correlation measures such as Kendall's τ, Spearman's ρ, and Pearson's Correlation [199, 29, 186, 84].

Kendall's τ measures the correlation between two rankings using a function of the number of pairwise swaps needed to transform one ranking into the other [186]. Spearman's ρ measures how closely the relationship between predicted and actual difficulty describes a monotonic function. Pearson's Correlation measures how closely the relationship describes a linear function. The range of all three is -1 to 1, where -1 corresponds to inverse correlation, 0 to completely independent predictions, and 1 to perfect agreement.

Let (p_1, a_1), (p_2, a_2), ..., (p_n, a_n) be a set of joint observations from the predicted (P) and actual (A) difficulty values.

$$\text{Kendall's } \tau = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

$$n_0 = \frac{n(n-1)}{2}$$

$$n_1 = \sum_i \frac{t_i(t_i - 1)}{2}$$

$$n_2 = \sum_j \frac{u_j(u_j - 1)}{2}$$

where n_c is the number of concordant pairs, n_d is the number of discordant pairs, and t_i and u_j are the number of tied values in the i^{th} and j^{th} group of ties for the predicted and actual values respectively. A pair, (p_i, a_i) and (p_j, a_j), is concordant if $(p_i > p_j \wedge a_i > a_j) \vee (p_i < p_j \wedge a_i < a_j)$, is discordant

if $(p_i < p_j \wedge a_i > a_j) \vee (p_i > p_j \wedge a_i < a_j)$, and is neither if $(p_i = p_j \vee a_i = a_j)$.

$$\text{Spearman's } \rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the ranks of p_i and a_i. For tied values, the associated rank is the arithmetic average of the ranks associated with the ties.

$$\text{Pearson's Correlation} = \frac{\sum_{i=1}^{n}(p_i - \overline{P})(a_i - \overline{A})}{\sqrt{\sum_{i=1}^{n}(p_i - \overline{P})^2}\sqrt{\sum_{i=1}^{n}(a_i - \overline{A})^2}}$$

Unfortunately, these measures do not give a complete picture of how effectively methods distinguish easy queries from hard queries. Kendall's τ is sensitive to any difference in ranking, but small differences are inconsequential [186]. In this regard, both Spearman's ρ and Pearson's Correlation are superior because they penalizes swaps by the square of the distance. However, none of the measures account for the distribution of easy and hard queries. For web search, the imbalance is very pronounced and most searches are easy (section 2.3.1). Consequently, on typical data sets, all three measures have bias towards the accuracy of predictions for easy queries.

6.4.2 Prediction Methods

The challenge is estimating query difficulty. The robust track of TREC 2004 challenged participants to predict the rank of queries when ordered by their Precision because it proved difficult in the question answering track of TREC 2002 [186].

Researchers [29, 199, 130, 84] have developed a number of methods for predicting the Precision of a query. The methods analyze the query, the documents, the corpus, and the distances between them. Query features include the frequency of query terms in the corpus and the syntactic complexity of natural language queries. Document and corpus features

include the score of the top ranked document, the overlap between the results of sub-queries and the full query, and the similarity of the relevant documents. Distances include Clarity (a measure of the distance between the query and the corpus) and the distinctiveness of the relevant documents from the rest of the corpus.

Using Spearman's ρ, He and Ounis [84] compared various implementations of five different approaches including the frequency of query terms and Simplified Clarity. Of all the approaches, they found that Simplified Clarity has the strongest correlation (0.399[3]) with Precision for short web-like queries. Yom-Tov et al. [199] compared several methods using Kendall's τ and found that methods using the frequency of query terms (0.223) are quite comparable to methods using the top scoring document (0.233). The remainder of this section examines the best approaches, including Clarity, in more detail.

6.4.2.1 Morphological Features

Mothe and Tanguy [130] investigated the usefulness of sixteen different morphological features such as the number of words and word length, syntactical features such as the use of conjunctions, prepositions, and the complexity of sentences, and semantic features such as the ambiguity of query terms. Using Pearson's Correlation they found correlations between sentence complexity and Precision (-0.191), between the ambiguity of query terms and Recall (-0.248), and no significant correlations with any other features. As web users mostly use unstructured queries for web searches and rarely look beyond the first page of results, this approach is unsuitable for the QAA.[4]

[3]Note: comparing correlation values across experiments is problematic because they generally relate to different evaluation measures, test conditions, corpora, etc., all of which can significantly affect the absolute values.

[4]This approach may be useful if applying the QAA in other contexts.

6.4.2.2 Document Overlap

Yom-Tov et al. [199] created a prediction method based on the overlap between the top 10 documents from the result set and the result sets from queries for individual query words. Yom-Tov et al. [199] evaluated the method on two corpora (TREC-8 with 528,155 documents and WT10g with 1,692,096 documents) using Kendall's τ. Using short web-like queries and with the actual ranking set using P@10, the best correlation between the predicted ranking and the actual ranking was 0.268 for TREC-8 and 0.187 for WT10g.

The problem for methods based on document overlap is that in larger corpora there are enough documents that the top ranked documents for most queries will all be different, lowering performance, and this may explain the difference between TREC-8 and marginally larger WT10g corpora. For search engine corpora that are 4 – 5 orders of magnitude larger than WT10g, the problem is likely to be even worse and even with an increased number of documents, there is still likely to be no overlap. For example, in the top 100 documents on Google there was no overlap between "Black" and "Black Bear" and only 1 document in common between "Bear" and "Black Bear". Consequently, approaches based on document overlap are unsuitable for search engine corpora.

6.4.2.3 Difficulty Model

Carmel et al. [29] tried to determine what makes queries difficult and identified the five factors shown in figure 6.1, where Q is the set of queries that express the user's search goal, C is the set of documents in the corpus, and R is the set of documents that are relevant to the user's search goal.

$d(Q, C)$ is the distance between the queries and the corpus and $d(Q, Q)$ is the diameter of the set of different queries that express the user's search goal. $d(Q, R)$ is the distance between the queries and the relevant documents. $d(R, C)$ is the distinctiveness of the relevant documents from the

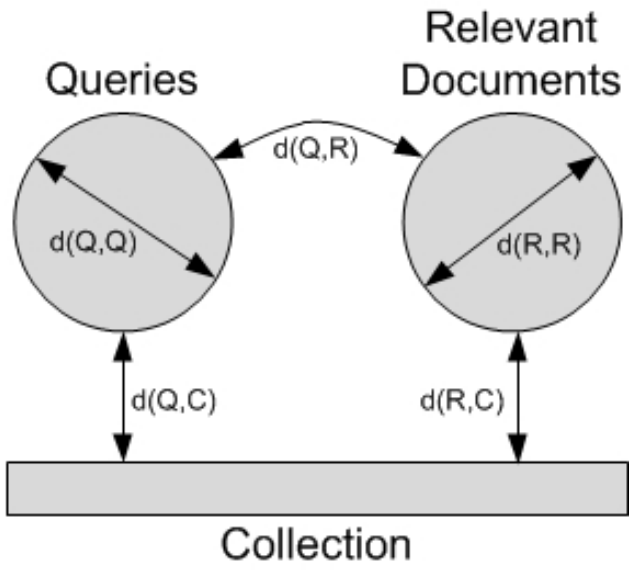

Figure 6.1: Five factors that may affect the difficulty of queries

rest of the corpus and $d(R, R)$ is the similarity between relevant documents.

Carmel et al. [29] evaluated each factor's correlation with the actual difficulty using Pearson's Correlation and Spearman's ρ on GOV2 using 100 queries from the TREC terabyte tracks of 2004 and 2005. As only one query was available for each search goal, they were unable to evaluate $d(Q, Q)$. However, the availability of manually identified relevant documents for each query made it possible to measure the other factors. Their results suggest that $d(R, C)$ is most important factor, that $d(Q, C)$ and $d(R, R)$ influence difficulty substantially, and that $d(Q, R)$ has almost no effect on difficulty.

Without a set of relevant documents, $d(R, C)$ and $d(R, R)$ cannot be computed, which leaves $d(Q, C)$ (Clarity) as the most useful factor. Carmel et al. [29] suggest estimating R and subsequently show using Pearson's Correlation that their distances are significantly better at predicting difficulty than document overlap. However, it is unclear to what extent the

non-estimated $d(Q, C)$ contributed to that improvement. Furthermore, as the evaluation used Pearson's Correlation, it is not possible to distinguish whether the improvement is due to improved performance on easy queries or improved performance on hard queries. As with AQE, the estimates are likely to be reasonable for easy queries (where most retrieved documents are relevant), but poor for hard queries (where few or no retrieved documents are relevant). Consequently, as with typical AQE methods, the incorrect estimates for hard queries may actually reduce the quality of difficulty predictions for hard queries.

6.4.2.4 Clarity

Clarity [51] measures the difficulty of a query by its ambiguity, which it measures using the relative entropy between the words in the documents retrieved by the query (query model) and the words in the corpus (corpus model).

$$\text{Clarity} = \sum_{w \in C} P(w|Q) \log_2 \frac{P(w|Q)}{P(w)}$$

where w is an individual word, Q is the user's query, and C is the corpus. The larger the difference between the probabilities of words in the query model and the corpus model, the higher the Clarity. Focused queries have a high Clarity because they retrieve documents that use a small range of terms frequently — the terms occur with much higher probability in the query model than in the corpus. In contrast, ambiguous queries have a low Clarity because they retrieve documents that use a wider range of terms less frequently — the terms occur with only slightly higher probability in the query model than in the corpus.

Simplified Clarity [84] approximates Clarity by considering just the words in the query.

$$\text{Simplified Clarity} = \sum_{w \in Q} P_s(w|Q) \log_2 \frac{P_s(w|Q)}{P_s(w)}$$

$$P_s(w|Q) = \frac{f(w,Q)}{|Q|}$$

$$P_s(w) = \frac{f(w,C)}{\sum_{w \in C} f(w,C)}$$

where $f(w,Q)$ is the frequency of w in Q, and $f(w,C)$ is the frequency of w in C. The less frequent the query words are in the corpus, the higher the Simplified Clarity. The idea is that specific queries (which typically use rare terms) are usually easier (and less ambiguous) than general queries (which typically use common terms).

Clarity and Simplified Clarity make no assumptions that conflict with web search and they should both correlate with difficulty because both measure factors related to search difficulty (section 2.3.4). However, it is reasonable to expect Clarity, which takes all the words in the corpus into account, to perform better.

Both Clarity and Simplified Clarity were evaluated using Spearman's ρ on TREC-7+8, with Clarity scoring 0.536 [51] and Simplified Clarity scoring 0.399 [84]. While this evidence shows Clarity is better, it is not conclusive, as strictly these values should not be compared as the evaluations were not back-to-back and they used different retrieval mechanisms. Collins-Thompson and Bennett [42] have subsequently shown that Clarity does outperform Simplified Clarity by 27% on GOV2 (0.137 *vs* 0.108) and 394% on WT10g (0.126 *vs* 0.032) according to Kendall's τ.

As Clarity is possibly the best existing measure for short web-like queries that is compatible with search engine corpora, this thesis evaluates the QAA's prediction method by comparing it with Clarity.

6.5 Query Aspect Approach

The Query Aspect Approach (QAA) is a novel approach that reduces the distance between the query models of users and search engines. As explained in section 2.5.1, users and search engines interpret queries in radi-

cally different ways and these differences make it harder for users to construct effective queries. In particular, users expect the order, presence, and semantics of the terms in their queries to affect the results, but search engines generally do not. For example, users expect the presence of query terms to narrow the results, but search engines often underrepresent some query aspects.

The QAA analyzes the order of the words in a query and uses this to identify the different query aspects. Then the QAA builds models of the vocabulary associated with each aspect to capture their semantics. Finally, the QAA uses the models to judge the relevance of a result set by measuring the presence (or representation) of the query aspects in the result set. This new relevance measure also provides a method for predicting query difficulty, because difficulty is merely the inverse of relevance.

6.5.1 Word Order and Aspects

For search engines to find relevant results, they must understand the user's search goal. As the user's query is the only source of information about the user's search goal, search engines must infer the user's search goal from the query. The problem is that one query may be consistent with multiple search goals.

Query elements (words or aspects) can have multiple interpretations because they may have different meanings or occur in different contexts. The cardinality of the set of interpretations for sets of elements is even larger and is potentially as large as the Cartesian product of the interpretations of individual elements. The many interpretations correspond with many different search goals, the problem is deciding which interpretation corresponds with the user's search goal.

By ordering search results by popularity, search engines pick a favourite interpretation and hope that it corresponds to the user's actual search goal. For short queries that have relatively few interpretations, popularity is

usually sufficient to select correctly, but for longer queries that have far more interpretations, popularity becomes more likely to select incorrectly. My analysis of the AOL query logs [139] corroborates this and finds that the very hard searches have longer queries than the hard searches, which have longer queries than the easy searches as shown in figure 6.2.

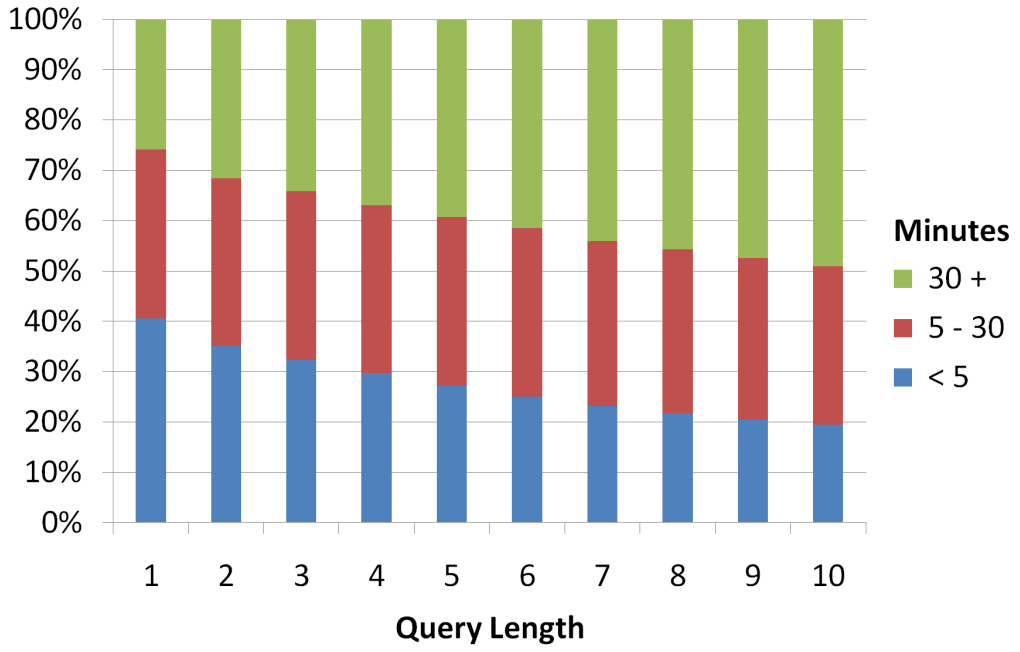

Figure 6.2: The relative fraction of queries of a given length for easy (blue), hard (red), and very hard (green) searches. The easy searches have shorter queries than the hard and very hard searches.

One way for search engines to improve relevancy is to reduce the number of interpretations. As some query aspects contain multiple words, queries may contain fewer aspects than words and therefore, identifying the query aspects can reduce the number of interpretations. The problem is identifying the multi-word aspects.

In both unstructured queries and natural language queries the order of words carries important semantic information: users typically group words related to a single aspect into phrases. For example, users may

search for "microsoft office 2007 reviews" or "reviews microsoft office 2007", but they probably would not search for "microsoft reviews 2007 office" or "2007 office reviews microsoft".

The QAA uses the order of words in the query and global document analysis to identify multi-word query aspects by finding sub-sequences of the query that commonly occur together as a phrase.

6.5.2 Word Semantics and Vocabulary Models

Users assume the semantics of the query terms are important, but search engines ignore semantics and merely seek documents containing the query terms. Query expansion methods add additional terms related to the query, but as they do not necessarily correlate with the query aspects, they often change the meaning of the query, causing query drift.

Different aspects invoke different vocabulary and this vocabulary captures the semantics of an aspect. For example, "microsoft office 2007" may invoke terms like "windows", "spreadsheet", and "word processor", while "review" may invoke terms like "compare", "performance", and "evaluate". The QAA represents the semantics of aspects by building models of the vocabulary of each aspect. The problem is identifying the vocabulary of an aspect.

Any query containing an aspect may find documents reflecting the aspect. These documents are more likely to use the aspect's vocabulary than other documents, but documents from queries with fewer aspects should be preferred because they are less likely to underrepresent aspects and less likely to contain vocabulary associated with other aspects. However, the model should still incorporate information from queries with multi-aspects, because this information can discriminate between the relevant and irrelevant interpretations of ambiguous aspects — generally, only a subset of an aspect's interpretations are relevant in the context of the query. For example, the aspect "Jaguar" may refer to a car, animal, operating sys-

tem, games console, tank, or aircraft, but in the context of a query containing the aspect "turret", only the tank interpretation is likely.

The QAA constructs the vocabulary of each aspect by analysing the term frequencies in documents returned from all the sub-queries consisting of a subset of the full query's aspects, but the QAA weights the sub-queries containing fewer aspects more heavily.

6.5.3 Word Presence and Document Focus

Users choose query terms that describe their search goal and expect the retrieved documents to have a substantial focus on every query term. However, search engines often retrieve irrelevant documents that underrepresent some query aspects.

The problem is that the mere appearance of words in a document is insufficient evidence of the document's focus. For example, the query "black bear attacks" may retrieve many irrelevant documents that contain all three query words. While the irrelevant documents may focus on the black bear aspect by discussing their habitat, diet, and behaviour, they may not focus on the attacks aspect, only mentioning in passing that sometimes they attack (as shown in figure 6.3), making the attacks aspect "underrepresented".

As documents that represent an aspect are more likely to contain many words from the aspect's vocabulary, the presence and absence of vocabulary can form a measure of document focus. If the documents from the original query do not contain enough of the vocabulary of an aspect, then that aspect is potentially underrepresented. For example, the query "microsoft office 2007 reviews" may retrieve documents that do not contain words such as "compare" and "performance", which are associated with "review", indicating that the review aspect may be underrepresented.

The QAA uses the aspect vocabulary models to measure the quality or relevance of a result set by scoring how well the result set focuses on

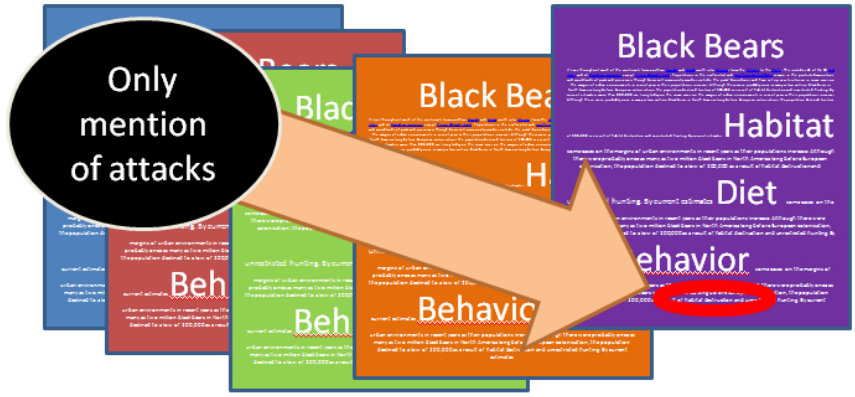

Figure 6.3: Documents for the query "black bear attacks" contain all the query terms, but do not focus on the attacks aspect — the result set underrepresents the "attacks" aspect.

each aspect. This measure has a number of applications: it can identify underrepresented aspects, predict the difficulty of queries, and estimate the performance of different query refinements.

6.6 Aspect Identification

This section describes the implementation of the aspect identification method used by the QAA, analyzes its real-world efficiency, evaluates its performance, and outlines directions for future improvement.

I did not attempt to tune or optimize the parameters of either the aspect identification method or the QAA. Instead, I manually selected reasonable values based on observations from a limited set of separate queries. There were three reasons for not tuning or optimizing the parameters. Firstly, parameter tuning is highly dependent on the corpus and underlying search engine, so any optimization of parameters would not be useful in future research. Secondly, parameter tuning would increase the cost of evaluation by increasing the number of relevance judgments required per query. Thirdly, as the QAA makes extensive use of global document analysis, it

must submit many queries to public search engines to collect the required query result set counts and search engines such as Google and Yahoo impose limits on the number of queries performed. The only drawback to the lack of optimization is that the measured performance may be less than what could have been obtained with optimization.

