Cluster-based collection selection in uncooperative distributed information retrieval

Bertold van Voorst
MSc. Thesis
July 7, 2010

University of Twente
Department of Computer Science

Graduation committee:
Dr. Ir. Djoerd Hiemstra
Ir. Almer Tigelaar
Ir. Dolf Trieschnigg

Abstract

Background The focus of this research is collection selection for distributed information retrieval. The collection descriptions that are necessary for selecting the most relevant collections are often created from information gathered by random sampling. Collection selection based on an incomplete index constructed by using random sampling instead of a full index leads to inferior results.
Contributions In this research we propose to use collection clustering to compensate for the incompleteness of the indexes. When collection clustering is used we do not only select the collections that are considered relevant based on their collection descriptions, but also collections that have similar content in their indexes. Most existing cluster algorithms require the specification of the number of clusters prior to execution. We describe a new clustering algorithm that allows us to specify the sizes of the produced clusters instead of the number of clusters.
Conclusions Our experiments show that that collection clustering can indeed improve the performance of distributed information retrieval systems that use random sampling. There is not much difference in retrieval performance between our clustering algorithm and the well-known k-means algorithm. We suggest to use the algorithm we proposed because it is more scalable.

Acknowledgments

I would like to thank my supervisors for their guidance during this research. Also I would like to thank my fellow students with whom I spent a lot of time at the university, and were there to help out and discuss various topics. Lastly, thanks to my family and close friends for their support.

Contents

Abstract **i**

Acknowledgments **iii**

1 Introduction **1**
1.1 Information retrieval . . . 1
1.2 Centralized search . . . 2
1.3 Distributed information retrieval . . . 2
1.4 Collection selection based on content similarity . . . 3
1.5 Research questions . . . 5
1.6 Thesis outline . . . 7

2 Literature **9**
2.1 Zipf's law . . . 9
2.2 Query-based sampling . . . 10
2.3 Cluster hypothesis . . . 11
2.4 Collection selection algorithms . . . 12
2.4.1 GlOSS . . . 12
2.4.2 Cue Validity Variance . . . 13
2.4.3 CORI . . . 14
2.4.4 Indri . . . 15
2.4.5 Discussion . . . 16
2.5 Clustering . . . 16
2.5.1 Clustering types . . . 17
2.5.2 K-means algorithm . . . 17
2.5.3 Bisecting k-means algorithm . . . 19
2.6 Cluster-based retrieval . . . 20
2.7 Summary . . . 21

3 Research **23**
3.1 The WT10g corpus . . . 23
3.2 Random sampling . . . 25
3.3 Clustering . . . 26
3.3.1 Reducing the calculations complexity . . . 26

3.3.2 K-means 26
3.3.3 Bisecting k-means 27
3.3.4 Indexing clusters 28
3.4 Ranking collections 28
3.5 Evaluation 29
3.6 Summary 31

4 Results 33
4.1 Query-based sampling 33
4.2 Cluster sizes and the number of clusters 34
4.3 Cluster quality 39
4.4 Comparing cluster algorithms 40
4.5 Query sets 42
4.6 Summary 43

5 Conclusion 45
5.1 Collection selection algorithms 45
5.2 Incomplete resource descriptions 46
5.3 Measuring collection selection performance 46
5.4 Cluster methods 47
5.5 Using clustering to improve collection selection 47
5.6 Summary 48

6 Future work 49
6.1 Replacing CORI 49
6.2 Conducting experiments using different corpora 49
6.3 Improving performance 50

Bibliography 51

A Bisecting k-means recall and precision 57

B K-means recall and precision 65

C K-means and bisecting k-means 67

D Recall and precision values 69

Chapter 1

Introduction

Distributed information retrieval is a promising technique to improve the quality and scalability of web search. A major part of distributed information retrieval is collection selection. We propose to use collection clustering to improve the performance of existing collection selection algorithms.

1.1 Information retrieval

One of the web's most important applications is search. People who use web search engines have an information need, that is expressed by a query. The query is a short description of the information need, usually consisting of a single or a few words. The goal of the search engine is to return a list of web pages that best match the information need as described by the user's query, ranked by estimated relevance.