6.6.1 Algorithm

Taking the user's query Q as input, the QAA removes a limited set of stop words to produce a query Q' of length n. The QAA segments Q' into aspects $A = a_1, a_2, \ldots, a_m$ such that each aspect is a distinct sequence of words from Q' and the ordered concatenation of a_1 through a_m reproduces Q'.

Any subsequence of Q' could represent an aspect. The QAA defines two factors that use global document analysis to determine when a subsequence is an aspect: *existence* and *support*. To be an aspect, the subsequence of words must *exist* — occur frequently enough relative to the frequency of the set of words, and have *support* — occur frequently enough relative to the frequency of all other permutations of the same words.

$$\textit{Existence}(s) = \frac{DP(s)}{D(s)}$$

$$\textit{Support}(s) = \frac{DP(s)}{\sum_{s' \in Perm(s) \setminus \{s\}} DP(s')}$$

where s is a subsequence of Q', $D(s)$ is the number of documents in the corpus that contain all the words in s, $DP(s)$ is the number of documents in the corpus that contain the phase s, and $Perm(s) \setminus \{s\}$ is the set of all permutations of the sequence s other than s itself.

For example, if s is "Air New Zealand", then $D(s)$ is the number of documents that contain the word "Air", the word "New", and the word "Zealand", $DP(s)$ is the number of documents that contain the phrase

"Air New Zealand", and $Perm(s) \backslash \{s\} = \{$"Air Zealand New", "New Air Zealand", "New Zealand Air", "Zealand Air New", "Zealand New Air"$\}$.

The score of a subsequence is the product of the *Existence* and *Support* factors for the subsequence. The QAA considers a subsequence to be an aspect if $\mathit{Existence}(s) \cdot \mathit{Support}(s) \geq 1.0$.

To segment the query into aspects, the QAA checks each subsequence from left to right, largest to smallest, until all words are associated with an aspect. For example, for the query "flights air new zealand sydney departures", the QAA identifies the four aspects "flights", "air new zealand", "sydney", and "departures" by checking the subsequences in the order shown in table 6.2.

Table 6.2: The process used by the QAA to segment the query "flights air new zealand sydney departures" into aspects.

Subsequence	Aspect
flights air new zealand sydney departures	No
flights air new zealand sydney	No
air new zealand sydney departures	No
flights air new zealand	No
air new zealand sydney	No
new zealand sydney departures	No
flights air new	No
air new zealand	**Yes**
flights	**Yes**
sydney departures	No
sydney	**Yes**
departures	**Yes**

6.6.2 Efficiency

Real-world implementations would pre-compute the aspects as part of the indexing process, reducing each run-time subsequence check to an efficient O(1) look-up. This section examines the practicality of pre-computing the aspects and the subsequent runtime cost in real-world applications, and finally explains the approach used in this thesis to deal with the query limits imposed by search engines.

6.6.2.1 Pre-computing Aspects

The practicality of pre-computing the aspects depends on three factors: the number of subsequences that need checking, the cost of checking if a subsequence is an aspect, and the number of aspects that need storing.

Although the number of possible aspects is infinite (as every number is a different aspect), in practice, the vast majority of the aspects are redundant or are of no practical significance. For example, most numbers would have nearly identical vocabulary and many aspects would occur in very few or even no documents.

Wikipedia provides a practical lower bound on the number of aspects: as of July 2010 there are 7.7 million pages and redirects in Wikipedia [193], each of which can be reasonably expected to correspond to an aspect. Google provides a practical upper bound: in their n-gram analysis of a corpus containing more than one trillion tokens [67] they found 3.8 billion subsequences of length one through five that occurred at least 40 times (table 6.3). While any subsequence might be an aspect, even if some excluded subsequences do correspond to semantically meaningful aspects, they are of no practical significance because they occur so infrequently.

The cost of checking a subsequence depends on the cost of calculating $DP(s)$ and $D(s)$. Both are efficient $O(1)$ look-ups as they can be pre-computed. $DP(s)$ would be pre-computed by traversing the corpus and counting the frequency of every phrase of length 1 through n. In fact,

Table 6.3: The number of distinct n-grams that appear at least 40 times in a corpus containing over one trillion tokens [67]

n-grams	Number (millions)
Uni-grams	14
Bi-grams	315
Tri-grams	977
Four-grams	1,314
Five-grams	1,176
Total	3,796

Google's n-gram analysis [67] did exactly this. $D(s)$ would be pre-computed by running queries to calculate the number of search results for each of the 3.8 billion subsequences identified as potential aspects by their frequency ($DP(s)$). It may seem expensive to perform 3.8 billion queries, but Google processed this number of queries every 32 hours in December 2009 [109].

While most, if not all, of the uni-grams are likely to be aspects, relatively few of the bi-grams and tri-grams will be aspects, and very few of the four-grams and five-grams are likely to be aspects. One reason for this is that many n-grams include stop words and punctuation, which is atypical of aspects.

To estimate the fraction that are aspects, I used the Yahoo Random Link selector [196] to sample random pages from the web. I extracted approximately 2000 bi, tri, four, and five-grams from those pages, removed subsequences with a web frequency of less than 500[5], and then manually identified those that were semantically meaningful aspects. The sample included stop words, but unlike the n-gram corpus, it did not include punctuation, so it likely over-estimates the probability of aspects.

[5]The higher cut-off compensates for the expectation that Google indexes far more than the 1 trillion tokens used for the n-gram corpus.

If aspect sequences occur less frequently than other sequences, the random sample may underestimate their probability; therefore, I corrected the estimates using the probability of each n-gram occurring on the web. Table 6.4 shows the resulting probabilities and expected number of aspects. A look-up table containing approximately 40 million entries is very practical and assuming a generous 25 bytes per entry, the table would easily fit in memory (1GB). Even in the worst case, 3.8 billion entries are quite practical and could still fit in memory on a single server (95GB).

Table 6.4: The probability of n-grams being aspects and the expected number of aspects on the web

n-grams	Probability	Number (millions)
Uni-grams	100%	13.6
Bi-grams	3%	9.4
Tri-grams	0.75%	7.3
Four-grams	0.5%	6.6
Five-grams[a]	0.25%	2.9
Total		39.9

[a] The probability of a five-gram being an aspect is an estimate because none of the sampled five-grams were aspects.

6.6.2.2 Runtime Cost

With the aspects stored in a look-up table, the runtime cost of identifying the query aspects depends on the number of subsequence checks per query. In the worst case (when every aspect is a single word), the number of subsequence checks is $\sum_{i=n-n_a+1}^{n-1} i = \frac{(2n-n_a)(n_a-1)}{2}$, where n is the length of the query and n_a is the minimum of n and the length of the longest aspect in the corpus. For example, 30 look-ups are needed for a query with $n = 10$ words if the longest aspect has $n_a = 5$ words.

Since most queries are short, the amortized average number of look-ups is very small. Consequently, the amortized cost of identifying query aspects is very low. Table 6.5 shows the number of look-ups required in the worst-case, based on the distribution of queries in the AOL query logs [139]. Even if there are aspects of length 10, on average only four O(1) look-ups are required per query — in practice even fewer look-ups would be required, because the performance is likely to be better than the worst case and the maximum aspect length is likely to be limited to 5.

Table 6.5: The worst-case amortized average number of look-ups required per query for various maximum aspect lengths

Max Length	Average Look-ups
1	0.00
2	1.75
3	2.75
4	3.30
5	3.60
6	3.77
7	3.86
8	3.93
9	3.97
10	4.00

6.6.2.3 Experiments

For the experiments in this thesis, $DP(s)$ and $D(s)$ were obtained from queries to Yahoo. Since the public search engines impose limits on the number of queries performed, it is desirable to reduce the number of queries. To this end, the experiments in this thesis check subsequences greedily from left to right, smallest to largest, making the assumption that if an i-

gram subsequence is not an aspect, then it cannot be extended to become an $(i+1)$-gram aspect. This reduces the number of queries required, but also reduces accuracy. For example, "air new zealand" cannot be identified because "air new" is not an aspect. Fortunately, aspects that violate this assumption are rare, and consequently, the performance is almost as good as it would be in a real-world implementation without this assumption.

6.6.3 Evaluation

The QAA aspect identification method was tested using the topic titles of all 100 queries from the hard tracks of TREC 2003 [2] (50 queries) and TREC 2005 [3] (50 queries). After removing non-aspect stop words and manually identifying the semantically meaningful query aspects, the queries had the characteristics shown in table 6.6. The following evaluation uses the 97 queries of length two or greater and excludes the single word queries, because they are obviously single aspect queries.

Table 6.6: A summary of the test set, which contains 100 queries from the TREC 2003 and TREC 2005 hard tracks

Query Length	Number of Queries	Average Number of Aspects
1	3	1.00
2	47	1.64
3	43	2.26
4	5	2.60
5	2	4.50

The evaluation compares three different aspect identification methods: Single, Wikipedia, and QAA. The Single method is the commonly used method of treating each individual word as an aspect [128, 199]. The Wikipedia method uses Wikipedia [194] as a dictionary and assumes a

subsequence is an aspect if a page or redirect exists on Wikipedia for the subsequence. The QAA method is as defined in section 6.6.1.

The evaluation removed the non-aspect stop words to improve the interpretability of the results by focusing on the interesting aspect and non-aspect instances and excluding the trivial non-aspect instances. For example, the queries "The History of Nanotechnology" and "The Lord of the Rings" would be transformed into "History Nanotechnology" and "The Lord of the Rings", removing the trivial non-aspects "the history", "history of", and "of nanotechnology", which all algorithms classified correctly.

6.6.3.1 Visual Comparison

Figure 6.4 shows a visual comparison of the three algorithms. Along the x-axis are all the checked subsequences, ordered by the QAA subsequence score, and along the y-axis are the three algorithms. Green indicates the algorithm classified the subsequence correctly (non-aspect between 0.0 and 1.0 or aspect between 1.0 and 1200.0) and red indicates an incorrect classification (aspect between 0.0 and 1.0 or non-aspect between 1.0 and 1200.0).

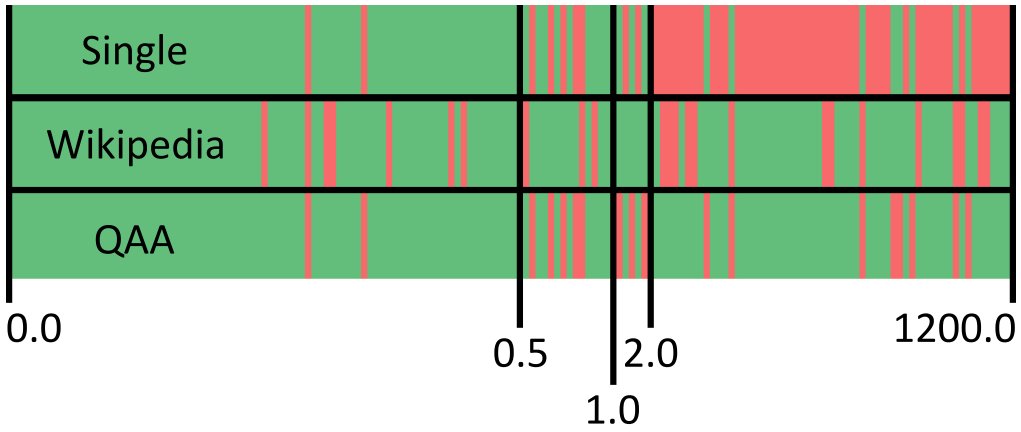

Figure 6.4: Correctly classified subsequences (green) and incorrectly classified subsequences (red)

An interesting observation is that 44% of the errors made by the QAA

occur between 0.5 and 2.0 (margin failures that occur in the region around the arbitrary cut-off of 1.0). Examining this region more closely, the incorrect classifications between 0.5 and 1.0 generally had higher existence and lower support than correct classifications in this region. Additionally, the incorrect classifications between 1.0 and 2.0 generally had lower existence and higher support than correct classifications in this region. This suggests that parameter optimization and a richer classification model such as a trained SVM could improve performance substantially.

As the Wikipedia method is independent of the QAA subsequence score and its overall performance is comparable, the combination of the two should improve performance. To test this, I considered a fourth method, QAA+W, which is identical to the QAA method, except the subsequence is checked using the Wikipedia method when the QAA subsequence score is between 0.5 and 2.0.

6.6.3.2 Macro Analysis

Table 6.7 shows the percentage of queries where every aspect was identified correctly. The results show that overall QAA+W has better performance than QAA, which has better performance than Wikipedia, which has better performance than Single. However, only QAA+W is significantly better than Wikipedia ($p < 0.05$), although all three algorithms are significantly better than Single ($p < 0.01$). The difference on four and five word queries is not significant as there are relatively few queries of that length.

Table 6.8 shows the percentage of checked subsequences that the algorithms classified correctly. Because the Single method never considers any subsequence to be an aspect, it correctly classifies all non-aspect subsequences, but incorrectly classifies all aspect subsequences. Wikipedia is possibly the largest manually constructed dictionary and it is not surprising that it performs well, but as expected, it failed on some subsequences because like all manual dictionary based approaches, it does not

Table 6.7: The percentage of queries with all aspects classified correctly

Words	Number	Single	Wikipedia	QAA	QAA+W
2	47	64%	83%	89%	94%
3	43	33%	74%	77%	84%
4	5	0%	80%	80%	60%[a]
5	2	50%	0%	0%	0%
All	97	46%	77%	81%	86%

[a] Wikipedia and QAA made mistakes with two different 4-grams and unfortunately, QAA+W included both of them. However, because there are only five 4-grams, this difference is not significant.

completely cover the ever-expanding set of aspects. The QAA did particularly well and although it is completely automatic, it actually outperformed Wikipedia (although not significantly). Again, QAA+W is significantly better than Wikipedia ($p < 0.01$) and all three algorithms are significantly better than Single ($p < 0.01$). Additionally, QAA is significantly better than Wikipedia ($p < 0.05$) at classifying the subsequences that are aspects correctly.

6.6.3.3 Micro Analysis of Failures

Although QAA performs well, it could be even better. Under close examination, the failures of QAA can be grouped into four categories: margin failures (44%), stop word failures (6%), common word failures (22%), and technical failures (28%). Section 6.6.3.1 has already discussed margin failures, which occur between a subsequence score of 0.5 and 2.0. Stop word failures cause the misclassification of valid aspects and occur when the removed stop words were a defining part of an aspect, such as "of" in "United States of America". Common word failures cause the misclassification of non-aspects and occur when the frequency of the two aspects is very different, such as in "new hydroelectric" and "recent earth-

Table 6.8: The percentage of subsequences classified correctly

	Number	Single	Wikipedia	QAA	QAA+W
All	157	63%	85%	89%	92%
Grouped by Words					
2	131	60%	85%	90%	92%
3	26	81%	81%	81%	88%
Grouped by Type					
Aspects	58	0%	78%	88%	95%
Non-Aspects	99	100%	89%	89%	90%

quakes" where "new" and "recent" occur far more frequently than "hydroelectric" and "earthquakes". Technical failures cover the remaining errors and these represent general algorithmic failures for which there was no specific cause.

QAA+W demonstrated one way of addressing margin failures with significant performance benefits; the next section outlines additional methods of addressing these failures of the QAA.

6.6.4 Future Improvements

There are several ways future research could address the four categories of failure. Future research could improve subsequence scoring, classification, and processing of stop words.

Term frequency correlates with the common word failures, suggesting that additional subsequence scoring factors could be useful. Future research could investigate a range of additional scoring factors to reduce the number of common word and technical failures.

As the evaluation identified, multiplying existence and support loses some information that would have been useful during classification. Instead of producing a tradeoff between existence and support, an imple-

mentation could consider each factor separately and incorporate them into a trained and optimized classifier such as a SVM. This would also be an ideal way to incorporate any additional scoring factors. Future research could investigate using an improved classification model to reduce the number of margin and technical failures.

Stop words can identify aspect boundaries, for example, the stop words "the" and "of" separate the aspects in the query "the history of nanotechnology". Additionally, as in stop word failures, some aspects contain stop words. Future research could improve the analysis of stop words and identify these differing usages of stop words to reduce the number of stop word and technical failures.

6.7 Aspect Model

This section describes how to identify an aspect's vocabulary and how to resolve aspect ambiguity, describes how the QAA constructs the vocabulary model for an aspect, and describes how the QAA uses the vocabulary model to score how well the result set focuses on each aspect. It then analyzes the real-world efficiency of the QAA, evaluates the QAA's performance at predicting query difficulty, and identifies the QAA's limitations. As with the aspect identification method, there was no attempt to tune or optimize the parameters.

6.7.1 Aspect Vocabulary, Queries, and Ambiguity

The QAA uses two main factors to select the terms for an aspect's vocabulary: the strength of association between a term and the aspect across the whole corpus and the frequency of the term in the result sets of aspect sub-queries.

An aspect's vocabulary is the set of terms that are likely to occur when discussing the aspect. For example, documents discussing "programming"

are likely to use terms like "language", "compiler", and "object". Consequently, terms that frequently co-occur with an aspect across the whole corpus are probably good candidates for an aspect's vocabulary and the frequency of co-occurrence would be a good measure of how closely they are related.

The problem is that many terms that frequently co-occur with an aspect should not be in the aspect's vocabulary. One example is stop words (which should obviously be excluded), but there are others too. Across the whole corpus, most documents that include the aspect's term probably do not focus on the aspect; the words that co-occur with the aspect in these documents should not affect the aspect's vocabulary. Therefore, it would be a bad idea to construct an aspect's vocabulary solely based on co-occurrence information.

The QAA assumes that the top-ranked documents for short single-aspect queries are generally relevant. Consequently, the top-ranked documents from a sub-query[6] for an individual aspect's term are likely to focus on the aspect and probably include terms that belong in the aspect's vocabulary. The problem is that even if the original query was unambiguous (*e.g.* "dealers car jaguar new"), on their own, some aspects may be ambiguous (*e.g.* "jaguar"). Therefore, some of the top-ranked documents from single aspect sub-queries may be irrelevant.

While individual aspects may be ambiguous, at least some aspect pairs are probably unambiguous. Consequently, adding aspect pair sub-queries will capture the context from the query and eliminate the ambiguity of individual aspects. However, the contribution of aspect pair sub-queries should be de-weighted to reduce the chance of introducing irrelevant terms. For example, on its own, "jaguar" might have a weighted vocabulary model, {(car, 0.5), (cat, 0.5)}, which includes some terms related to cars and other terms related to cats. However, in the context of a query that also contains

[6]The query is called a sub-query because it contains a subset of the original query's aspects.

the aspect "dealers", the vocabulary model for the pair of aspects "jaguar" and "dealers", {(car, 0.5), (dealer, 0.5)}, contains information that can resolve the ambiguity. By combining the two models, the weighted vocabulary model for "jaguar", {(car, 0.5), (cat, 0.3), (dealer, 0.2)}, is no longer ambiguous because the terms related to the relevant car interpretation are reinforced.

6.7.2 Vocabulary Model

To construct vocabulary models for each aspect of a query, the QAA runs sub-queries for all aspects and all pairs of aspects, and finds all the terms that occur in the documents returned. For example, for the 3-aspect query "flights air new zealand sydney", the QAA runs 6 sub-queries "flights", "air new zealand", "sydney", "flights air new zealand", "flights sydney", and "air new zealand sydney". Note that the QAA does not use aspect pair sub-queries for 2-aspect queries, because the sub-query would match the original query. To identify the terms, the QAA splits the documents into sentences based on punctuation and html tags, and then after trimming leading and trailing stop words, treats all n-grams from each sentence as terms.

For efficiency, the result set is limited to just the first 10 documents for each sub-query. Retrieving more documents would improve the statistics, but the measure would then need to weight the information from the top-ranked documents more heavily because they are more likely to be relevant, and consequently, more likely to focus on the query aspects.

For each aspect, the QAA ranks the terms (excluding stop words) according to document frequency (the number of retrieved documents containing the word from the sub-queries that contained the aspect), retaining only the top 200 terms. Then, it ranks these 200 terms according to their term weight, a function of the term's strength of co-occurrence with the aspect, retaining only the top 50 terms, which form the vocabulary model

of the aspect.

The co-occurrence strength, $CS(t, a)$, measures how strongly a term t and an aspect a are related across the entire corpus. It is the ratio between the actual and the expected co-occurrence frequency of a term and an aspect. The actual co-occurrence frequency, $CF_a(t, a)$, is the fraction of documents that contain both the aspect and the term. The expected co-occurrence frequency, $CF_e(t, a)$, is calculated by assuming that the aspect and the term are independent and computing the product of the fraction of documents that contain the aspect and the fraction of documents that contain the term.

$$CF_a(t, a) = \frac{DP(t \wedge a)}{D}$$

$$CF_e(t, a) = \frac{DP(t)}{D} \cdot \frac{DP(a)}{D}$$

$$CS(t, a) = \frac{CF_a(t, a)}{CF_e(t, a)} = \frac{DP(t \wedge a) \cdot D}{DP(a) \cdot DP(t)}$$

where $DP(t \wedge a)$, $DP(t)$, and $DP(a)$ are the number of documents in the corpus that contain the phrases for both term t and aspect a, the phrase for term t, and the phrase for aspect a respectively. D is the total number of documents in the corpus.

The term weight, $w(t, a, q)$, measures how strongly a term t and an aspect a are related in the context of query q. It is the sum of the contributions to term t from the sub-queries of q. For sub-queries that contain a and whose documents contain t, the contribution is the co-occurence strength divided by the number of aspects in the sub-query. For other sub-queries, the contribution is zero.

$$w(t, a, q) = \sum_{q' \in \textit{sub-queries}(q)} \frac{C(t, a, q') \cdot CS(t, a)}{|q'|}$$

where *sub-queries*(q) is the set of sub-queries for all aspects and all pairs of aspects from q, $C(t, a, q')$ is 1 if q' contains a and t is in at least one of its documents, otherwise it is 0, and $|q'|$ is the number of aspects in sub-query q'.

The vocabulary model, *Vocab*(a, q), reflects the terms most closely related to the aspect in the context of the query. It is a normalized weighted vector of the 50 terms that have the highest term weight.

$$\textit{Vocab}(a, q) = \big\{\{t_1, w_1\}\{t_2, w_2\} \cdots \{t_{50}, w_{50}\}\big\}$$

$$w_i = \frac{w(t_i, a, q)}{\sum_{j=1}^{50} w(t_j, a, q)}$$

6.7.3 Aspect Representation

The QAA measures the quality or relevance of a query's result set by using the aspect vocabulary models to score how well the result set represents or focuses on each aspect. The measure serves to identify underrepresented aspects and because many hard queries suffer from underrepresented query aspects (section 6.3.2), this measure should also serve as a good predictor of query difficulty.

The raw aspect score, $RAW(a, q)$, is the dot product of the weight vector of the vocabulary model of aspect a and the term frequencies in the documents of the result set for query q. The relative aspect scores, $RAS(a, q)$, are the raw aspect scores normalized so they sum to one, and indicate the relative likelihood that aspects are underrepresented.

An aspect is considered underrepresented if its relative aspect score is below a threshold, which depends on the number of aspects $|q|$. For queries with two aspects, the threshold is 33%; for queries with three aspects, it is 25%.

$$RAS(a, q) < \frac{1}{|q| + 1} \longrightarrow a \text{ is underrepresented}$$

The query difficulty estimate, $QD(q)$, is the difference between the threshold and the relative aspect score of the least represented aspect. A query difficulty estimate of less than 0 corresponds to an easy query with no underrepresented aspects and an estimate of more than 0 corresponds to a

hard query with underrepresented aspects.

$$QD(q) = \frac{1}{|q|+1} - \min_{a \in q}\{RAS(a,q)\}$$

For efficiency, the result set is limited to just the first 10 documents when computing the raw aspect score. Retrieving more documents would improve the statistics, but the measure would then need to weight the information from the top-ranked documents more heavily as the relevance of the top-ranked documents is the most important because users rarely look beyond the first result page.

6.7.4 Efficiency

Real-world implementations would pre-compute as much information about the vocabulary models as possible to reduce the runtime cost of scoring result sets. This section examines what information applications could pre-compute, the runtime cost in a real-world application, and finally the implications of the storage medium on throughput.

Note that where necessary, the calculations in this section assume every query word is an aspect (the worst-case) and that the distribution of queries is the same as in the AOL query logs [139].

6.7.4.1 Pre-computing Vocabulary Models

The cost of computing the vocabulary model of an aspect depends on three factors: the number of aspects in the query, the cost of finding the terms and term frequencies from the sub-queries for aspects and aspect pairs, and the cost of finding the co-occurrence strength between a term and an aspect. The pre-computations should minimize these costs.

The number of query aspects is important, as queries with two aspects are much easier to compute because there are no aspect pair sub-queries to consider. Pre-computing the final vocabulary models for the Cartesian

product of all aspects and all queries is completely impractical — the number of models is approximately $40{,}000{,}000 \times 2^{40{,}000{,}000}$ (the number of aspects multiplied by the number of queries, which is the size of the power set of aspects). However, for two aspect queries, only 40 million vocabulary models are required — one for each aspect. Assuming 300 bytes per model (4 bytes per term and 2 bytes per normalized term weight), these models could fit in memory on a single server (12GB).

By traversing the corpus, a real-world implementation would pre-compute an index and an inverted index to minimize the cost of finding the terms and term frequencies. The index would map documents to terms and their frequencies in descending order of frequency and the inverted index would map aspects to documents and have approximately 40 million entries. These steps are practical because search engines already perform similar steps to create an inverted index of uni-grams to documents with approximately 14 million entries.

With these indexes, finding the terms and term frequencies for an aspect sub-query is a simple $O(1)$ look-up and finding them for all aspect pair sub-queries is comparable to the cost of performing the original query. The main cost of a query is the $O(ql)$ intersection of the document lists, which involves one scan through a document list of length l for each of q query words. Finding the terms for n aspect pairs seemingly requires n queries and should be $O(nql)$. However, there are dependencies between the aspect pairs and the original query. All the intersections for all n aspect pair sub-queries are already being performed in computing the intersections for the original query; they are simply not being recorded. By recording these intersections in parallel with the original query, the n sub-queries are eliminated and the total cost is consequently comparable to the single original query; specifically, it is still $O(ql)$.

The cost of calculating the co-occurrence strength depends on the cost of $DP(t \wedge a)$, $DP(t)$, and $DP(a)$. All three can be pre-computed by one traversal of the corpus and thereby reduced to $O(1)$ look-ups. In fact,

$DP(t)$ and $DP(a)$ were already pre-computed for aspect identification.[7] However, $DP(t \wedge a)$ requires far more storage. Assuming each frequency[8] consumes 2 bytes and the number of terms is equal to the number of aspects[9], 3200TB of storage is required. Rather than storing $DP(t \wedge a)$, $DP(t)$, and $DP(a)$ separately, $CS(t, a)$ would be computed and stored in the same space. Hence, finding the co-occurrence strength is an $O(1)$ look-up.

3200TB is a small amount of data for large search engines and could even be stored in distributed memory (approximately 50,000 servers each with 64GB of memory) — Google was estimated to have one million servers in 2007 with 100,000 more being added every three months [138].

6.7.4.2 Runtime Cost

The aspect representation scores for queries with fewer than three aspects are cheap to compute because they involve no aspect pairs and the vocabulary models have been pre-computed. For queries with one aspect, there is no cost because the sole aspect is by definition represented. For queries with two aspects, there are two $O(1)$ look-ups for the aspect vocabulary models and two 50 element dot products (between the vocabulary models and the result set). Consequently, as at least 55% of queries have fewer than three aspects, the scores are cheap to compute at runtime for the majority of queries.

The scores for queries with three or more aspects are more expensive to compute. However, they are still relatively cheap because real-world implementations can deduce the most expensive information while running the original query and the remaining information has been pre-computed. A real-world implementation would identify the terms for each aspect pair in parallel with the computation of the original query at negligible

[7] Both are subsets of $DP(s)$ since both involve frequent subsequences.

[8] It would be better to store the log frequency.

[9] The set of terms is roughly equivalent to the set of aspects as both reflect meaningful subsequences where stop words are atypical.

cost. Having these, the cost of constructing the vocabulary model is low because it involves performing simple operations on a very small set of elements. Specifically, it involves aggregating the terms (the union of several 200 element lists), finding the top 200 terms, calculating the weight for 200 terms, finding the top 50 terms, normalizing the weights of 50 terms, and finally, computing several 50 element dot products.

6.7.4.3 Throughput

While calculating the term weights involves just 200 $O(1)$ co-occurrence strength look-ups, the throughput is limited by the seek time of the storage medium. While this is no problem for implementations using distributed memory, it is problematic for implementations using traditional hard drives.

Google performed almost 88 billion queries in December 2009 [109], which is on average 33,000 per second[10], or 12.6 million co-occurrence strength look-ups per second. Using traditional hard drives that perform 100 random reads per second, 126,000 hard drives would be required — far more than the 1600 2TB drives required to store the data. While 126,000 hard drives are still practical for large search engines and still less expensive than an in-memory solution, it is not the optimal implementation under either performance or financial constraints.

For performance, an in-memory solution is best and as of 2010, is just six times more expensive than traditional hard drives. The in-memory solution will support future growth and demand peaks for a very long time, whereas the traditional hard drives can only cope with current demand. Furthermore, most of the CPU time on 50,000 servers would be available for other purposes — reducing the effective financial cost of the in-memory solution.

For financial cost, 6400 512GB solid-state hard drives are best and as of

[10]However, searches are unlikely to be evenly distributed across time and peak usage is probably even higher.

2010, are half as expensive as the traditional hard drives. Using solid-state hard drives that perform 20,000 random reads per second, the 6400 hard drives have a throughput of 128 million look-ups per second — an order of magnitude more than necessary and an order of magnitude more than the much larger number of traditional hard drives. Furthermore, using solid-state drives substantially reduces the ongoing cost for power and cooling.

6.7.5 Evaluation

The QAA aspect model is evaluated by comparing its performance at predicting query difficulty against the Clarity measure (section 6.4.2.4). Performance is measured by the correlation to the actual ranking using Kendall's τ, Spearman's ρ, and Pearson's Correlation. This section also compares the rankings visually and against the optimal split of easy and hard queries.

Although the aspect identification method QAA+W performed better than QAA, this and subsequent evaluations in this chapter use the aspects identified by QAA.

6.7.5.1 Test Set

The experiments in this section and the later sections of this chapter used a test set consisting of the fifteen queries shown in table 6.9 that originate from the topic titles of TREC 2005 hard track queries [3]. Zhang et al. [206] identified seven different categories of query in the TREC 2005 hard track; the test set includes the first two or three topics (by topic number) from each of the seven categories and provides a representative set of typical web queries. The queries range in length from two to five words, have between one and three aspects, and consist of both easy and hard queries.

My experiments calculated Precision using relevance judgements based on the description and narrative from the TREC 2005 hard track. The description specifies the user's search goal and the narrative gives specific

Table 6.9: The test set of 15 queries used for evaluation

Query	P@5	P@10
Abuses of Email	40%	40%
Airport Security	20%	40%
Antarctica Exploration	0%	0%
Automobile Recalls	0%	0%
Black Bear Attacks	80%	50%
Cult Lifestyles	0%	10%
Hubble Telescope Achievements	60%	50%
International Art Crime	0%	10%
Iran Iraq Cooperation	80%	70%
Marine Vegetation	0%	0%
New Hydroelectric Projects	60%	60%
Radio Waves and Brain Cancer	80%	70%
Three Gorges Project	80%	80%
Transportation Tunnel Disasters	20%	20%
Wildlife Extinction	40%	30%

details on what would be relevant and irrelevant documents.

6.7.5.2 Precision is not Difficulty

Precision is usually a good measure of query difficulty, but fails when the query inaccurately reflects the user's search goal. For example, the query "car" is easy because it finds pages about cars, but if the user's search goal were to find trucks, then its Precision would incorrectly indicate it is hard. Consequently, it is only valid to use Precision as a proxy for difficulty when the query matches the user's search goal.

Two queries in the test set, "Automobile Recalls" and "Antarctica Exploration", suffer from this problem because they have missing aspects

(section 2.3.3.2) and retrieve documents that are relevant to the query, but irrelevant to the search goal. For example, "Antarctica Exploration" found pages related to historic expeditions, but none related to current or future expeditions (the search goal). Consequently, Precision incorrectly indicates that "Antarctica Exploration" is a hard query, when it is actually an easy query.

To resolve this problem, "Automobile Recalls" and "Antarctica Exploration" were excluded from the test set for the query difficulty evaluation. However, they are included in the evaluations later in this chapter where Precision directly measures performance.

6.7.5.3 Clarity

Cronen-Townsend et al. [51] estimated $P(w|Q)$ for the Clarity measure using Bayesian inversion with the word frequencies estimated by linearly smoothing the collection frequencies with the word frequencies in a sample of 500 documents from the set of documents containing any individual query term. While it is impractical to calculate Clarity over every word in the corpus of public search engines, far more accurate estimates of $P(w|Q)$ can be computed using term frequencies[11] from the entire corpus rather than just a sample of 500 documents. In its comparisons with Clarity, this thesis uses the more accurate estimates of $P(w|Q)$, but rather than use all the words in the corpus, it only uses the most informative terms (the most frequent terms that are not stop words).

Table 6.10 shows the average of Kendall's τ, Spearman's ρ, and Pearson's Correlation between Clarity and the actual ranking when using different numbers of informative terms to compute Clarity. The experiments show that Clarity performs best using 20 terms and that adding additional terms may reduce its performance. Consequently, subsequent experiments use 20 terms to compute Clarity.

[11]The term frequencies are found by issuing queries to public search engines.

Table 6.10: The average correlation between Clarity and the actual ranking as determined by P@5 and P@10

	Correlation	
Number of Terms	P@5	P@10
10	0.246	0.149
20	0.279	0.159
30	0.264	0.152
40	0.266	0.143
50	0.244	0.118

6.7.5.4 Results

Table 6.11 shows the degree of correlation between the difficulty rankings predicted by Clarity and the QAA and the actual rankings given by P@5 and P@10. It also shows the correlation between the rankings of P@5 and P@10, which provides a measure of how accurately P@5 and P@10 reflect the true difficulty ranking — if both reflected the true ranking, then P@5 and P@10 should correlate perfectly. In all cases and under all three measures of correlation, the QAA correlates with the actual rankings significantly more than Clarity. That P@5 & P@10 are only slightly more correlated than QAA & P@5 and QAA & P@10 suggests that the QAA is predicting query difficulty very accurately.

Since the test set contained six easy queries and seven hard queries, a more balanced mix of query difficulties than typical test sets, the correlations presented in table 6.11 should provide an accurate picture of how effectively the methods distinguish easy queries from hard queries. To confirm this hypothesis, figures 6.5 and 6.6 compare the predicted rankings visually and figures 6.7 and 6.8 compare the predicted rankings against the optimal split of easy and hard queries.

Figures 6.5 and 6.6 show the precision for each query in rank order.

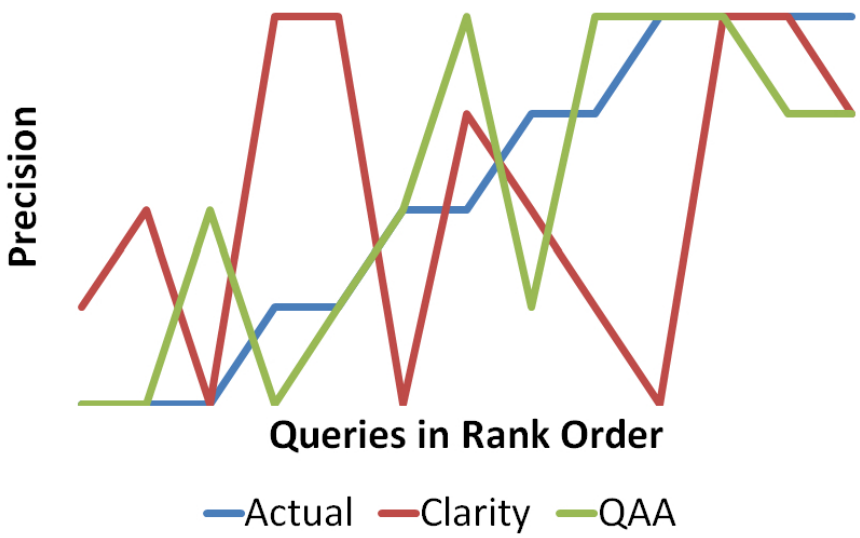

Figure 6.5: The predicted and actual rankings for P@5

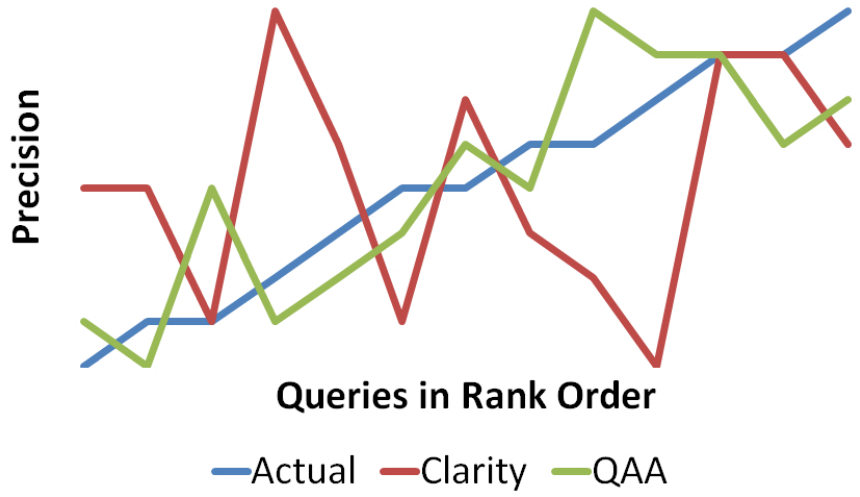

Figure 6.6: The predicted and actual rankings for P@10

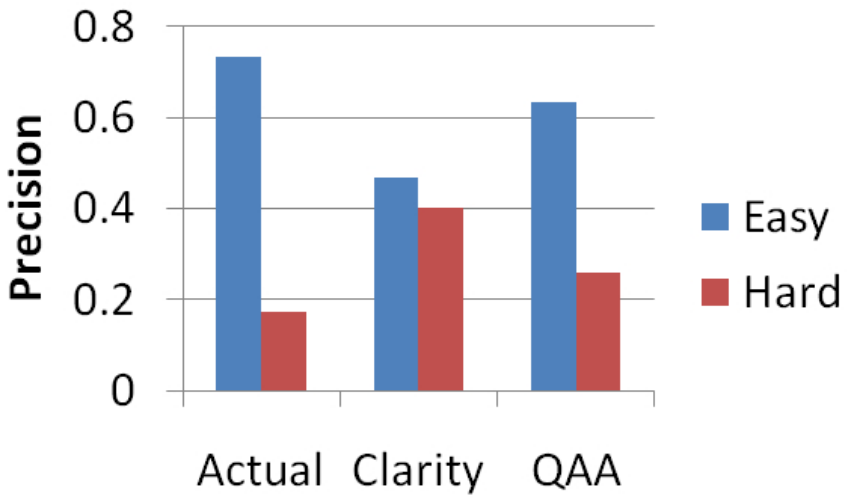

Figure 6.7: Average P@5 of the six easiest and the seven hardest queries according to the predicted and actual rankings.

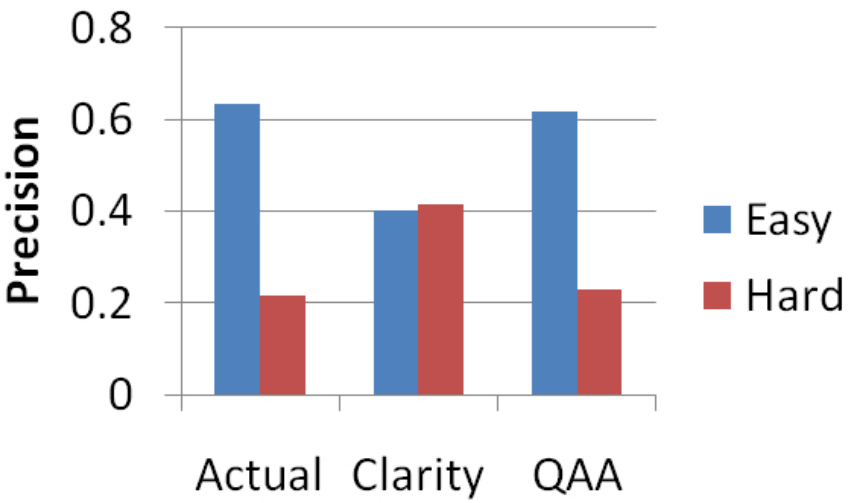

Figure 6.8: Average P@10 of the six easiest and the seven hardest queries according to the predicted and actual rankings.