In order to do this, a search engine needs information about the documents that are available on the web. This information is gathered by a process called crawling, in which web pages are retrieved and stored. The most common way to make all the retrieved information easily searchable, is by creating an index which keeps track of the words that each document contains.

The matching of a user query against the documents that are present in an index, is much like looking up a word in the index in the back of a book. Retrieval methods have been developed that not only select the documents that contain the query words but also rank them by relevance. A very popular method is $tf \cdot idf$ which uses the number of occurrences of a word in a document, the total number of words in that document and the number of documents in the collection containing the word to calculate a ranking score. Other methods use the number of links from other web pages to a document as a measurement of relevance, or language models that calculate the probability that the query was generated by a given document.

1.2 Centralized search

Web search is currently dominated by centralized search engines. These search engines use a single large index for searching purposes. Even though the services may run on a large distributed cluster the control over the data is still centralized. Centralized control can be an advantage for large companies like Google or Yahoo, but it also has a number of disadvantages.

Crawling, indexing and searching a considerable part of the web requires a huge amount of resources. Even the biggest search engines index only a small part of the web. In 2005, the four biggest search engines together had indexed no more than 30% of the total visible web pages [16]. These numbers are only about the surface web. The deep or invisible web consists of pages that are hidden behind the query forms of searchable databases. For example: web pages that use AJAX[1] for displaying dynamic content to the user. Automatic web crawlers often can not access this data since they are not able to fill in and submit web forms. The deep web is estimated to be 500 times larger than the surface web [17].

Due to the size and dynamic nature of the web it is very hard, if not impossible, to keep a large index fresh. Changes in web pages are not noticed until the page is crawled again.

A non-technical issue is the monopoly of large search engines. A few companies control the way we can search information on the web and, as a result, can control which information becomes publicly available [22].

1.3 Distributed information retrieval

In distributed information retrieval -also called federated search or metasearch- a broker sends a query to multiple search engines at the same time. These search engines may use classic search indexes that contain the web pages, but may also have direct access to the data that is hidden behind web forms. When web pages are dynamically generated from data in a database, a search engine can search in this database instead of searching in an index based on previously generated web pages. Each collection evaluates the query and returns the results to the broker. The broker then merges the results and presents them to the user as a single result list. Distributed search consists of three steps: collection description, collection selection and results merging. An overview of this process is show in Figure 1.1.

Collection description Collection description is the task where the broker learns which collection are available and what information is contained by each collection. Collection descriptions are mostly built on statistics

[1]Asynchronous JavaScript and XML

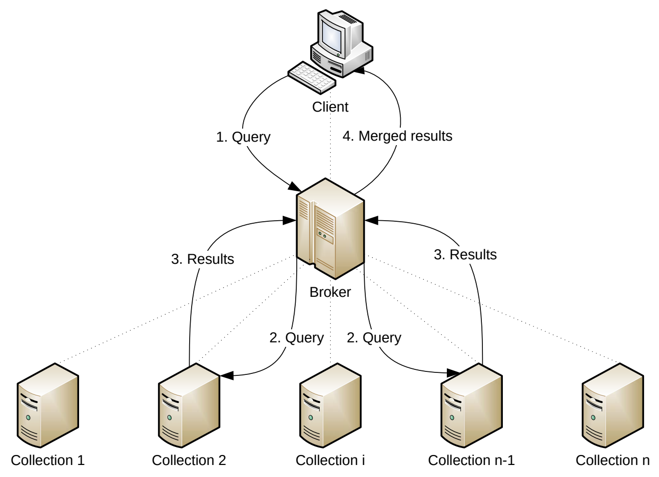

Figure 1.1: The broker forwards a query to a number of collections and returns the merged result list. Image taken from [6].

about word distributions in the collection, but can also contain information obtained from the search engine's interface or manually added metadata.

Collection selection A distributed search engine may be able to direct queries to thousands of search engines. Sending a user query to all of them will generate too much overhead. Therefore the broker must select a limited number of search engines to use. This task is called collection selection. The main goal is to select only those collections that will return the most relevant results in response to the user's query.

Results merging When the results from the search engines are returned to the broker the results must be merged. Duplicate results are removed and a single result list is generated, typically ordered by relevance. Merging algorithms can use a wide range of available information about the retrieved results, from their local ranks, their titles and snippets, to the full documents of these results [25].

1.4 Collection selection based on content similarity

Collection selection algorithms mostly depend on content summaries derived from the search engines they address. These content summaries can be retrieved in two possible ways.

First, the search engine can cooperate and generate the content summaries. These summaries are based on the full document collection that is present in the search engine's index. However, we can not always fully trust the search engines, because they might intentionally or unintentionally provide incorrect descriptions.

Second, resource descriptions can be obtained by random sampling techniques such as query-based sampling [8]. Queries are sent to the search interface of a search engine to retrieve a subset of the indexed documents. These queries can be generated from lexicons, previously crawled documents or taxonomies. Most sampling techniques are developed to retrieve a random, unbiased set of documents from a search engine, but focused probing techniques that retrieve documents about certain topics have also been proposed [3, 18].

All resource description summaries that are constructed by random sampling techniques suffer from the same problem. They are constructed from a small subset of documents from the collection they represent. Zipf's law [48, 29] states that given some corpus of natural language, the frequency of any word is inversely proportional to its rank in the frequency table. The most frequent word will occur approximately twice as often as the second most frequent word, which occurs twice as often as the fourth most frequent word, and so on. This means that most words in a collection occur only a few times and are thus not likely to be present in the small subset of retrieved sample documents. As a result, the content summary of a collection does not contain the majority of words that is present in the collection.

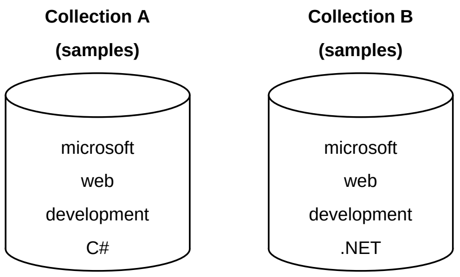

Figure 1.2: Two incomplete sets of samples.

An example of the problem is show in Figure 1.2. Two collections are indexed using query-based sampling. The image shows the words that are sampled for each of the collections. If a user now poses the query ".NET", it is clear that collection B is relevant and will get a high ranking. Collection A will get a low ranking because the word ".NET" was not sampled and indexed for this collection. Based on other sampled words we can assume that also collection A will contain documents that are relevant to the query.

A possible solution to this problem is to not only select the best search engine for a given query, but also selecting other search engines that have similar content in their indexes. According to Van Rijsbergen's cluster hypothesis, closely associated documents tend to be relevant to the same requests [43]. If we can assume that this hypothesis not only holds for documents but also for collections of documents, closely associated collections are also relevant to the same requests. Content summaries of topically similar collections can therefore complement each other.

1.5 Research questions

The focus of this research is the applicability of the cluster hypothesis to document collections instead of documents. We will research the possibility to improve collection selection methods using this hypothesis. The main research question is stated as follows:

> *Can the clustering of collections improve the performance of collection selection methods for distributed web search?*

What we want to prove is that the cluster hypothesis does not only hold for documents, but also for collections of documents. The cluster hypothesis is therefore rewritten as follows:

> *Closely associated collections tend to contain documents that are relevant to the same requests.*

We further refer to this hypothesis as the *collection cluster hypothesis.*

In this thesis a number of research questions are answered. First of all we take a look at previous work on collection selection algorithms and choose an existing collection selection algorithm and use it for conducting experiments using collection clustering.

1. *Which collection selection algorithm can be used in combination with collection clustering for this research?*

This part of the research is purely based on literature about the subject. As a result we give an overview of the current state of the research in the field, including a description of four well known collection selection algorithms.