Table 6.11: Correlation between the predicted and actual rankings for Clarity and the QAA

			Kendall	Spearman	Pearson
Clarity	&	P@5	0.293	0.377	0.167
QAA	&	P@5	0.749	0.723	0.557
Clarity	&	P@10	0.173	0.226	0.079
QAA	&	P@10	0.824	0.782	0.632
P@5	&	P@10	0.941	0.937	0.859

A method that made perfect difficulty predictions would be monotonically increasing and mirror the actual ranking shown in blue. Although the QAA makes local ranking errors, it clearly separates easy queries from hard queries. Figures 6.7 and 6.8 confirm this, showing that the QAA separates the easy and hard queries almost as well as the actual ranking.

6.7.6 Assumptions and Limitations

The QAA aspect model assumes all single aspect queries are easy. However, some single aspect queries may be hard. Future research could investigate how to predict the difficulty of individual aspects and how this knowledge can improve multi-aspect models.

The QAA aspect model makes an independence assumption similar to Naive Bayes, in that it treats the weightings from multi-aspect sub-queries as independent of the weightings from single-aspect sub-queries. Performance may improve by using a richer Bayesian model.

The QAA aspect model treats synonymous aspects as though they are distinct. Performance may improve by modelling the relationships between query aspects.

The QAA aspect model treats sub-queries as sets of words, the same way as in the original query. It is possible that performance would im-

prove by treating the aspects as phrases in the sub-queries, because using phrases would reduce ambiguity. However, using sets is more robust because it allows the aspect models to compensate for misclassified aspects.

6.8 AbraQ

The QAA is very powerful and can improve web search in numerous ways. AbraQ is an automatic query expansion method that uses the QAA to improve queries with underrepresented aspects.

This section identifies how to refine queries with underrepresented aspects, describes the implementation of AbraQ, analyzes its real-world efficiency, evaluates its performance, and identifies its limitations and methods of addressing them. As with QAA, there was no attempt to tune or optimize the parameters.

6.8.1 Refining Queries with Underrepresented Aspects

Constructing an effective query for a search goal involves finding a conjunction of terms that occur together in documents that satisfy the search goal. The terms must be both distinctive and discriminating: they must occur in many of the desired documents, while not appearing together in undesired documents.

Users typically have trouble refining queries with underrepresented aspects because they often limit their choice of refinements to a small set of descriptive terms that are not necessarily very discriminating. Search experts fare better by also considering a larger set of semantically orthogonal keywords (section 2.3.5) that co-occur with the descriptive terms — many of which are effective because they are both distinctive and discriminating. However, it is incredibly hard for even search experts to identify semantically orthogonal keywords, because users naturally think descriptively.

The QAA provides not only a mechanism for identifying queries with underrepresented aspects, but also the necessary tools for refining them. Finding an effective refinement involves two parts: identifying possible refinement terms and measuring their effectiveness. The QAA vocabulary models are a good source of semantically orthogonal keywords, and consequently, candidate refinements, since they contain terms that co-occur frequently with an aspect's descriptive term. Although some refinement terms from the vocabulary models of underrepresented aspects are likely to be good refinements, others may cause query drift, decreasing performance. The QAA aspect scores are a good way of measuring refinement effectiveness, because good refinements should produce result sets with no underrepresented aspects — underrepresented aspects are an indication of query drift.

A significant benefit of this QAA derived approach is that refinement generation is independent of whether the original query retrieves relevant documents. Consequently, this approach is applicable for even the hardest queries that initially retrieve no relevant results.

6.8.2 Algorithm

AbraQ uses the QAA to identify the query aspects and to identify the underrepresented aspects. If there are no underrepresented aspects or there is only one query aspect, AbraQ makes no refinement and presents the result set of the original query to the user. If there are underrepresented aspects, AbraQ picks the aspect with the worst relative aspect score and tries to improve the representation of that aspect.

The vocabulary models from the QAA use term weightings based on the co-occurrence strength of the term with the aspect. AbraQ identifies candidate refinement terms by selecting the N highest weighted terms from the vocabulary model of the least represented aspect (the experiments use $N = 40$). For each term, AbraQ constructs a new refinement

query consisting of the old query plus the refinement term, runs the new query, then re-computes all the aspect scores (as in QAA), but using the result set of the refined query instead of the original query.

To score a refinement q', AbraQ sums the new aspect scores, weighting the previously underrepresented aspects more heavily.

$$RS(q') = \sum_{a \in A} \frac{RAW(a, q')}{RAS(a, q)}$$

where A is the set of aspects in the original query and q is the original query. AbraQ then presents the result set of the highest scoring query refinement to the user.

6.8.3 Efficiency

The cost of AbraQ depends on two parts: the run-time components of the QAA and the additional queries.

The run-time components of the QAA are insignificant compared to the cost of the additional queries. Identifying the aspects of the original query is very efficient as explained in section 6.6.2.2. Scoring the original query is also very efficient, because it involves simple operations on what are relatively very small sets (section 6.7.4.2) — specifically, the cost of scoring is insignificant compared to the cost of a query. AbraQ must also score the refinement queries, but these are even less expensive to score than the original query, because there are correlations between the computations required for the original query and the refinement queries.

The major cost of AbraQ is the 40 additional refinement queries. The impact on search responsiveness is small because AbraQ can perform the additional queries in parallel. While AbraQ enhanced queries will consequently be no slower from the user's perspective, additional hardware will be required to process the same number of queries. However, the amount of additional hardware required is somewhat less than might be expected.

For most queries, AbraQ does not need to perform any additional queries. AbraQ performs no additional queries for the 35%[12] of queries that contain just one aspect because AbraQ terminates after identifying the query aspects. AbraQ performs no additional queries for the 79%[13] of queries that are easy (search engines already perform well and the queries consequently have no underrepresented aspects) because AbraQ terminates after identifying there are no underrepresented aspects. However, AbraQ must perform the additional queries for the 60%[14] of hard queries that have underrepresented aspects. Assuming all the single aspect queries are easy, AbraQ performs the 40 additional queries for just 13% ($60\% \times (100\% - 79\%)$) of all queries.

The additional queries are less expensive than the average query. The mean query length is approximately 2.75 words (section 2.2), and consequently, the average query involves scanning 2.75 document lists. However, each additional query for AbraQ only involves scanning one additional document list, because there are correlations between the additional queries. In total, the 40 additional queries involve scanning 41 additional document lists (one for each additional query term and one for the result of the original query) as compared to the 110 document lists required for 40 average queries.

Accounting for the frequency of underrepresented aspects and the reduced cost of the additional queries, the cost of AbraQ is very practical. The amortized average cost per query of AbraQ is just two additional queries ($\frac{41}{110} \times 13\% \times 40$). In practise, the cost would be even lower, because the analysis does not account for the reduced need for users to refine hard queries.

[12]based on my sample of 150 random queries from the AOL query logs (section 6.1)

[13]based on the AOL query logs (section 2.3.1)

[14]based on my sample of 30 random hard queries from the AOL query logs (section 6.3.2)

6.8.4 Evaluation

I evaluated AbraQ using the non-interactive method of evaluating search results as described in chapter 3. The evaluation used Google as both the baseline search engine and corpus, used $P@N$ to measure performance (specifically, $P@5$ and $P@10$), and used Wilcoxon's signed-rank test for significance testing. Wilcoxon's signed-rank test is a reliable non-parametric hypothesis test for related samples that does not assume the population is normally distributed — Zobel [210] found it superior to ANOVA and Student's t-test when measuring relative performance of search systems.

The 15 queries shown earlier in table 6.9 are used as the test set.[15] It is important that the test set include hard queries, because easy queries already have good performance and do not require refinement. To this end, the test set was selected from the hard track of TREC 2005, which consists of hard queries that no TREC system was able to handle well in previous years [3]. Surprisingly, Google found many relevant results for six of the fifteen queries, confirming my hypothesis that Google is a strong baseline. Of the nine hard queries, Google found no relevant results for five of them and only a few relevant results for the other four.

The rest of this section compares AbraQ to the baseline Google, to a set of automatic query expansion methods, and to a set of interactive query expansion methods. Finally, it identifies the limitations of AbraQ.

6.8.4.1 Google

AbraQ expanded eight of the fifteen queries in the test set as shown in table 6.12. AbraQ did not expand the remaining seven queries because it determined that they did not contain any underrepresented aspects. Qualitatively, the expansion terms chosen by AbraQ appear similar to the se-

[15] except the results in section 6.8.4.2, which were collected before the rest using just 10 of the queries in table 6.9 — Abuses of Email, Antarctica Exploration, Automobile Recalls, Iran Iraq Cooperation, and Wildlife Extinction were not included

mantically orthogonal keywords that search experts would choose. For example, the term "injury" is not directly related to "Transportation Tunnel Disasters", but disasters typically result in injuries, so one would expect pages about "Transportation Tunnel Disasters" to include the term "injury" if the disaster aspect were represented.

Table 6.12: AbraQ expanded eight of the fifteen queries in the test set

Query	Expansion
Abuses of Email	Violations
Airport Security	Protection
Black Bear Attacks	Victim
Cult Lifestyles	Religion
International Art Crime	Selling
Marine Vegetation	Plant
Transportation Tunnel Disasters	Injury
Wildlife Extinction	Conservation

Figures 6.9 and 6.10 compare the performance of Google and AbraQ using $P@5$ and $P@10$ respectively, for all queries, the queries expanded by AbraQ, and the queries not expanded by AbraQ. The results show that AbraQ improves the performance of hard queries, while not affecting the easy queries. Specifically, the results show that AbraQ performs significantly better ($p \leq 0.01$) than Google under both $P@5$ and $P@10$.

The benefit of AbraQ for users is even greater than is initially apparent: improving hard queries can mean the difference between satisfying the search goal or not, whereas easy queries already have sufficient answers and so the benefit of improving easy queries is less.

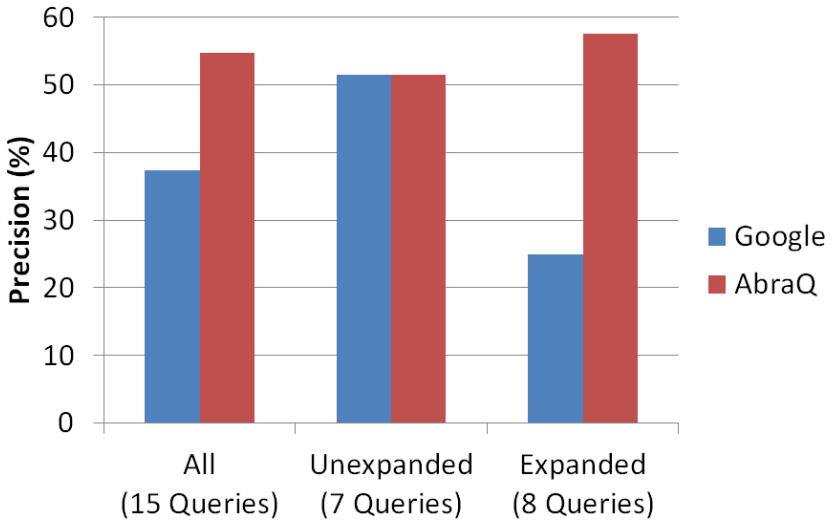

Figure 6.9: Comparing AbraQ with Google using $P@5$ on all queries, unexpanded queries, and expanded queries

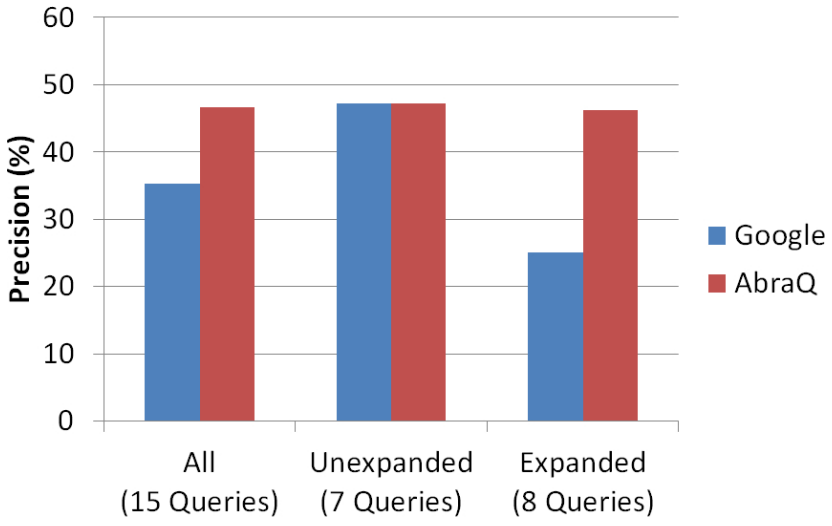

Figure 6.10: Comparing AbraQ with Google using $P@10$ on all queries, unexpanded queries, and expanded queries

6.8.4.2 Automatic Query Expansion

AbraQ provides automatic query expansion (AQE), which means that it directly presents the user with a result set and takes no input from the user beyond their original query. I compared AbraQ against six different AQE methods (strictly, Pseudo-Relevance Feedback methods). As is typical of AQE methods [181], the three term ranking methods used were document frequency (df), term frequency (tf), and term frequency inverse document frequency (tf-idf). The two selection methods used to select the terms for query expansion were the top term (1), and the disjunction of the top five terms (5). This gave six methods (df1, df5, tf1, tf5, tf-idf1, and tf-idf5) for AQE, all of which assume the top five documents are relevant.

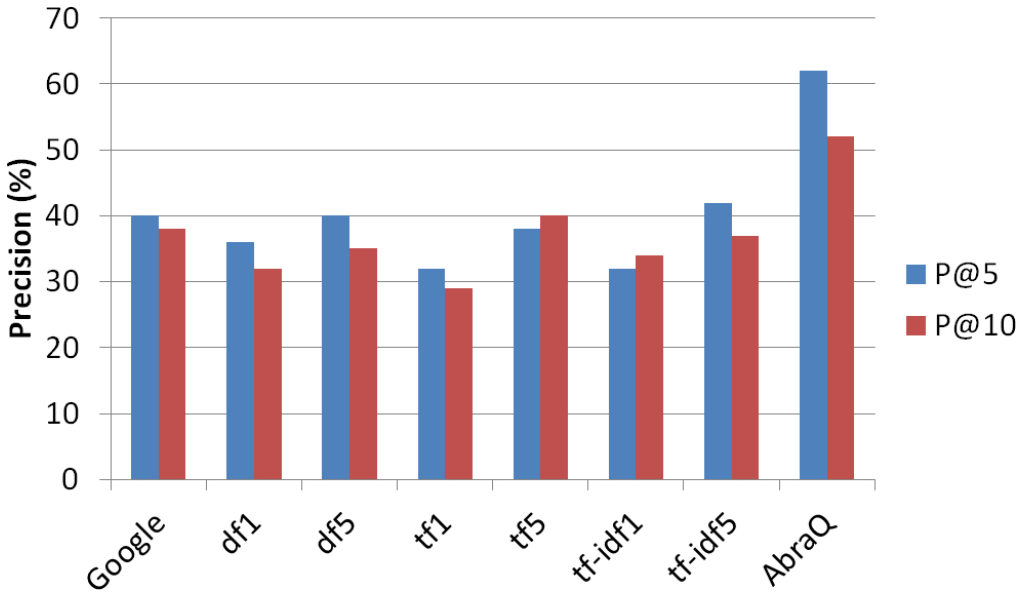

Figure 6.11: Comparing AbraQ with Automatic Query Expansion methods and Google using $P@5$ and $P@10$

While AQE methods are typically used to enhance recall and not precision [181], the overall effect on precision is usually slightly positive [128]. However, the results shown in figure 6.11 disagree and show that traditional AQE methods have a slightly negative impact on precision.

The composition of the test set and the baseline search engine explain the disagreement between the results. The test set consisted of an unusually large number of hard queries and it is well established [128, 199] that traditional AQE methods fail on hard queries. The underlying search engine, Google, is a very strong baseline that probably implements an equivalent to traditional AQE. In general, the effects of search improvements are neither additive nor independent, and consequently, methods that improve weak baselines may hurt strong baselines [7]. This reinforces the significance of using strong baseline systems to evaluate performance.

Added to strong baseline systems like Google, at best, the traditional AQE methods have little effect on precision and may reduce it. In contrast, AbraQ provided substantial improvement to many queries without decreasing performance of any others. There is no significant difference between Google and any of the six traditional AQE methods, but AbraQ was significantly better ($p \leq 0.01$) than Google and each of the other six methods under both $P@5$ and $P@10$.

6.8.4.3 Interactive Query Expansion

There are many term selection methods (section 6.2.2) that are superior to those used by traditional AQE methods. However, unlike AbraQ, they are interactive query expansion (IQE) methods that depend on collecting additional information from the user. Consequently, IQE methods do not compete with AbraQ and search engines could use them in concert with AbraQ to solve other types of problematic query (section 2.3.3).

The IQE methods represent something akin to an upper bound on term selection — AbraQ would be doing well to approach the performance of methods that have the added advantage of user input. This section compares AbraQ to the one-step refinement performance of a representative range of IQE methods that includes clustering, query log analysis, relevance feedback, and pseudo-relevance feedback methods.

Methods

AbraQ was compared against three of the most successful clustering approaches — Suffix Tree Clustering (STC) [201], Lingo [137], and Query Directed Clustering (QDC) — as well as a Query Log Analysis method (Mamma), a relevance feedback method (RAQE), a pseudo-relevance feedback method (PRIQE), and an interactive variation of AbraQ (Optimal).

Clustering methods group similar documents from the result set together to form different refinements. STC and Lingo use phrases to form clusters and QDC (the clustering algorithm introduced in chapter 5) uses the relationship between words and the query to form clusters. The evaluation limited the methods to producing a maximum of 15 clusters.

Query Log Analysis methods use large query logs to suggest possible refinements. The Mamma search engine [117] suggests a number of refinements alongside the search results of each query based on query log analysis. The refinements are comparable to Google's subsequently released related search feature (section 2.6.2).

Relevance Feedback methods are similar to AQE methods, but rather than assuming the top N documents are relevant, they require the user to specify which documents in the initial result set are relevant. RAQE has the user specify the relevant documents in the top 10, then ranks (using tf-idf) the terms that occur in the relevant documents that do not occur in irrelevant documents, and finally, selects the best term for refinement.

Pseudo-Relevance Feedback methods are identical to AQE methods, but instead of automatically applying the top ranked term, they present the top ranked terms to the user and require the user to choose the best one. PRIQE presents the user with the top 15 terms according to tf-idf.

Optimal (an interactive variation of AbraQ) presents the user with the top 15 terms according to the AbraQ refinement score.

Results

For each IQE method, the evaluation assumed the user is perfect and that they select optimally from the refinement suggestions or correctly choose to make no refinement. This is an optimistic assumption that will credit the IQE methods with a higher score than would be achieved in practice. Magennis and van Rijsbergen [114] found that from the perspective of recall enhancement, while user selections improve performance, they failed to reach the optimal performance and in general performed worse than even AQE techniques. Users are unlikely to fare better at selecting precision enhancing refinements and therefore, the relative performance of AbraQ would be better in practice than this evaluation suggests.

Figure 6.12 shows the results for AbraQ, Google, and all IQE methods averaged over all 15 queries in the test set. Figures 6.14 and 6.13 show the results averaged over the queries that were and that were not expanded by AbraQ respectively.

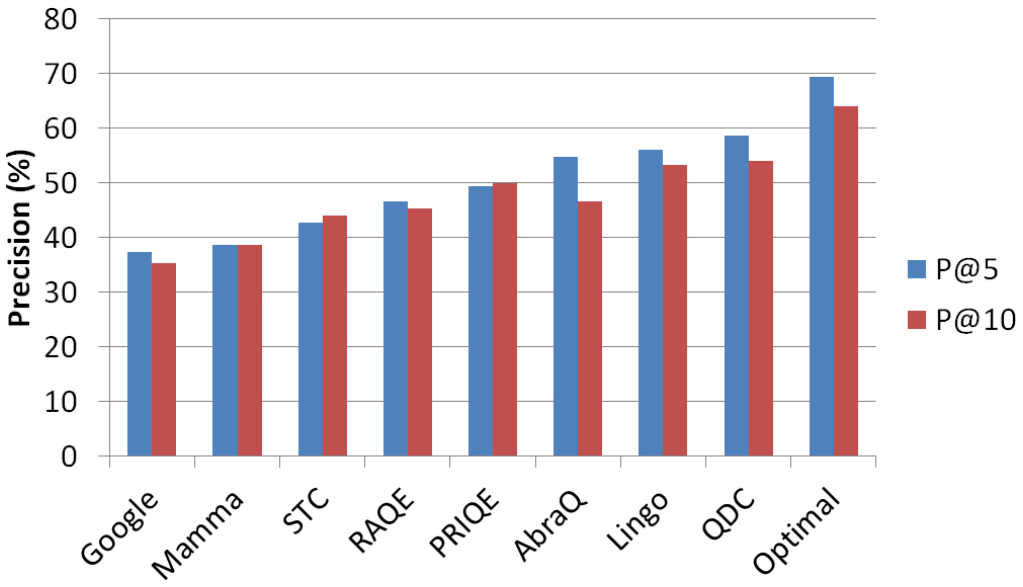

Figure 6.12: Comparing AbraQ with Interactive Query Expansion methods and Google on all queries using $P@5$ and $P@10$

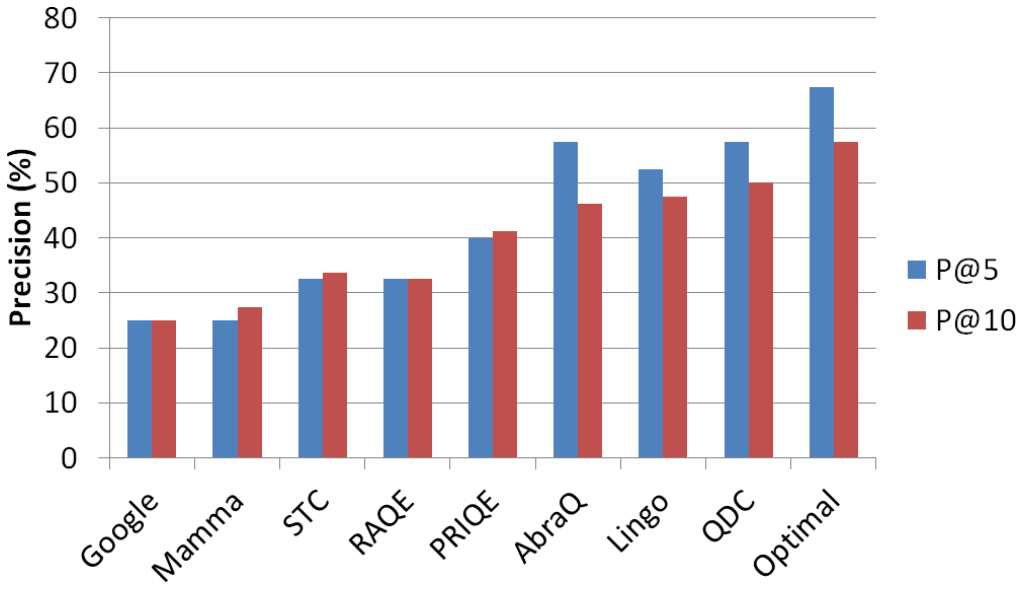

Figure 6.13: Comparing AbraQ with Interactive Query Expansion methods and Google on expanded queries using $P@5$ and $P@10$

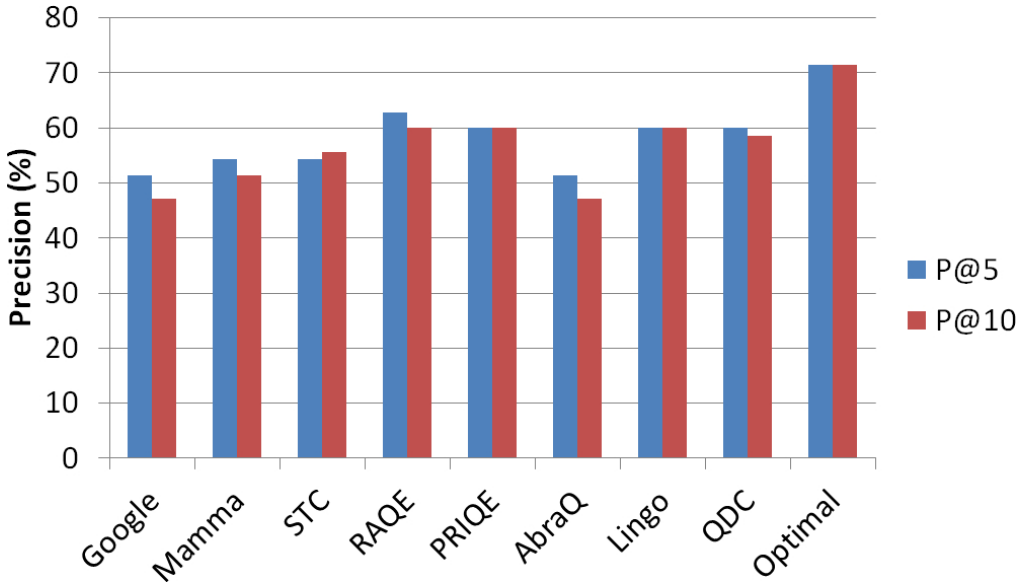

Figure 6.14: Comparing AbraQ with Interactive Query Expansion methods and Google on unexpanded queries using $P@5$ and $P@10$

The results show that AbraQ performs well against methods that exploit further user input even when they receive the best possible input. Additionally, Optimal outperforms all other methods, even on the queries AbraQ did not expand. This suggests that the QAA derived term selection method used by AbraQ is more broadly applicable than just underrepresented query aspects.

Broadly, the methods divide into three distinct groups according to their relative performance. Group 1 contains Google, Mamma, STC, and RAQE, group 2 contains PRIQE, AbraQ, Lingo, and QDC, and group 3 contains Optimal. In general, there was no significant difference between any intra-group pair and most members of groups 2 and 3 are significantly better than each member of the earlier groups. Table 6.13 shows the results of the statistical significance tests.

The results also provide further evidence that QDC (chapter 5) is a good clustering algorithm. With the exception of Optimal, QDC outperformed all other algorithms — although the improvement was only significant when compared to Google, Mamma, and STC on $P@5$ and $P@10$, and RAQE on $P@5$. These results suggest that even though QDC is designed for easy ambiguous queries, it performs at least as well as other good clustering algorithms and other IQE approaches on queries outside its purview.

6.8.4.4 Limitations

While AbraQ performed particularly well, it did make some mistakes. However, for the most part, the robustness of the QAA limited the impact of these failures.

Aspect Identification

AbraQ incorrectly identified the aspects in two queries, "New Hydroelectric Projects" and "Three Gorges Project". However, AbraQ correctly iden-

Table 6.13: The statistical significance of one algorithm's improvement over another. Each value indicates whether the row algorithm significantly outperforms the column algorithm and if so, the corresponding p-value, $p \leq 0.01$ or $p \leq 0.05$.

P@5	Google	Mamma	STC	RAQE	PRIQE	AbraQ	Lingo	QDC	Optimal
Google	No	No	No	No	No	No	No	No	No
Mamma	No	No	No	No	No	No	No	No	No
STC	No	No	No	No	No	No	No	No	No
RAQE	No	No	No	No	No	No	No	No	No
PRIQE	0.05	0.05	No	No	No	No	No	No	No
AbraQ	0.01	0.05	0.05	No	No	No	No	No	No
Lingo	0.01	0.01	0.01	No	No	No	No	No	No
QDC	0.01	0.01	0.01	0.05	No	No	No	No	No
Optimal	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.05	No

P@10	Google	Mamma	STC	RAQE	PRIQE	AbraQ	Lingo	QDC	Optimal
Google	No	No	No	No	No	No	No	No	No
Mamma	No	No	No	No	No	No	No	No	No
STC	0.01	0.05	No	No	No	No	No	No	No
RAQE	No	No	No	No	No	No	No	No	No
PRIQE	0.01	0.01	0.05	No	No	No	No	No	No
AbraQ	0.01	0.05	No	No	No	No	No	No	No
Lingo	0.01	0.01	0.05	No	No	No	No	No	No
QDC	0.01	0.01	0.05	No	No	No	No	No	No
Optimal	0.01	0.01	0.01	0.05	0.05	0.01	No	0.05	No

tified both as easy queries and did not expand them. Had the QAA vocabulary models been less robust (*e.g.* by using phrases as opposed to sets of words as discussed in section 6.7.6), AbraQ may have incorrectly expanded these queries.

Underrepresented Aspects

AbraQ failed to correct the underrepresented aspects in two queries, "Cult Lifestyles" and "International Art Crime". Although AbraQ improved performance for both queries, it did not improve performance much — AbraQ found only one or two more documents that are relevant in the top 10 than Google.

For "Cult Lifestyles", AbraQ identified the "Lifestyles" aspect as underrepresented when both aspects were actually underrepresented. The problem is that AbraQ and the QAA use relative, rather than absolute values to measure representation. Consequently, when a result set equally underrepresents all aspects, AbraQ may incorrectly consider them represented.

For "International Art Crime", AbraQ identified the "Art" aspect as underrepresented when different documents actually represented different subsets of the aspects. The problem is that AbraQ and the QAA measure the aggregate representation across the result set, rather than the representation in individual documents. Consequently, even though individual documents represent only a subset of the aspects, AbraQ may incorrectly consider all aspects represented.

Expanded and Unexpanded Queries

AbraQ expanded eight of the fifteen queries in the test set, improving the precision of seven of them within the first five results and improving all eight within the first ten results. Five of the seven unexpanded queries were those for which Google provided good initial results and seven of

the eight expanded queries were those for which Google provided poor initial results.

The two hard queries that were not expanded were "Automobile Recalls" and "Antarctica Exploration". As discussed in section 6.7.5.2, these queries have low precision because they have missing aspects (a problem that AbraQ does not attempt to address). Since these queries do not have any underrepresented aspects, it is correct for AbraQ to leave them unexpanded.

The easy query that AbraQ incorrectly expanded was "Black Bear Attacks", the problem being that it was incorrectly identified as having underrepresented aspects. However, in the first five results performance was not affected and in the first ten results performance improved very slightly. This was expected, because when aspects are already represented, AbraQ effectively selects terms that are similar to those selected by traditional AQE methods, which have negligible effect on performance as discussed in section 6.8.4.2. However, the improvement of Optimal over IQE methods for easy queries suggests that even in these failure cases, AbraQ probably selects better terms than traditional AQE methods.

On a larger set of queries with more failures, at worst, the performance in the failure cases is likely to mirror that of traditional AQE methods. Specifically, most queries would be unaffected (slightly positive or slightly negative), but a few would suffer query drift and have significantly lower performance. The key advantage of AbraQ is that the number of failures is very small because the QAA does a very good job of distinguishing hard queries (where AbraQ performs well) from easy queries (where AbraQ sometimes performs less well).

6.8.5 Future Improvements

Future research could address the limitations of AbraQ by improving how the QAA measures underrepresented aspects. By using an absolute, rather

than relative measure of representation, AbraQ could correctly identify when result sets equally underrepresent all aspects. By measuring representation at the level of individual documents, AbraQ could correctly identify when individual documents represent only a subset of aspects, even when in aggregate they cover all aspects. These improvements would also be beneficial to QAA and improve performance of other QAA derived applications such as query difficulty prediction.

AbraQ focuses on term selection and my experiments ran atop Google, but like other AQE methods, AbraQ could benefit from integrating with search engines at a lower level. Specifically, AbraQ could add additional expansion terms and use the vector space model to weight the relative importance of different query terms — these additions should make AbraQ more robust and improve performance.

6.9 Qasp

Another application of the QAA is Qasp, an interactive query expansion (IQE) method that uses the QAA to suggest refinements for hard ambiguous queries.

This section identifies how to refine hard ambiguous queries, describes the implementation of Qasp, describes some usability enhancements to Qasp, analyses its real-world efficiency, evaluates its performance, and identifies its limitations and methods of addressing them. As with QAA, there was no attempt to tune or optimize the parameters.

6.9.1 Refining Hard Ambiguous Queries

Hard ambiguous queries typically involve multiple aspects and are troublesome for users to refine for the same reasons it is hard for users to refine queries with underrepresented aspects (section 6.8.1). Existing IQE methods such as clustering can suggest useful refinements when the result set

contains some relevant documents and work well when the vocabulary of the different interpretations is distinct. However, hard ambiguous queries may not retrieve any relevant documents and the vocabulary of different interpretations can be similar.

Finding effective refinements for hard ambiguous queries is very similar to finding effective refinements for queries with underrepresented aspects. As with AbraQ, the QAA provides a mechanism for finding effective refinements that is independent of the original query's result set. This mechanism is equally suited to finding refinements for hard ambiguous queries, because it does not depend on the result set containing relevant documents, nor does it depend on the vocabulary of the different interpretations being distinct. Additionally, the quality of refinements for hard ambiguous queries can be scored in the same way as refinements for queries with underrepresented aspects, as in both cases, good refinements should produce result sets with no underrepresented aspects.

The challenge is distinguishing the refinements for one query interpretation from another. When the query is ambiguous, the refinements are only useful when at least one corresponds to the user's search goal. By finding a diverse range of good quality refinements, rather than just the set of individually best refinements, the refinements have a higher probability of being useful.

When refinements correspond to different query interpretations, their result sets will differ. Consequently, maximizing refinement diversity involves minimizing the similarity of their result sets. The simplest measure of result set similarity is the size of their intersection. However, the documents in two result sets may be completely different and yet still have very similar content, and consequently, relate to the same search goal. Therefore, a good measure of result set similarity should consider the document content and not merely the documents.

6.9.2 Algorithm

Qasp uses the QAA to identify the query aspects and their associated vocabulary. Like AbraQ, Qasp selects N candidate refinements from the aspect vocabularies, but rather than only selecting from the vocabulary of the most underrepresented aspect, it considers the higher weighted terms from each aspect's vocabulary. However, since hard ambiguity and underrepresented aspects are not mutually exclusive, the number of terms taken from each aspect, $N(a)$, is proportional to the relative likelihood that the aspect is underrepresented. Like AbraQ, the experiments use $N = 40$ candidate refinements.

$$N(a) = N \cdot \frac{\frac{1}{RAS(a,q)}}{\sum_{a' \in q} \frac{1}{RAS(a',q)}}$$

For example, if a query contained three aspects with relative aspect scores of 0.5, 0.3, and 0.2, then Qasp would select 8 terms from the most represented aspect, 13 terms from the second aspect, and 19 terms from the most underrepreseted aspect.

Qasp uses a greedy approach to select the set of refinements to show the user. At each step, Qasp selects the refinement with the highest combined score, $Score(r)$. The experiments stopped after selecting 15 refinements — in line with the AbraQ IQE experiments (section 6.8.4.3).

$$Score(r) = \frac{RS(r)}{\text{argmax}_{r' \in R}\{sim(r, r')\}}$$

where $RS(r)$ is AbraQ's refinement scoring function, R is the set of previously selected refinements, and $sim(r, r')$ measures the similarity of the result sets of the refinements.

Qasp measures the similarity of two result sets, $sim(r, r')$, using the cosine similarity [14], $cos(t_1, t_2)$, between the term vectors t_1 and t_2 for the documents in each result set.

$$cos(t_1, t_2) = \frac{t_1 \bullet t_2}{|t_1||t_2|}$$

Qasp weights the term vectors by tf-idf and removes stop words. Additionally, to focus on the differences due to query ambiguity, Qasp also removes terms that are stop words in the context of the query. Specifically, Qasp removes from the term vectors all terms that occur in more than two thirds (26) of the refinement result sets.

6.9.3 Usability Enhancements

When queries are ambiguous, users must provide further information about the intent of their search goal. IQE may elicit their intent, but when the query is non-ambiguous it is bothersome to ask the user for unnecessary information and the user is prone to making mistakes [140]. Therefore, the system should only request such information when it is essential. Qasp achieves this by measuring the homogeneity of the refinement result sets.

To measure the homogeneity of a set of refinements, Qasp clusters the refinements. It clusters them using an average-link agglomerative clustering algorithm [14] (section 5.2.3) that terminates when the cluster cohesion of a newly merged cluster would be less than the product of the cluster cohesion of its component clusters. Qasp defines cluster cohesion as the average similarity between the cluster's candidate refinements and defines the average similarity for singleton clusters as the maximum similarity between the refinement and any other candidate refinement. When there is only one cluster, Qasp considers the refinements homogeneous and when there are multiple clusters, it considers the refinements heterogeneous.

When the set of refinements is homogeneous, there is no ambiguity and no need for user input. If QAA considers all aspects represented, Qasp does not suggest or apply any refinements. When QAA considers an aspect underrepresented, Qasp automatically applies the best refinement and performs comparably to AbraQ (close to optimal). Automatically applying the expected best refinement is justified, because users typically fail to make optimal decisions and typically perform worse than AQE tech-

niques [114].

When the set of refinements is heterogeneous, there is ambiguity and user input is necessary. Therefore, Qasp presents the refinements to the user. When displayed, Qasp shows the refinements in groups based on the refinement clusters, but applies an additional threshold to stop the clustering earlier. Specifically, clustering is terminated when cluster cohesion falls below 20%. This stopping criterion keeps the refinements within a cluster closely related, so users can understand the effect of all refinements in a cluster after viewing the results of a single refinement from the cluster.

As further extensions to the clustering, Qasp orders the refinements and links them with query aspects. To help users identify the best refinements in each cluster, Qasp orders the refinements in each cluster by their expected quality, as measured by the refinement score ($RS(r)$). To help users understand the refinements, Qasp specifies the aspect from whose vocabulary model the refinement originated alongside each refinement.

6.9.4 Efficiency

The amortized cost of AbraQ was low because relatively few queries had underrepresented aspects. Qasp is more expensive because it applies to all multi-aspect queries, which constitute 65% of all queries (section 6.8.3). As with AbraQ, the correlations between the additional queries reduce their cost and the amortized average cost per query of Qasp is ten additional queries ($\frac{41}{110} \times 65\% \times 40$).

Applying clustering to enhance usability further increases the cost of Qasp and agglomerative clustering algorithms in particular are quite expensive ($O(n^2 \log n)$). However, Qasp only needs to cluster a small number of refinements (15), and consequently, clustering is a relatively small part of Qasp's total cost.

6.9.5 Evaluation

Qasp consists of two parts: the core algorithm that selects a diverse range of refinements and the usability enhancements that help users understand the effect of different refinements. I evaluated the core algorithm both quantitatively and qualitatively and evaluated the usability enhancements qualitatively.

Firstly, this section evaluates Qasp's core algorithm. Then it evaluates the effectiveness of Qasp's automatic refinement of unambiguous queries, its clustering of refinements, and its two extensions to clustering. Finally, it identifies the limitations of Qasp.

6.9.5.1 Core Algorithm

Qasp selects a diverse range of refinements, expecting this to improve performance by increasing the probability of finding refinements close to the user's search goal. The quantitative evaluation substantiates this expectation and shows that selecting a diverse range of refinements is superior to selecting the individually best refinements.

The quantitative evaluation of Qasp mirrored the approach used with the IQE methods in the evaluation of AbraQ (section 6.8.4.3). Figure 6.15 shows a comparison of Qasp's core algorithm (excluding usability enhancements) against Optimal from section 6.8.4.3, a method that merely selected the individually best refinements. Qasp improves on Optimal by a significant ($p \leq 0.01$) margin of 11% under $P@10$, and by a smaller and insignificant margin of 7% under $P@5$. Qasp was also significantly ($p \leq 0.01$) better than all other evaluated methods (Google, AbraQ, STC, Lingo, QDC, Mamma, RAQE, PRIQE) under both $P@5$ and $P@10$.

Of the fifteen queries in the test set, five were unambiguous[16] and ten were ambiguous. Qasp identified a more diverse set of refinements for

[16]These may actually be ambiguous and have more than one meaningful interpretation, but none of the expansion methods identified them.

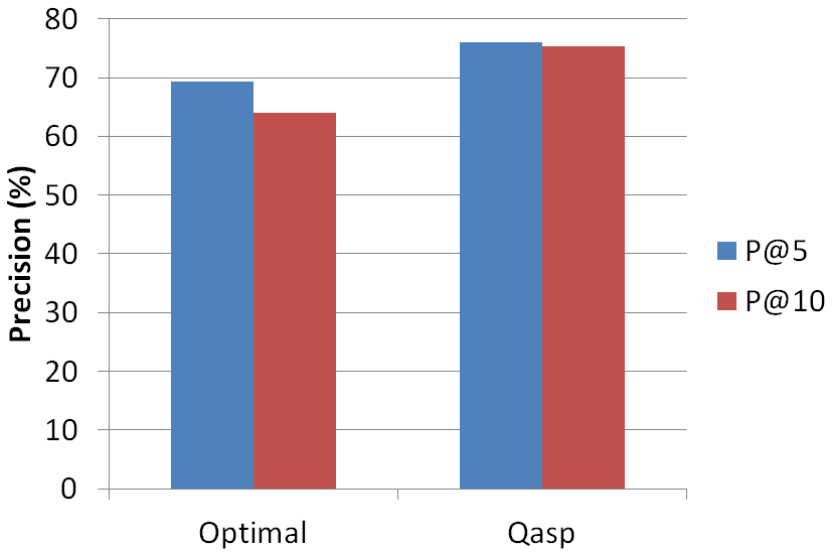

Figure 6.15: Comparing Qasp (diverse refinements) with Optimal AbraQ (individually best refinements) using $P@5$ and $P@10$

the ten ambiguous queries than any other IQE method. Table 6.14 shows the query interpretations that correspond to the refinements identified by Qasp and Lingo for the ten ambiguous queries. At best, other IQE methods performed comparably to Lingo, but generally performed worse (identified fewer distinct refinements).