In this research we use the random sampling to construct descriptions of the available collections. The number of sampled documents directly influences the amount of data transport and processing power that is needed and should therefore be kept as low as possible. As an effect the resource descriptions are always incomplete. Zipf's law can be applied here, which means that most keywords that are present in a collection will not be present in the collection description. Experiments are conducted to show the influence of this problem on collection selection.

2. *How big is the problem of incomplete resource descriptions?*
    (a) *What is the difference in performance of collection selection between scenarios where a full content summary is available and scenarios where content summaries are created using query based sampling techniques?*
    (b) *What is the effect of the number of sampled documents from which the content summaries are constructed?*

To answer these questions, we conduct collection selection experiments using the full collection data. We compare the results of these experiments to experiments conducted using query-based sampling with different numbers of samples. The results of these experiments show the relation between the number of samples and the performance of the collection selection algorithm.

In this research we setup a system for conducting clustering and collection selection experiments. We need to simulate a distributed information retrieval environment. As test data the WT10G corpus is used, containing real-world data and queries. This corpus is split into collections so every collection can simulate a search engine as part of a distributed system. The retrieval experiments deliver ranked lists of collections. We need a way to measure the quality of the rankings in order to compare the results of the experiments. In traditional information retrieval, the most common measurements are recall and precision, but these measurements can not be used directly for collection selection. This leads to the following research question:

3. *How can the performance of different collection selection algorithms be measured and compared?*
    (a) *How can the WT10G corpus be used for distributed information retrieval experiments?*
    (b) *What are the most suitable measurements that can be used to compare collection selection algorithms?*

If the test results of the modified algorithms show a significant improvement over the original algorithms, we will assume that this positive effect is caused by the application of the cluster hypothesis. This would show that the cluster hypothesis is valid for collection selection in distributed search engines.

In the rewritten cluster hypothesis we mention "closely related" collections. Clusters of collections will be created based on the relations between these collections. We need a technique to perform clustering of the collections automatically.

4. *What is the best technique for collection clustering?*

Two widely used cluster algorithms are the k-means algorithm and the bisecting k-means algorithm. We will conduct experiments using both algorithms and different parameters to determine which algorithm performs best.

The focus of this research is the use of collection clustering for collection selection. We will conduct experiments to evaluate the effects of clustering on collection selection. From the results we will be able to answer the following question:

5. *What are the effects of clustering on the collection selection performance?*

1.6 Thesis outline

The next chapter describes previous work that is relevant for this research and describes the collection selection and cluster algorithms that are used. Chapter 3 describes the research method. It describes the setup of the experiments, the data that is used and the evaluation procedure. Chapter 4 discusses the results of the experiments. The conclusions of our research are given in Chapter 5 and Chapter 6 gives some suggestions for future work.

Chapter 2

Literature

This chapter gives an overview of the relevant literature on the topics related to the research described in this thesis. We start by explaining Zipf's law and describing query-based sampling. This gives more insight in the cause of the problem of incomplete collection descriptions. Next, we describe the cluster hypothesis which is the theory on which our solution is based. In section 2.4 we discuss a number of collection selection algorithms. None of these algorithms use clustering to improve their performance. Section 2.5 discusses the k-means and bisecting k-means clustering algorithms. Section 2.6 describes some work that uses clustering to improve the quality of retrieval systems.

2.1 Zipf's law

Zipf's law [48, 29] states that given some natural language corpus, the frequency of any word is inversely proportional to its rank in the frequency table. Simply said, many words occur very few times and a few words occur very often.

The most frequent word will occur approximately twice as often as the second most frequent word, which will occur approximately twice as much as the fourth word, and so on. For example, in the British National Corpus[1], the most frequent word is 'the' which accounts for slightly over 6% of all word occurrences, the word 'of' accounts for almost 3% and the third most occurring word 'and' accounts for 2.7% of all words. Only 157 different words are needed to account for half the corpus. A graph that shows the word frequencies of the corpus is shown in Figure 2.1. Figure 2.2 shows the same data, but plotted on logarithmic axes. This graph shows an almost straight line which indicates a power function.

Zipf found that this distribution can be described by the function $f(r) = \frac{C}{r^{\alpha}}$, where C is the coefficient of proportionality, r is the word rank and α is the exponent of the power law which typically has a value close to 1.

[1]http://ucrel.lancs.ac.uk/bncfreq/

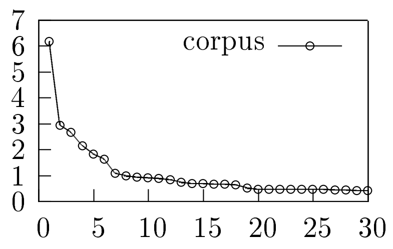

Figure 2.1: Word frequencies of the top 30 words from the British National Corpus.

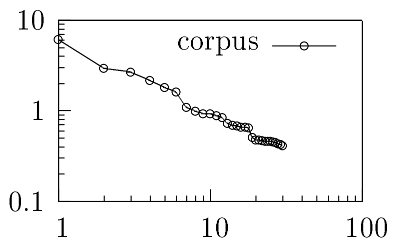

Figure 2.2: The same graph but plotted on logarithmic axes.

This function is found to apply not only to English texts but also to spoken language and non-English and non-Latin languages [32].

2.2 Query-based sampling

Query-based sampling [5, 9, 8] can be used to construct collection descriptions for collections that can not or will not cooperate, or can not be trusted. Query-based sampling uses only the most basic functions of a collection: the possibility to submit queries to a search interface and retrieve a set of documents from the result set. Most query-based sampling methods are designed to give a uniform and unbiased sample of the documents in a collection. If the sample is uniform and unbiased, the resource descriptions resemble the resource descriptions that would have been constructed if they were created from the full data collection. At the same time we want to minimize the costs of constructing these collection descriptions by keeping the number of interactions with the search interface and the number of retrieved documents as low as possible. The most straightforward query-based sampling algorithm is outlined below.

1. Select a one-term query.
2. Submit the selected one-term query to the search interface.
3. Retrieve the top n documents from the result set.
4. Update the resource description based on the content of the retrieved documents.
5. If the stopping criterion has not been reached, go to step 1.

Most implementations of the algorithm vary on the choice of the query terms. During the first iteration the learned language model is empty so the term is chosen from an external resource like a dictionary or a previously created language model. In subsequent iterations the query terms can be chosen from the language model that is learned from the retrieved documents. A random term can be selected, but statistics about the number of occurrences of the words may also be used. Prior research [41] shows that using the least frequent terms in a sample yields a better resource description than randomly chosen terms for large collections.

Query-based sampling suffers from two types of biases. Query bias is a bias towards longer documents that are more likely to be retrieved for a given query. Ranking bias is caused by the fact that search engines give certain documents higher ranks and query-based sampling only retrieves documents up to rank n [40].

Bar-Yossef and Gurevich [3] describe two methods that are not affected by these biases and guarantee to produce near-uniform samples from a collection. The samples that are taken first are biased, but receive a weight which represents the probability of the document being sampled. These weights are used to apply stochastic simulation methods on the samples and obtain uniform unbiased samples from the collection.

The work described above focuses on retrieving uniform and unbiased samples. This is necessary for making size and overlap estimations of search engines. The question is whether unbiased samples are needed for creating useful resource descriptions. For describing resources, biased samples may be more representative. Gravano et al. [15] describe a technique called focused query probing which creates a topic specific description. This approach is effective in scenarios in which resources contain topic specific and homogeneous content.

2.3 Cluster hypothesis

The cluster hypothesis is based on the idea that if a document is relevant to a given query, then similar documents will also be relevant to this query. This was formulated by Van Rijsbergen [43] as:

> *Closely associated documents tend to be relevant to the same requests.*

If similar documents are grouped into clusters, then one of these clusters will contain the documents that are relevant to a query and the retrieval of the relevant documents is reduced to the identification of this cluster. This type of information retrieval is called cluster-based retrieval.