Lingo and the other IQE methods typically found a set of closely related refinements that principally correspond to a single query interpretation. Specifically, Lingo found just 1 interpretation for 5 queries, 2 interpretations for 4 queries, and 3 interpretations for only 1 query. In contrast, Qasp generally found refinements covering a much wider range of interpretations. Specifically, Qasp found 1 interpretation for just 1 query, 2 interpretations for 3 queries, 3 interpretations for 5 queries, and 5 interpretations for 1 query.

The primary difference is that Lingo and the other IQE methods only found interpretations that occurred frequently in the original result set and often found only the most popular or most dominant interpretation. In

Table 6.14: Different query interpretations identified by Qasp's and Lingo's refinements. Those highlighted in red are common to both.

Query	Qasp's Interpretations
• Abuses of Email	Crime, News, Prevention
• Airport Security	Computer Device, Cost, Procedures
• Antarctica Exploration	Climate Change, Fiction, Geography, History, Space
• Automobile Recalls	Cost, Motorcycle, Vehicle
• Black Bear Attacks	Incidents, Survival
• Cult Lifestyles	Religion
• Marine Vegetation	Harvesting, Plants, Research
• Radio Waves and Brain Cancer	Mobile Phones, Treatment
• Three Gorges Project	Dam, Resettlement, Tourism
• Wildlife Extinction	Causes, Conservation

Query	Lingo's Interpretations
• Abuses of Email	Prevention
• Airport Security	Jobs, Procedures
• Antarctica Exploration	History
• Automobile Recalls	Vehicle
• Black Bear Attacks	Incidents, Survival
• Cult Lifestyles	Marketing, Religion
• Marine Vegetation	Conservation, Plants, Research
• Radio Waves and Brain Cancer	Mobile Phones
• Three Gorges Project	Dam, Resettlement
• Wildlife Extinction	News

contrast, Qasp, identified interpretations that were rare in the original result set or even completely absent from it — this is a result of Qasp using the QAA to select terms independently of the original result set. For example, only Qasp identified refinements such as "safari"[17] that correspond to a relatively rare interpretation of "Airport Security" — the security of Apple's wireless networking device named Airport.

Lingo and the other IQE methods (except Optimal) typically found many refinements that were irrelevant in the context of the query. For example, for "Black Bear Attacks", Lingo found 1 refinement related to surviving attacks, 3 refinements related to attack incidents, and 11 irrelevant refinements that primarily retrieved documents containing general information about bears where the "attacks" aspect was underrepresented. In contrast, Qasp found 5 refinements related to surviving attacks, 7 related to attack incidents, and 3 irrelevant refinements. Qasp and Optimal perform better because they consider aspect representation by using the QAA to score refinements.

Although Qasp found a much more diverse range of refinements than other IQE methods, the range is not exhaustive. However, the range does capture a useful cross-section of the most popular interpretations. For example, Climate Change was a relatively hot topic during the experiments. The most popular interpretations are captured because the vocabulary models are derived from terms in top ranked documents for each aspect, which themselves are dominated by the most popular interpretations because of the way search engines typically rank documents (section 2.5.3).

6.9.5.2 Usability - Automatic Refinement

Users have trouble choosing good refinements. Qasp can save the user effort and confusion when the query is unambiguous by automatically applying the best refinement.

[17] the name of Apple's web browser

For queries with underrepresented aspects that are unambiguous (only one cluster), Qasp saves the user effort by automatically applying the best refinement. This occurred with two queries "International Art Crime" and "Transportation Tunnel Disasters". In both cases Qasp selected good refinements that improved performance against Google. However, an even larger improvement was possible if the user selected optimally from Qasp's refinements.

For other queries that are unambiguous, Qasp avoids confusing the user by presenting no refinements and leaving the query unaltered. This occurred with three queries "Hubble Telescope Achievements", "Iran Iraq Cooperation", and "New Hydroelectric Projects". Google already had good performance for all of these queries, but the user could have improved performance if they selected optimally from Qasp's refinements.

By applying refinements automatically or choosing to show the original result set, Qasp potentially leaves performance on the table. However, the user's decision is simplified and users would probably struggle to select the best refinements in these cases. An interactive user-study would be required to determine the best action (automatically apply refinement or present refinements to user) in these cases.

6.9.5.3 Usability - Clustering

Users may perform poorly when selecting refinements because they do not understand their effect. Good refinement terms are often quite obscure and may require the user to possess substantial domain knowledge before understanding their effect. An advantage of Qasp is that it helps users interpret and choose the best refinements by grouping the refinements into clusters.

Qasp successfully identified the unambiguous queries by grouping all their refinements into one cluster — this is what occurred with the five queries in section 6.9.5.2. If Qasp presented the refinements for these unambiguous queries to users, they would benefit from seeing them grouped

into one cluster. Specifically, the user would learn that their query was unambiguous and this would help them interpret the results and refinements.

Qasp successfully identified the ambiguous queries by grouping their refinements into multiple clusters. This occurred with the ten queries shown in table 6.14.[18] Generally, each cluster of refinements corresponded to a different query interpretation from table 6.14 and the result sets of each refinement within a cluster corresponded to the same interpretation.

The primary benefit of clustering was discoverability. The clustering allows users to distinguish different categories of refinement and to use recognized associations with one refinement as a basis for exploring related refinements that they would have otherwise felt were irrelevant. For example, Qasp grouped the refinements "safari" and "news"[19] into one cluster for the query "Airport Security". Users looking for Apple's Airport device may identify with "safari" because it is the name of Apple's browser, but would normally overlook "news" and consider it an irrelevant outlier. By grouping them together, the user learns that the "news" refinement also corresponds to their search goal. Consequently, clustering is probably most useful for hard ambiguous queries where many of the best refinements initially seem irrelevant to users, because the best refinements are often not descriptive and simply co-occur with other more descriptive terms.

The most common problem with Qasp's clustering was that it stopped too soon — different clusters that relate to the same interpretation still need joining. This suggests that Qasp's stopping criterion needs improvement. However, further clustering using Qasp's agglomerative algorithm would have joined unrelated interpretations, suggesting the algorithm needs improvement and not just its stopping criterion.

[18]Qasp identified more than one cluster for "Cult Lifestyles", but only one contained relevant refinements, and consequently, table 6.14 shows only one interpretation.

[19]The Apple Airport device was prominent in news articles at the time of the experiments.

Overall, clustering appeared to be useful and its application to helping users discover and understand refinements warrants further investigation.

6.9.5.4 Usability - Clustering Extensions

Qasp extended the clustering in two ways: by ordering the refinements to help users identify the best refinements and by linking refinements with aspects to help users understand them. However, neither proved particularly useful.

Ordering the refinements within a cluster by their refinement score appeared to have no benefit. This should have been expected, because as identified in section 6.7.5.4, the QAA is good at ranking refinements at the macro-level (distinguishing good from bad), but worse at ranking refinements at a micro-scale (correctly ranking two good queries or two bad queries). In general, all the refinements within a cluster are good (or in the case of clusters for irrelevant refinements, all bad), and consequently, ranking them using their refinement score is not particularly useful.

Qasp associated each refinement with the aspect from which Qasp derived it. This was helpful in understanding why Qasp retrieved some irrelevant refinements. For example, the association with the "Recalls" aspect helped identify that the irrelevant "recipes" refinement for "Automobile Recalls" was the result of documents about food recalls being used to form the vocabulary of the "Recalls" aspect. However, the associations did not appear to aid the selection of appropriate refinements, and consequently, this feature is probably not useful for end-users.

6.9.5.5 Limitations

Like other IQE approaches, Qasp finds irrelevant refinements and its refinement descriptions are often inadequate.

Qasp finds fewer irrelevant refinements than other IQE methods, but

finds more than Optimal. However, unlike other IQE methods, it usually groups those it finds into distinct clusters, making it easier for users to identify them — other IQE methods intermingle relevant and irrelevant refinements making it hard for users to distinguish them. Qasp finds more irrelevant refinements than Optimal, because its diverse selection strategy tends to include more outliers, but this is exactly what enables Qasp to find a more diverse range of refinement interpretations. In practice, this tradeoff is suited to their respective applications, because AbraQ (the application of Optimal's refinements) needs consistency and robustness to make good refinements automatically, whereas Qasp is more tolerant of errors because the user can simply dismiss them.

To choose the best refinement, users need to understand the effect of different refinements. Qasp helps in this regard by grouping similar refinements together, but sometimes none of a cluster's refinements is self-explanatory — users have to try the refinements and examine their result sets to understand their effect. This is particularly problematic for hard ambiguous queries, where good refinements are often obscure because they merely co-occur with other query terms in relevant documents.

6.9.6 Future Improvements

While Qasp performed very well, there is room for improvement. Directions for improvement include improving the clustering algorithm (as discussed in section 6.9.5.3), limiting the inclusion of irrelevant refinements, and building useful descriptions of clusters.

Qasp finds irrelevant refinements because it seeks a diverse range of refinements. However, when clustered, many of these irrelevant refinements were in singleton clusters, whereas the relevant refinements were typically in larger clusters. Future research could investigate whether excluding singleton clusters or more generally small clusters would improve the set of refinements. It could also investigate clustering a larger number

of candidate refinements, even if not ultimately shown to users, because this would increase the evidence available for filtering.

Refinements are only useful when users understand their effect. Future research could investigate methods of building cluster descriptions to guide the user. One method would be to show a subset of the most distinct vocabulary contained in the result sets of a cluster's refinements. This could be improved by selecting terms that are closely related to the original query, and consequently, perhaps more descriptive and meaningful to users. For example, for the cluster containing "news" and "safari" for "Airport Security", the description might include terms such as "apple", "wireless", and "network" — making the effect of the cluster's refinements more obvious.

6.10 Other Applications

AbraQ and Qasp are two different applications of the QAA that help users with underrepresented query aspects and hard ambiguous queries respectively. The QAA is a very powerful technique and it has many other useful applications to web search. This section outlines four additional applications of the QAA that future research could explore.

6.10.1 Query Phrases

A very simple application of the QAA is to change the query from a set of terms, to a set of phrases, where each phrase corresponds to a query aspect. As the QAA accurately identifies query aspects, this might reduce the ambiguity of queries where the order of the query terms is important. For example, "Air Canada flights to New Zealand" would not be confused with "Air New Zealand flights to Canada".

6.10.2 Query Reduction

When a query contains too many keywords, the search engine often finds very few or even no matching documents. However, these queries actually contain more information and should theoretically produce better results. The problem is that the exact match model typically used by search engines deteriorates rapidly when presented with superfluous information and generally results in empty result sets. One solution is Automatic Query Reduction (AQR) methods that reduce the number of query terms [112].

The QAA and AbraQ provide an ideal foundation for performing AQR. Firstly, an AQR method based on the QAA would formulate a new query by finding a subset consisting of the most informative query terms that retrieve sufficiently many results. Then it would apply a variation of AbraQ to maximize the utility of the information contained in the original query. Specifically, it could use AbraQ to add a new term, but use all the original query's aspects when computing the refinement scores, rather than just the aspects in the reduced query, thereby capturing the information represented by the removed terms.

6.10.3 Search Sessions

For hard searches, users often perform a series of related queries, each of which contains useful information about their search goal. However, search engines typically ignore this useful information as they treat queries independently. By giving users the option to delineate search sessions, the refinement scores used by AbraQ, Qasp, and other applications of the QAA might improve by accounting for the aspects identified in previous queries. However, the information from a search session is much richer than that in a single query, and consequently, the QAA may benefit even more than normal from a richer aspect model.

In a temporal context, the weighting of different aspects and related

aspects takes on a greater importance. There will undoubtedly be many more related aspects than in an individual query and the relative frequency of those aspects might signify their relative importance to the user. Additionally, the user may make subtle changes to their search goal without starting a new search session, making it important to de-weight older aspects.

6.10.4 Relevance Feedback

Relevance feedback is a useful source of information and in the context of search sessions, pseudo-relevance feedback may be even more useful — specifically, methods could assume the results of previous searches were irrelevant. One application of the QAA would be to combine AbraQ or Qasp with relevance feedback and to use the feedback to guide the selection of future refinements.

6.11 Recent Developments at Google

Google has recently changed from an exact match model to a best match model as explained in section 2.6.2. An additional consequence of this change is its impact on the QAA. Like the refinement by semantically orthogonal keywords strategy that is similarly affected, the QAA relies on the addition of query terms to change the meaning of queries; with a best match model, it is possible that adding terms will not change the meaning of a query at all, because the retrieved documents are no longer required to contain the added terms. For example, "dinner lunch food c#" finds many documents that do not contain the term "c#".

While this change does not necessarily stop the QAA from being useful atop Google, it may result in the QAA producing a smaller performance benefit or improving fewer queries. Therefore, future experiments should account for this change, either by integrating with search engines such as

Google at a lower level where the QAA could have more influence on the retrieval process, or by running atop different search engines that perform an exact match and therefore use all the query terms.

Additionally, note that when I presented the QAA at KDD in 2007, there was substantial interest from all the major search engines. It is entirely possible that since running my experiments Google and other public search engines have implemented AbraQ or something similar. Consequently, if true, as search improvements are not additive, the secondary application of AbraQ should not be expected to improve performance further. This is obviously a significant drawback to using a strong constantly evolving proprietary baseline for evaluation, but it also signifies the research communities need for a strong constantly evolving versioned baseline accompanied by a large static web scale dataset.

Chapter 7

Conclusion and Future Work

Web search is a common task performed regularly by a huge number of people that has reshaped the world and how everyone accesses information. The problem is that many searches are still very hard, even for experts, and for the average user, query refinement for these searches is at best troublesome and at worst impossible. This thesis has made progress towards addressing this challenging problem by developing novel approaches that help users refine their queries and by investigating how to properly compare and evaluate different approaches.

This thesis has contributed to three main areas within the web search domain: evaluation, clustering, and query refinement. Section 7.1 presents the conclusions of chapter 3's survey of web search result evaluation methods and best practices. Section 7.2 presents the conclusions of chapter 4's investigation of web page clustering evaluation methods and its new QC4 cluster evaluation method. Section 7.3 presents the conclusions of chapter 5's investigation of web page clustering algorithms and its new Query Directed Clustering algorithm, QDC. Finally, section 7.4 presents the conclusions of chapter 6's investigation of query difficulty, its new Query Aspect Approach (QAA), and its applications of the QAA, AbraQ and Qasp. Each section also outlines some directions for future research.

7.1 Web Search Evaluation

This thesis has surveyed the research literature on approaches to web search evaluation and based on this, determined best practice for conducting an evaluation of web search results, which was subsequently used to evaluate AbraQ and Qasp in chapter 6. Specifically, chapter 3 provided guidance on selecting a methodology, selecting a corpus, obtaining a query test set, choosing a baseline, and selecting performance measures.

The thesis concluded that a non-interactive methodology using search engine corpora such as Google or Yahoo is most suited for evaluating web search results when methods are scale-dependent. It argued that while TREC datasets provide a good source of queries, the associated relevance judgments are not reusable on large collections because they are incomplete, and consequently, do not rank the relative performance of algorithms correctly and even worse, the best performing algorithms are the most negatively affected. It concluded that even small test sets are suitable for evaluation, because they are sufficient to detect large differences between algorithms and users do not notice small differences. It concluded that binary relevance judgments made by a single user are suitable for evaluation: although alternate methods changed the absolute measure of performance, they did not affect the relative performance of algorithms. It found that the commercial search engines outperformed typical TREC systems (across a wide range of queries, but not in specific instances against specific systems) and that of the commercial search engines, Google slightly outperformed its rivals. It therefore concluded that Google was the best baseline for comparison. Finally, it identified that P@5 and P@10 are good measures of performance, because most users never look beyond the first page of the result set.

In the future, a strong baseline search engine should be developed and optimized for a large static web-scale dataset. To ensure it remains relevant and reusable, the search engine should be versioned and constantly

evolved to represent the state-of-the-art.

7.2 QC4 Clustering Evaluation

Chapter 4 identified the properties of good web page clustering evaluation methods. It argued that good evaluation methods should measure both quality and coverage and measure them separately, because clustering algorithms often tradeoff between them. It argued that measures should avoid bias due to cluster or topic size, should only give perfect scores to perfect clusters, and should give the worst performance to random clusterings, singleton clusterings, and giant clusterings as they are worthless to users because they provide no added value over the original result set. It argued that the measures should account for cluster composition and segmentation because these affect the usefulness of clusters and should account for overlapping and hierarchical clusterings because documents often relate to multiple topics. It also argued that the measures should account for the limited time of users, and consequently, that there is more relative value in adding documents to small clusters than to large clusters.

Chapter 4 then showed that none of the existing standard evaluation measurements met all those criteria and that most failed to meet even half the criteria. The chapter rectified this by defining a new measurement called QC4 that allows the fair comparison of all clustering algorithms, even those that produce clusterings with vastly different characteristics (cluster granularity: coarse or fine, clustering structure: hierarchical or flat, disjoint or overlapping, and cluster size: large or small). QC4 achieved this by generalizing the gold-standard approach to use a richer ideal clustering that can describe ideal clusterings of varying characteristics and by introducing four new overall measurements that function with clusterings of different characteristics fairly in terms of cluster quality and topic coverage.

The thesis evaluated QC4 and the existing measurements in three ways:

on an extensive set of synthetic test cases, on a range of real world web clustering tasks, and on real world hierarchical clustering tasks. The synthetic test cases showed that only QC4 meets all the requirements of a good evaluation measurement. The real world web clustering tasks showed that only QC4 always made the correct conclusions regarding the relative performance of algorithms. The real world hierarchical clustering tasks showed that only QC4 and one other measure correctly accounted for hierarchical clusterings.

In the future, standard test data sets and the corresponding ideal clusterings should be constructed and then used to evaluate standard clustering algorithms to provide a baseline for comparison. As discovered while creating the ideal clusterings for this thesis, manually constructing ideal clusterings is time consuming and laborious. Researchers could investigate whether tools can assist in the construction of ideal clusterings and although the current merge-then-cluster approach is unsuitable, because it cannot produce test sets that are both realistic and accurate, it provides an obvious starting point for future research on this issue. The QC4 measurements are reasonably complex, researchers could investigate whether it is possible to create simpler measurements that satisfy all the properties identified in section 4.2. Researchers could also investigate the applicability of QC4 to other clustering domains, especially those where clusterings have different characteristics.

7.3 QDC Clustering

Chapter 5 identified the conditions under which web page clustering algorithms are effective and the problems that cause them to fail. It found that clustering methods work well on ambiguous queries for easy searches because the different query interpretations have distinct vocabularies, but that existing clustering methods do not work well on other queries because the document contents do not align with the query interpretations,

which leads to clusters that are semantically meaningless to users.

Even for easy ambiguous queries, this thesis found that existing algorithms often generate some low quality clusters that are ambiguous, overly specific, low value, or incomprehensible (semantically meaningless) because the algorithms rely too heavily on local document properties. It found that methods typically increase coverage by merging similar clusters, but current methods require a high threshold of similarity to avoid merging semantically unrelated clusters, which impairs topic coverage, and, even so, some unrelated clusters are merged lowering cluster quality. It found that algorithms generally show a fixed number of clusters, but often there are fewer topics than clusters and this can confuse users as multiple clusters relate to the same query intention or the additional clusters are semantically meaningless to users. Finally, it found that the order of documents in clusters should differ from the order of documents in the original search results, because the relevance of a document to a cluster is different from the relevance of a document to the query.

Chapter 5 then presented QDC, a new query directed web page clustering algorithm with five key innovations that addressed the problems with current algorithms by improving semantic understanding. Firstly, QDC identifies better clusters using a query directed cluster quality guide that considers the relationship between a cluster's descriptive terms and the query terms. Secondly, QDC increases the merging of semantically related clusters and decreases the merging of semantically unrelated clusters by comparing the descriptions of clusters in addition to comparing the overlap of document contents between clusters. Thirdly, QDC fixed the cluster chaining (drifting) problem using a new cluster splitting method. Fourthly, QDC chooses better clusters to show the user by improving the ESTC cluster selection heuristic to consider cluster quality and the number of clusters to select. Finally, QDC improves the clusters by ranking the documents according to cluster relevance.

The evaluation in chapter 5 showed that QDC produces clusters that

are more semantically meaningful to users and that have much higher quality and much higher coverage than state-of-the-art algorithms on both full text and snippet data sets. The evaluation in chapter 6 provided further evidence in favor of QDC and showed that it outperformed (but not always significantly) other interactive query expansion methods including clustering, web-log analysis, relevance feedback, and pseudo-relevance feedback on hard ambiguous queries, even though QDC was designed for easy ambiguous queries. The only methods it failed to outperform were those developed in this thesis specifically for hard queries using the Query Aspect Approach.

While the results are already very impressive, QDC only considers single words; STC, Lingo, and other clustering algorithms have shown that using phrase information can provide a dramatic improvement. One obvious direction for future work is to extend QDC to use phrases rather than just words. Another direction for future improvement is to consider multiple terms from the cluster descriptions when merging clusters instead of just considering the most descriptive term.

7.4 Query Aspect Approach

Chapter 6 identified that the queries for many hard searches have multiple aspects and that the harder the search, the more likely it is to involve multiple aspects. It then confirmed that the primary reason hard searches fail is underrepresented query aspects and found that of those that did not, many failed due to hard ambiguity.

Chapter 6 then presented the novel Query Aspect Approach (QAA), which reduces the distance between the query models of users and search engines by capturing the information users embed in queries that is typically ignored by search engines. Firstly, the QAA uses the order of the words in the query to identify the different query aspects by considering the relative frequency of each subsequence of words. Then it builds mod-

els of the vocabulary associated with each aspect to capture their semantics by running sub-queries for each aspect and each aspect pair to resolve ambiguities. Finally, it uses the models to judge the quality of a result set by measuring the representation of each query aspect in the result set.

Chapter 6 then showed the QAA is a powerful, practical, and general technique with many applications because it works independently of document characteristics, the frequency of the query, and is even independent of there being any relevant documents among those initially retrieved. The evaluation showed that the QAA can accurately identify query aspects and that it predicts query difficulty more accurately than the best alternative (Clarity). Specifically, the QAA successfully discriminates between easy queries and hard queries, enabling targeted application of query expansion methods. Identified applications of the QAA include automatically improving queries with underrepresented aspects (AbraQ), suggesting refinements for hard ambiguous queries (Qasp), automatically reducing query ambiguity using phrases, automatically reducing queries to combat keyword overload, leveraging user search sessions to learn more about their search goal, and making better use of relevance feedback and pseudo-relevance feedback.

Chapter 6 then presented AbraQ, an automatic query expansion method that uses the QAA to improve queries with underrepresented aspects. AbraQ identifies the query aspects and any underrepresented aspects using the QAA. Then if the original result set underrepresents any aspects, AbraQ uses the QAA to evaluate candidate refinements from the vocabulary models of the most underrepresented aspect and applies the best refinement automatically. The evaluation showed that AbraQ substantially improves the performance of queries for hard searches and that even though it was completely automatic, its performance was almost as good as the best interactive approaches with perfect user input.

Chapter 6 then presented Qasp, an interactive query expansion method that uses the QAA to help users refine hard ambiguous queries. Qasp

identifies the query aspects using the QAA and then selects a diverse set of good refinements from the aspect vocabulary models. It then clusters the refinements and if there is more than one cluster, the query is ambiguous and Qasp presents the refinements to the user. The evaluation showed that for hard ambiguous queries, selecting a diverse range of refinements was superior to selecting the individually best refinements, that Qasp significantly outperformed other interactive expansion methods, and that Qasp found both more refinements that are relevant and refinements for a much wider range of relevant search goals. The evaluation also showed that clustering refinements is a promising technique that sometimes helps users to understand the effect of refinements that would otherwise be meaningless to them, enabling users to choose useful refinements more frequently.

Future research on the QAA could go in three main directions: improving QAA, improving QAA applications, and new applications. To improve the QAA, researchers could investigate additional factors for identifying aspects, richer ways of classifying aspects, modelling the relationships between aspects, using absolute rather than relative measures of representation, and measuring the representation at the document rather than result set level. To improve Abraq, researchers could investigate lower-level integration with search engines. To improve Qasp, researchers could investigate clustering additional refinements, methods of identifying irrelevant refinements, and methods of generating cluster descriptions. Future research could also implement and investigate the outlined applications or other new applications of the QAA.

Bibliography

[1] AGICHTEIN, E., BRILL, E., AND DUMAIS, S. Improving web search ranking by incorporating user behavior information. In *The 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)* (2006), pp. 19–26.

[2] ALLAN, J. Hard track overview in trec 2003: High accuracy retrieval from documents. In *The 12th Text REtrieval Conference (TREC'03)* (2003), pp. 24–37.

[3] ALLAN, J. Hard track overview in trec 2005 high accuracy retrieval from documents. In *The 14th Text REtrieval Conference (TREC'05)* (2005).

[4] ALPERT, J., AND HAJAJ, N. We knew the web was big... `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, 2008.

[5] AMATI, G., AND VAN RIJSBERGEN, C. J. Probabilistic models of information retrieval based on measuring divergence from randomness. *ACM Transactions on Information Systems 20*, 4 (2002), 357–389.

[6] ANDREW TROTMAN, D. J. Ir evaluation using multiple assessors per topic. In *The 12th Australasian Document Computing Symposium* (2007), pp. 9–16.

[7] ARMSTRONG, T. G., MOFFAT, A., WEBBER, W., AND ZOBEL, J. Improvements that don't add up: ad-hoc retrieval results since 1998. In *The 18th ACM Conference on Information and Knowledge Management (CIKM'09)* (2009), pp. 601–610.

[8] ARUL PRAKASH ASIRVATHAM, K. K. R. Web page classification based on document structure, 2001.

[9] ASLAM, J. A., AND YILMAZ, E. A geometric interpretation and analysis of r-precision. In *The 14th ACM International Conference on Information and knowledge management (CIKM'05)* (2005), pp. 664–671.

[10] BACK, J., AND OPPENHEIM, C. A model of cognitive load for ir: implications for user relevance feedback interaction. *Information Research 6*, 2 (2001).

[11] BAILEY, P., CRASWELL, N., AND HAWKING, D. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing and Management* (2003), 853–871.

[12] BEITZEL, S. M., JENSEN, E. C., CHOWDHURY, A., FRIEDER, O., AND GROSSMAN, D. Temporal analysis of a very large topically categorized web query log. *Journal of the American Society for Information Science and Technology 58*, 2 (2007), 166–178.

[13] BENNETT, G., SCHOLER, F., AND UITDENBOGERD, A. A comparative study of probabilistic and language models for information retrieval. In *The 19th Conference on Australasian database (ADC'08)* (2007), pp. 65–74.

[14] BERKHIN, P. Survey of clustering data mining techniques. Tech. rep., Accrue Software, San Jose, CA, 2002.

[15] BIALYNICKA-BIRULA, I. Clustering web search results.

[16] BICKEL, S., AND SCHEFFER, T. Multi-view clustering. In *The 4th IEEE International Conference on Data Mining (ICDM'04)* (2004), pp. 19–26.

[17] BOLEY, D., GINI, M., GROSS, R., HAN, E.-H., KARYPIS, G., KUMAR, V., MOBASHER, B., MOORE, J., AND HASTINGS, K. Partitioning-based clustering for web document categorization. *Decision Support System - Special issue on WITS '97 27*, 3 (1999), 329–341.

[18] BRAND, M. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications 415*, 1 (2006), 20–30.

[19] BRENES, D. J., AND GAYO-AVELLO, D. Stratified analysis of aol query log. *Information Sciences 179*, 12 (2009), 1844–1858.

[20] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30* (1998), 107–117.

[21] BRODER, A. A taxonomy of web search. *ACM SIGIR Forum 36*, 2 (2002), 3–10.

[22] BRUZA, P., MCARTHUR, R., AND DENNIS, S. Interactive internet search: keyword, directory and query reformulation mechanisms compared. In *The 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)* (2000), pp. 280–287.

[23] BTTCHER, S., CLARKE, C. L. A., AND SOBOROFF, I. The trec 2006 terabyte track. In *The 15th Text REtrieval Conference (TREC 2006)* (2006).

[24] BUCKLEY, C. Why current ir engines fail. In *The 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)* (2004), pp. 584–585.

[25] BUCKLEY, C., SALTON, G., ALLAN, J., AND SINGHAL, A. Automatic query expansion using smart: Trec 3, 1994.

[26] BUCKLEY, C., DIMMICK, D., SOBOROFF, I., AND VOORHEES, E. Bias and the limits of pooling for large collections. *Information Retrieval 10*, 6 (2007), 491–508.

[27] CALVO, R. A., AND LEE, J. Coping with the news: the machine learning way. In *The 9th Australian World Wide Web Conference (AUSWEB'03)* (2003).

[28] CAO, G., SONG, D., AND BRUZA, P. Suffix tree clustering on post-retrieval documents. Tech. rep., DSTC, 2003.

[29] CARMEL, D., YOM-TOV, E., DARLOW, A., AND PELLEG, D. What makes a query difficult? In *The 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)* (2006), pp. 390–397.

[30] CARPINETO, C., DE MORI, R., ROMANO, G., AND BIGI, B. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems 19*, 1 (2001), 1–27.

[31] CHAKRABARTI, S. *Mining the Web - Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.

[32] CHEN, H., AND DUMAIS, S. Bringing order to the web: automatically categorizing search results. In *The SIGCHI Conference on Human factors in computing systems* (2000), pp. 145–152.

[33] CHENG, D., VEMPALA, S., KANNAN, R., AND WANG, G. A divide-and-merge methodology for clustering. In *The 24th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2005), pp. 196–205.

[34] CHEUNG, K.-W., AND SUN, Y. Mining web sites clusters from link topology and site hierarchy. In *The 2003 IEEE/WIC International Conference on Web Intelligence (WI'03)* (2003), p. 271.

[35] CHITU, A. Fewer shopping sites in google's results. `googlesystem.blogspot.com/2009/10/see-fewer-shopping-sites-in-googles.html`, October 2009.

[36] CHIU WONG, W. *Incremental Document Clustering for Web Page Classification*. PhD thesis, The Chinese University of Hong Kong, 2000.

[37] CHIU WONG, W., AND FU, A. Incremental document clustering for web page classification. In *IEEE 2000 International Conference on Information Society in the 21st century: emerging technologies and new challenges (IS2000)* (2000).

[38] CHUA, S., AND KULATHURAMAIYER, N. Semantic feature selection using wordnet. In *The 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)* (2004).

[39] CILIBRASI, R., AND VITANYI, P. M. B. Automatic meaning discovery using google. `www.cwi.nl/paulv/papers/amdug.pdf`, 2004.

[40] CILIBRASI, R. L., AND VITANYI, P. M. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering 19*, 3 (2007), 370–383.

[41] COCK, M. D., AND CORNELIS, C. Fuzzy rough set based web query expansion. *International workshop on Rough Sets and Soft Computing in Intelligent Agent and Web Technologies* (2005), 9–16.

[42] COLLINS-THOMPSON, K., AND BENNETT, P. N. Predicting query performance via classification. In *Advances in Information Retrieval, 32nd European Conference IR Research (ECIR'10)* (2010), pp. 140–152.

[43] COLLINS-THOMPSON, K., AND CALLAN, J. Query expansion using random walk models. In *The 14th ACM International Conference on Information and knowledge management (CIKM'05)* (2005), pp. 704–711.

[44] CRABTREE, D., GAO, X., AND ANDREAE, P. Improving web clustering by cluster selection. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)* (2005), pp. 172–178.

[45] CRABTREE, D., GAO, X., AND ANDREAE, P. Standardized evaluation method for web clustering results. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)* (2005), pp. 280–283.

[46] CRABTREE, D., ANDREAE, P., AND GAO, X. Query directed web page clustering. In *The 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)* (2006), pp. 202–210.

[47] CRASWELL, N., AND HAWKING, D. Overview of the trec-2002 web track. In *The 11th Text REtrieval Conference (TREC'02)* (2002).

[48] CRASWELL, N., AND HAWKING, D. Overview of the trec-2004 web track. In *The 13th Text REtrieval Conference (TREC'04)* (2004).

[49] CROFT, B. B., TURTLE, H. R., AND LEWIS, D. D. The use of phrases and structured queries in information retrieval. In *The 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91)* (1991), pp. 32–45.

[50] CROFT, W., AND HARPER, D. Using probabilistic models of information retrieval without relevance information. *Journal of Documentation 35*, 4 (1979), 285–295.

[51] CRONEN-TOWNSEND, S., ZHOU, Y., AND CROFT, W. B. Predicting query performance. In *The 25th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval (SIGIR'02) (2002), pp. 299–306.

[52] CUI, H., WEN, J.-R., NIE, J.-Y., AND MA, W.-Y. Probabilistic query expansion using query logs. In *The 11th International Conference on World Wide Web (WWW'02)* (2002), pp. 325–332.

[53] DEAN, J., AND GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113.

[54] DEERWESTER, S., DUMAIS, S., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[55] DENNIS, S., MCARTHUR, R., AND BRUZA, P. Searching the world wide web made easy? the cognitive load imposed by query refinement mechanisms. *The third Australian document computing symposium (ADCS'98)* (1998), 65–71.

[56] DENNIS, S., BRUZA, P., AND MCARTHUR, R. Web searching: A process-oriented experimental study of three interactive search paradigms. *JASIST 53*, 2 (2002), 120–133.

[57] DHILLON, I. S., FAN, J., AND GUAN, Y. Efficient clustering of very large document collections. In *Data Mining for Scientific and Engineering Applications*, R. Grossman, G. Kamath, and R. Naburu, Eds. Kluwer Academic Publishers, 2001.

[58] DING, C., HE, X., HUSBANDS, P., ZHA, H., AND SIMON, H. Pagerank, hits and a unified framework for link analysis. Tech. Rep. 49372, LBNL, 2002.

[59] DRUCKER, H., SHAHRARAY, B., AND GIBBON, D. Support vector machines: relevance feedback and information retrieval. *Information Processing and Management 38*, 3 (2002), 305–323.

[60] DUMAIS, S., BANKO, M., BRILL, E., LIN, J., AND NG, A. Web question answering: is more always better? In *The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02)* (2002), pp. 291–298.

[61] EASTMAN, C. M., AND JANSEN, B. J. The appropriate (and inappropriate) use of query operators and their effect on web search results. *American Society for Information Science and Technology 41*, 1 (2004), 274–279.

[62] EICK, C. F., ZEIDAT, N., AND ZHAO, Z. Supervised clustering algorithms and benefits. In *The 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)* (2004), pp. 774–776.

[63] FERGUSON, P., SMEATON, A. F., AND WILKINS, P. Dublin city university at the trec 2006 terabyte track. In *The 15th Text REtrieval Conference (TREC'06)* (2006).

[64] FERRAGINA, P., AND GULLI, A. A personalized search engine based on web-snippet hierarchical clustering. In *Special interest tracks and posters of the 14th International Conference on World Wide Web* (2005), pp. 801–810.

[65] FOWLKES, E. B., AND MALLOWS, C. L. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association 78*, 383 (1983), 553–569.

[66] FOX, E. A., NEVES, F. D., YU, X., SHEN, R., KIM, S., AND FAN, W. Exploring the computing literature with visualization and stepping stones & pathways. *Communications of the ACM 49*, 4 (2006), 52–58.

[67] FRANZ, A., AND BRANTS, T. All our n-gram are belong to you. `http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html`, 2006.

[68] FUNG, B. C., WANG, K., AND ESTER, M. Hierarchical document clustering using frequent itemsets. In *The SIAM International Conference on Data Mining* (2003).

[69] GAYO-AVELLO, D. A survey on session detection methods in query logs and a proposal for future evaluation. *Information Sciences 179*, 12 (2009), 1822–1843.

[70] GELGI, F., DAVULCU, H., AND VADREVU, S. Term ranking for clustering web search results. In *The 10th International Workshop on the Web and Databases (WebDB'07)* (2007).

[71] GREENE, K. Peter norvig on 'the future of search'. `http://www.technologyreview.com/Biztech/19050/`, July 2007.

[72] GUHA, S., RASTOGI, R., AND SHIM, K. Cure: An efficient clustering algorithm for large databases. In *The 1998 ACM SIGMOD International Conference on Management of data* (1998), pp. 73–84.

[73] GURRIN, C., AND SMEATON, A. F. Improving the evaluation of web search systems. In *Advances in Information Retrieval: 25th European Conference on IR Research (ECIR'03)* (2003), pp. 25–40.

[74] GURRIN, C., AND SMEATON, A. F. Replicating web structure in small-scale test collections. *Information Retrieval 7*, 3-4 (2004), 239–263.

[75] HALKIDI, M., BATISTAKIS, Y., AND VAZIRGIANNIS, M. On clustering validation techniques. *Journal of Intelligent Information Systems 17*, 2-3 (2001), 107–145.

[76] HALKIDI, M., NGUYEN, B., VARLAMIS, I., AND VAZIRGIANNIS, M. Thesus: Organizing web document collections based on link semantics. *The International Journal on Very Large Data Bases 12*, 4 (2003), 320–332.

[77] HAMMOUDA, K. M., AND KAMEL, M. S. Incremental document clustering using cluster similarity histograms. In *The 2003 IEEE/WIC International Conference on Web Intelligence (WI'03)* (2003), p. 597.

[78] HARMAN, D. *Relevance feedback and other query modification techniques*. Englewood Cliffs: Prentice Hall, 1992, ch. 11, pp. 241–263.

[79] HARTUV, E., AND SHAMIR, R. A clustering algorithm based on graph connectivity. *Information Processing Letters 76*, 4–6 (2000), 175–181.

[80] HAVELIWALA, T. H., GIONIS, A., KLEIN, D., AND INDYK, P. Evaluating strategies for similarity search on the web. In *The 11th International Conference on World Wide Web (WWW'02)* (2002), pp. 432–442.

[81] HAWKING, D., AND CRASWELL, N. Very large scale retrieval and web search. *TREC: Experiment and Evaluation in Information Retrieval* (2005).

[82] HAWKING, D., CRASWELL, N., THISTLEWAITE, P., AND HARMAN, D. Results and challenges in Web search evaluation. *Computer Networks 31*, 11–16 (1999), 1321–1330.

[83] HAWKING, D., CRASWELL, N., BAILEY, P., AND GRIFFITHS, K. Measuring search engine quality. *Information Retrieval 4*, 1 (2001), 33–59.

[84] HE, B., AND OUNIS, I. Inferring query performance using pre-retrieval predictors. In *Symposium on String Processing and Information Retrieval* (2004), Springer Verlag, pp. 43–54.

[85] HE, D., AND GÖKER, A. Detecting session boundaries from web user logs. In *The BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research* (2000), pp. 57–66.

[86] HE, X., DING, C. H., ZHA, H., AND SIMON, H. D. Automatic topic identification using webpage clustering. In *The 1st IEEE International Conference on Data Mining (ICDM'01)* (2001), p. 195.

[87] HOU, J., AND ZHANG, Y. Utilizing hyperlink transitivity to improve web page clustering. In *The 14th Australasian database Conference on Database technologies* (2003), vol. 17, pp. 49–57.

[88] HSU, M.-H., TSAI, M.-F., AND CHEN, H.-H. Query expansion with conceptnet and wordnet: An intrinsic comparison. In *AIRS* (2006), pp. 1–13.

[89] HUANG, J., AND EFTHIMIADIS, E. N. Analyzing and evaluating query reformulation strategies in web search logs. In *The 18th ACM Conference on Information and Knowledge Management (CIKM'09)* (2009), pp. 77–86.

[90] IDE, E. New experiments in relevance feedback. In *The SMART Retrieval System: Experiments in Automatic Document Processing* (1971), G. Salton, Ed., Prentice Hall, pp. 337–354.

[91] IPROSPECT. iprospect search engine user behavior study. `www.iprospect.com/premiumPDFs/WhitePaper_2006_SearchEngineUserBehavior/`, 2006.

[92] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys (CSUR) 31*, 3 (1999), 264–323.

[93] JANSEN, B. J., SPINK, A., BATEMAN, J., AND SARACEVIC, T. Real life information retrieval: a study of user queries on the web. *SIGIR Forum 32*, 1 (1998), 5–17.

[94] JANSEN, B. J., SPINK, A., AND PEDERSEN, J. A temporal comparison of altavista web searching. *Journal of the American Society for Information Science and Technology 56*, 6 (2005), 559–570.

[95] JANSEN, B. J., BOOTH, D. L., AND SPINK, A. Determining the informational, navigational, and transactional intent of web queries. *Information Processing and Management 44*, 3 (2008), 1251–1266.

[96] JOACHIMS, T., AND RADLINSKI, F. Search engines that learn from implicit feedback. *Computer 40*, 8 (2007), 34–40.

[97] JOACHIMS, T., GRANKA, L., PAN, B., HEMBROOKE, H., RADLINSKI, F., AND GAY, G. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems 25*, 2 (2007), 7.

[98] JOHO, H., AND SANDERSON, M. Document frequency and term specificity. In *The 8th Recherche d'Information Assiste par Ordinateur Conference (RIAO'07)* (2007).

[99] JOHO, H., AND SANDERSON, M. The spirit collection: an overview of a large web collection. *SIGIR Forum 38*, 2 (2004), 57–61.

[100] JONES, K. S., AND VAN RIJSBERGEN, C. J. Report on the need for and provision of an ideal information retrieval test collection. british library research and development report 5266. Tech. rep., Computer Library, University of Cambridge, 1975.

[101] KHAN, M. S., AND KHOR, S. Enhanced web document retrieval using automatic query expansion. *Journal of the American Society for Information Science and Technology 55*, 1 (2004), 29–40.

[102] KHOO, C. S. G., AND POO, D. C. C. An expert system approach to online catalog subject searching. *Information Processing and Management 30*, 2 (1994), 223–238.

[103] KOHAVI, R., HENNE, R. M., AND SOMMERFIELD, D. Practical guide to controlled experiments on the web: listen to your customers not

to the hippo. In *The 13th ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD'07)* (2007), pp. 959–967.

[104] KONTOSTATHIS, A., POTTENGER, W. M., AND DAVISON, B. D. Identification of critical values in latent semantic indexing. In *Foundations of Data Mining and Knowledge Discovery* (2005), Springer-Verlag, pp. 333–346.

[105] KUMMAMURU, K., LOTLIKAR, R., ROY, S., SINGAL, K., AND KRISHNAPURAM, R. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *The 13th International Conference on World Wide Web (WWW'04)* (2004), pp. 658–665.

[106] KURALENOK, I. E., AND NEKRESTYANOV, I. S. Evaluation of text retrieval systems. *Programming and Computer Software 28*, 4 (2002), 226–242.

[107] LANCASTER, F. W. *Information Retrieval Systems: Characteristics, Testing and Evaluation*. Wiley, New York, 1968.

[108] LEWIS, D. D., YANG, Y., ROSE, T. G., AND LI, F. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research 5* (2004), 361–397.

[109] LIPSMAN, A. Global search market growth of 46 percent in 2009. `http://www.comscore.com/Press_Events/Press_Releases/2010/1/Global_Search_Market_Grows_46_Percent_in_2009`, 2010.

[110] LIU, B. Personal evaluations of search engines: Google, yahoo!, and msn. `http://www.cs.uic.edu/%7Eliub/searchEval/SearchEngineEvaluation.htm`, 2006.

[111] LIU, B. Personal evaluations of search engines: Google, yahoo!, and msn - comparison of evaluation results of fall 2006 and of fall 2007, 2006.

[112] LU, X. A., AND KEEFER, R. B. Query expansion/reduction and its impact on retrieval effectiveness. In *The 3rd Text Retrieval Conference (TREC'95)* (1995), pp. 231–239.

[113] MACKAY, D. J. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[114] MAGENNIS, M., AND VAN RIJSBERGEN, C. J. The potential and actual effectiveness of interactive query expansion. In *The 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)* (1997), pp. 324–332.

[115] MAIL, D. Google nets more than half of the world's web searches. `http://www.dailymail.co.uk/pages/live/articles/technology/technology.html?in_article_id=487100&in_page_id=1965&ito=1490`, October 2007.

[116] MALDE, K., COWARD, E., AND JONASSEN, I. Fast sequence clustering using a suffix array algorithm. *BIOINFORMATICS 19*, 10 (2003), 1221–1226.

[117] Mamma.com: www.mamma.com, 2007.

[118] MANDHANI, B., JOSHI, S., AND KUMMAMURU, K. A matrix density based algorithm to hierarchically co-cluster documents and words. In *The 12th International Conference on World Wide Web (WWW'03)* (2003), pp. 511–518.

[119] MANGLANO, V., BEAULIEU, M., AND ROBERTSON, S. E. Evaluation of interfaces for irs: Modelling end-user searching behaviour. In *BCS-IRSG Annual Colloquium on IR Research* (1998).

[120] MANNING, C. D., RAGHAVAN, P., AND SCHTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[121] MEILA, M. Comparing clusterings. Tech. Rep. 418, Department of Statistics, University of Washington, 2002.

[122] MENCZER, F. Combining link and content analysis to estimate semantic similarity. In *The 13th International Conference on World Wide Web - Alternate track papers and posters (WWW'04)* (2004), pp. 452–453.

[123] MENCZER, F. Lexical and semantic clustering by web links. *Journal of the American Society for Information Science and Technology 55*, 14 (2004), 1261–1269.

[124] MENDES RODRIGUES, E., AND SACKS, L. A scalable hierarchical fuzzy clustering algorithm for text mining. In *The 5th International Conference on Recent Advances in Soft Computing* (2004), pp. 269–274.

[125] MILLER, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review 63*, 2 (1956), 81–79.

[126] MILLIGAN, G. W., SOON, S. C., AND SOKOL, L. M. The effect of cluster size, dimensionality and the number of clusters on recovery of true cluster structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 1 (1983), 40–47.

[127] MIRKIN, B. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[128] MITRA, M., SINGHAL, A., AND BUCKLEY, C. Improving automatic query expansion. In *The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)* (1998), pp. 206–214.

[129] MODHA, D. S., AND SPANGLER, W. S. Feature weighting in k-means clustering. *Machine Learning 52*, 3 (2003), 217–237.

[130] MOTHE, J., AND TANGUY, L. Linguistic features to predict query difficulty. In *ACM SIGIR Workshop on Predicting Query Difficulty* (2005).

[131] MUH-CHYUN, T., AND YING, S. Evaluation of web-based search engines using user-effort measures. *LIBRES: Library and Information Science Research Electronic Journal 13*, 2 (2003).

[132] MURAYAMA, N., SAITO, S., AND OKUMURA, M. Are web pages characterized by color? In *The 13th International Conference on World Wide Web - Alternate track papers and posters (WWW'04)* (2004), pp. 248–249.

[133] NEVES, F. A. D., FOX, E. A., AND YU, X. Connecting topics in document collections with stepping stones and pathways. In *The 14th ACM International Conference on Information and knowledge management (CIKM'05)* (2005), pp. 91–98.

[134] NORVIG, P. Artificial intelligence as the future of search. `www.youtube.com/watch?v=0zRUozxcOxo`, November 2007.

[135] OSIŃSKI, S., AND WEISS, D. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In *The International IIS: Intelligent Information Processing and Web Mining Conference* (2004), Advances in Soft Computing, Springer, pp. 369–378.

[136] OSINSKI, S., AND WEISS, D. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems 20*, 3 (2005), 48–54.

[137] OSIŃSKI, S., STEFANOWSKI, J., AND WEISS, D. Lingo: Search results clustering algorithm based on singular value decomposition. In *The International IIS: Intelligent Information Processing and Web Mining*

Conference (2004), Advances in Soft Computing, Springer, pp. 359–368.

[138] PANDIA. Google: one million servers and counting. `http://www.pandia.com/sew/481-gartner.html`, 2007.

[139] PASS, G., CHOWDHURY, A., AND TORGESON, C. A picture of search. In *The 1st International Conference on Scalable Information Systems* (2006).

[140] PECHENIZKIY, M., TSYMBAL, A., AND PUURONEN, S. Pca-based feature transformation for classification: issues in medical diagnostics. In *The 17th IEEE Symposium on Computer-Based Medical Systems* (2004), pp. 535–540.

[141] PORTER, M. F. An algorithm for suffix stripping. *Program 14*, 3 (1980), 130–137.

[142] RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 336 (1971), 846–850.

[143] RANKSTAT.COM. Most people use 2 word phrases in search engines. `http://www.rankstat.com/html/en/seo-news1-most-people-use-2-word-phrases-in-search-engines.html`, 2006.

[144] REID, J. A task-oriented non-interactive evaluation methodology for information retrieval systems. *Information Retrieval 2*, 1 (2000), 115–129.

[145] RICCA, F., TONELLA, P., GIRARDI, C., AND PIANTA, E. An empirical study on keyword-based web site clustering. In *The 12th IEEE International Workshop on Program Comprehension (IWPC'04)* (2004), p. 204.

[146] ROBERTSON, S. E., AND JONES, K. S. Relevance weighting of search terms. *Journal of the American Society for Information Science 27*, 3 (1976), 129–146.

[147] ROBERTSON, S. E., WALKER, S., HANCOCK-BEAULIEU, M., GULL, A., AND LAU, M. Okapi at trec-3. In *The 13th Text REtrieval Conference (TREC'92)* (1992), pp. 21–30.

[148] ROCCHIO, J. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing* (1971), G. Salton, Ed., Prentice Hall, pp. 313–323.

[149] ROSE, D., AND LEVINSON, D. Understanding user goals in web search. In *The 13th International Conference on World Wide Web (WWW'04)* (2004), pp. 13–19.

[150] RUTHVEN, I., AND LALMAS, M. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review 19*, 2 (2003), 95–145.

[151] SAHOO, N., CALLAN, J., KRISHNAN, R., DUNCAN, G., AND PADMAN, R. Incremental hierarchical clustering of text documents. In *The 15th ACM International Conference on Information and knowledge management (CIKM'06)* (2006), pp. 357–366.

[152] SALTON, G., WONG, A., AND YANG, C. S. A vector space model for automatic indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

[153] SALTON, G., FOX, E. A., AND WU, H. Extended boolean information retrieval. *Communications of the ACM 26*, 11 (1983), 1022–1036.

[154] SCHENKER, A., LAST, M., AND KANDEL, A. A term-based algorithm for hierarchical clustering of web documents. In *Proceedings of IFSA / NAFIPS* (2001), pp. 3076–3081.

[155] SCHENKER, A., LAST, M., BUNKE, H., AND KANDEL, A. A comparison of two novel algorithms for clustering web documents. In *The 2nd International Workshop on Web Document Analysis (WDA'03)* (2003), pp. 71–74.

[156] SHIPENG, Y., CAI, D., WEN, J., AND MA, W. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *The 12th International World Wide Web Conference* (2003), pp. 11–18.

[157] SIHVONEN, A., AND VAKKARI, P. Subject knowledge, thesaurus-assisted query expansion and search success. In *The RIAO 2004 Conference* (2004), pp. 393–404.

[158] SILVERSTEIN, C., HENZINGER, M., MARAIS, H., AND MORICZ, M. Analysis of a very large web search engine query log. Tech. Rep. 1998-014, Digital SRC, 1998.

[159] SINGHAL, A., AND KASZKIEL, M. A case study in web search using TREC algorithms. In *The 10th International Conference on World Wide Web (WWW'01)* (2001), pp. 708–716.

[160] SMYTH, B. A community-based approach to personalizing web search. *Computer 40*, 8 (2007), 42–50.

[161] SOBOROFF, I. A comparison of pooled and sampled relevance judgments in the trec 2006 terabyte track. In *The 1st International Workshop on Evaluating Information Access (EVIA'07)* (2007).

[162] SPINK, A., KOSHMAN, S., PARK, M., FIELD, C., AND JANSEN, B. J. Multitasking web search on vivisimo.com. In *International Conference on Information Technology: Coding and Computing (ITCC'05)* (2005), vol. II, pp. 486–490.

[163] STATS, I. W. Internet usage statistics: The internet big picture. `http://www.internetworldstats.com/stats.htm`, 2010.

[164] STEFANOWSKI, J., AND WEISS, D. Carrot^2 and language properties in web search results clustering. In *The 1st International Atlantic Web Intelligence Conference (AWIC'03)* (2003), vol. 2663 of *Lecture Notes in Computer Science*, pp. 240–249.

[165] STEIN, B., AND ZU EISSEN, S. M. Automatic document categorization: Interpreting the performance of clustering algorithms. In *The 26th German Conference on Artificial Intelligence (KI'03)* (2003), vol. 2821 of *LNCS of Lecture Notes in Computer Science*, pp. 254–266.

[166] STEINBACH, M., KARYPIS, G., AND KUMAR, V. A comparison of document clustering techniques. In *KDD Workshop on Text Mining* (2000).

[167] STREHL, A. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, Faculty of the Graduate School of The University of Texas at Austin, 2002.

[168] STREHL, A., AND GHOSH, J. A scalable approach to balanced, high-dimensional clustering of market-baskets. *The 7th International Conference on High Performance Computing 1970* (2000), 525–536.

[169] STREHL, A., GHOSH, J., AND MOONEY, R. Impact of similarity measures on web-page clustering. In *The 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI'00)* (2000), pp. 58–64.

[170] STROHMAIER, M., PRETTENHOFER, P., AND LUX, M. Different degrees of explicitness in intentional artifacts - studying user goals in a large search query log. In *The Workshop on Commonsense Knowledge and Goal Oriented Interfaces (CSKGOI'08)* (2008).

[171] SU, Z., YANG, Q., ZHANG, H., XU, X., AND HU, Y. Correlation-based document clustering using web logs. *The 34th Annual Hawaii*

International Conference on System Sciences (HICSS-34) 5, 34 (2001), 5022.

[172] SULLIVAN, D. More spotting googles related searches at bottom of page. `searchengineland.com/more-spotting-googles-related-searches-at-bottom-of-page-10261`, January 2007.

[173] SUROWIECKI, J. *The Wisdom of Crowds*. Anchor, 2005.

[174] TANG, X. A critical analysis of clustering for web search. Tech. rep., University of Regina, Regina, 2002.

[175] THEOBALD, M., SCHENKEL, R., AND WEIKUM, G. Efficient and self-tuning incremental query expansion for top-k query processing. In *The 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)* (2005), pp. 242–249.

[176] TOMLINSON, S., OARD, D. W., BARON, J. R., AND THOMPSON, P. Overview of the trec 2007 legal track. In *The 16th Text Retrieval Conference (TREC'07)* (2007).

[177] TONELLA, P., RICCA, F., PIANTA, E., GIRARDI, C., LUCCA, G. D., FASOLINO, A. R., AND TRAMONTANA, P. Evaluation methods for web application clustering. In *The 5th International Workshop on Web Site Evolution* (2003), p. 33.

[178] TONELLA, P., RICCA, F., PIANTA, E., AND GIRARDI, C. Using keyword extraction for web site clustering. In *The 5th International Workshop on Web Site Evolution* (2003), p. 41.

[179] VAN RIJSBERGEN, C. J. *Information Retrieval*, 2 ed. Buterworths, London, 1979.

[180] VARLAMIS, I., VAZIRGIANNIS, M., HALKIDI, M., AND NGUYEN, B. Thesus, a closer view on web content management enhanced with link semantics. *IEEE Transactions on Knowledge and Data Engineering 16*, 6 (2004), 685–700.

[181] VÉLEZ, B., WEISS, R., SHELDON, M. A., AND GIFFORD, D. K. Fast and effective query refinement. In *The 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)* (1997), pp. 6–15.

[182] VICEDO, J. L., AND FERRÁNDEZ, A. Importance of pronominal anaphora resolution in question answering systems. In *The 38th Annual Meeting on Association for Computational Linguistics (ACL'00)* (2000), pp. 555–562.

[183] VINAY, V., WOOD, K. R., MILIC-FRAYLING, N., AND COX, I. J. Comparing relevance feedback algorithms for web search. In *The 14th International Conference on World Wide Web (WWW'05)* (2005), pp. 1052–1053.

[184] VOORHEES, E. M. Overview of the trec 2005 robust retrieval track. In *The 14th Text Retrieval Conference (TREC'05)* (2005).

[185] VOORHEES, E. M. Variations in relevance judgments and the measurement of retrieval effectiveness. In *The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)* (1998), pp. 315–323.

[186] VOORHEES, E. M. Overview of the trec 2004 robust retrieval track. In *The 13th Text REtrieval Conference (TREC'04)* (2004).

[187] WANG, Y., AND KITSUREGAWA, M. An link-contents coupled clustering for web search results.

[188] WANG, Y., AND KITSUREGAWA, M. On combining link and contents information for web page clustering. In *The 13th International Conference on Database and Expert Systems Applications (DEXA'02)* (2002), pp. 902–913.

[189] WANG, Y., AND KITSUREGAWA, M. Combining link and contents in clustering web search results to improve information interpretation. In *Proceedings of 2002 Data Engineering Workshop (DEWS'02)* (2002), pp. C4–2.

[190] WANG, Y., AND KITSUREGAWA, M. Use link-based clustering to improve web search results. In *The 2nd International Conference on Web Information System Engineering (WISE'2001)* (2002), pp. 115–124.

[191] WEISS, R., VELEZ, B., SHELDON, M. A., NEMPREMPRE, C., SZILAGYI, P., DUDA, A., AND GIFFORD, D. K. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *The 7th ACM Conference on Hypertext* (1996).

[192] WHITE, R. W., BILENKO, M., AND CUCERZAN, S. Studying the use of popular destinations to enhance web search interaction. In *The 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)* (2007), pp. 159–166.

[193] WIKIMEDIA. Wikipedia statistics english. `http://stats.wikimedia.org/EN/TablesWikipediaEN.htm`, 2010.

[194] WIKIPEDIA. Wikipedia. `en.wikipedia.org`, 2007.

[195] XU, J., AND CROFT, W. B. Query expansion using local and global document analysis. In *The 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)* (1996), pp. 4–11.

[196] YAHOO. Yahoo random link. `http://stats.wikimedia.org/EN/TablesWikipediaEN.htm`.

[197] YANG, K. *Combining Text-, Link-, and Classification-based Retrieval Methods to Enhance Information Discovery on the Web*. PhD thesis, The University of North Carolina at Chapel Hill, 2002.

[198] YAO, Z., AND CHOI, B. Bidirectional hierarchical clustering for web mining. In *The 2003 IEEE/WIC International Conference on Web Intelligence (WI'03)* (2003), p. 620.

[199] YOM-TOV, E., FINE, S., CARMEL, D., AND DARLOW, A. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *The 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)* (2005), pp. 512–519.

[200] YU, X., NEVES, F. A. D., AND FOX, E. A. Hard queries can be addressed with query splitting plus stepping stones and pathways. *IEEE Data Engineering Bulletin 28*, 4 (2005), 29–38.

[201] ZAMIR, O., AND ETZIONI, O. Web document clustering: A feasibility demonstration. In *The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)* (1998), pp. 46–54.

[202] ZAMIR, O., AND ETZIONI, O. Grouper: a dynamic clustering interface to web search results. *Computer Networks 31*, 11–16 (1999), 1361–1374.

[203] ZAMIR, O. E. *Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results*. PhD thesis, University of Washington, 1999.

[204] ZENG, H.-J., HE, Q.-C., CHEN, Z., MA, W.-Y., AND MA, J. Learning to cluster web search results. In *The 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)* (2004), pp. 210–217.

[205] ZHANG, D., AND DONG, Y. Semantic, hierarchical, online clustering of web search results. In *The 6th Asia Pacific Web Conference (APWEB'04)* (2004).

[206] ZHANG, J., SUN, L., LV, Y., AND ZHANG, W. Relevance feedback by exploring the different feedback source and collection structure. In *The 14th Text REtrieval Conference (TREC'05)* (2005).

[207] ZHAO, Y., AND KARYPIS, G. Criterion functions for document clustering: Experiments and analysis. Tech. rep., Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.

[208] ZHAO, Y., AND KARYPIS, G. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning 55*, 3 (2004), 311–331.

[209] ZHAO, Y., KARYPIS, G., AND FAYYAD, U. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery 10*, 2 (2005), 141–168.

[210] ZOBEL, J. How reliable are the results of large-scale information retrieval experiments? In *The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)* (1998), pp. 307–314.

[211] ZU EIEN, S. M., AND STEIN, B. Analysis of clustering algorithms for web-based search. *Practical Aspects of Knowledge Management 2569* (2002), 168–178.

[212] ZU EISSEN, S. M., STEIN, B., AND POTTHAST, M. The suffix tree document model revisited. In *The 5th International Conference on Knowledge Management (I-KNOW'05)* (2005).