Cluster-based retrieval was at first seen as a method of improving the efficiency of information retrieval systems. The amount of data that needs to be compared to the query is reduced by first selecting the clusters that are searched. Jardine and Van Rijsbergen [21] found that not only the efficiency could be improved, but also the effectiveness. The reason for this is that cluster-based search takes into account the relationship between documents.

2.4 Collection selection algorithms

The purpose of collection selection is to select those collections that contain documents that are relevant to a user's query. Many collection selection algorithms have been proposed in literature. This section describes four well known collection selection algorithms that are based on different methods. From these algorithms we choose to use CORI for the experiments in this research.

2.4.1 GlOSS

GlOSS (Glossary-of-Servers Server) [13, 14] is one of the first and well studied database selection algorithms. The original version of GlOSS was based on the rather primitive Boolean model for document retrieval. A generalized and more powerful version named gGlOSS was presented which is based on the vector-space retrieval model.

gGlOSS represents each collection c_i by 2 vectors that contain the following values:

1. Document frequency f_{ij}: the number of documents in collection c_i that contain term t_j.
2. The sum of the weights w_{ij} of term t_j over all documents in c_i. The weight of a term t_j in a document d is typically a function of the number of times that t_j appears in d and the number of documents in the collection that contain t_j.

gGlOSS defines the ideal ranking $Ideal(l)$ as the ranking of the collections according to their goodness. The goodness of a collection c with respect to query q at threshold l is defined as

$$Goodness(l, q, c) = \sum_{d \in \{c | sim(q,d) > l\}} sim(q, d) \tag{2.1}$$

where $sim(q, d)$ is a similarity function which calculates the similarity between a query q and a document d.

Because the information that gGlOSS keeps about each collection is incomplete, assumptions are made about the distribution of the terms and their weights across the documents in the collection. An estimation of the $Ideal(l)$ rank is made using these assumptions. Two functions $Max(l)$ and $Sum(l)$ can be used as estimators.

To derive $Max(l)$, gGlOSS assumes that if two words occur in a user query, then these words will appear in the collection document with the highest possible correlation. This means that if a query contains two terms t_1 and t_2 that occur in respectively f_{i1} and f_{i2} documents and $f_{i1} \leq f_{i2}$, it is assumed that every document in collection c_i that contains t_1 also contains t_2.

A disjoint scenario is estimated by $Sum(l)$, where it is assumed that two terms that appear in a user query do not both appear in the same document. This means that the set of documents in c_i that contains t_1 is disjoint with the set of documents in c_i that contains t_2, if $t_1 \neq t_2$.

2.4.2 Cue Validity Variance

The Cue Validity Variance method (CVV) [47] compares the variance of the cue validity of the query terms across all collections. The cue validity of term t for collection c_i measures the degree to which term t distinguishes documents in c_i from documents in other collections. CVV uses only document frequency data to produce the rankings. The cue validity can be calculated using the function

$$CV_{ij} = \frac{\frac{DF_{ij}}{N_i}}{\frac{DF_{ij}}{N_i} + \frac{\sum_{k \neq i}^{|C|} DF_{kj}}{\sum_{k \neq i}^{|C|} N_k}} \tag{2.2}$$

where N_i is the number of documents in c_i and $|C|$ is the number of collections in the system.

The cue validity variance CVV_j is the variance of the cue validities for all collections with respect to t_j. It can be used to measure the usefulness of a query term for distinguishing one collection from another. The larger the variance is the more useful the term is to differentiate collections. The collections are ranked based on a goodness score. Given a set of collections C, the goodness of a collection $c_i \in C$ with respect to query q with M terms is defined as

$$Goodness(c, q) = \sum_{j=1}^{M} CVV_j \cdot DF_{ij} \tag{2.3}$$

where DF_{ij} is the document frequency of term j in collection c_i. CVV_j is the variance of CV_j which is the cue validity of term j across all collections.

CVV is found to be very accurate when complete resource descriptions are used, but can not effectively be used in combination with query-based sampling [30]. Further, it has a small bias towards collections with long documents and collections with many documents [11].

2.4.3 CORI

CORI [10] is an algorithm that takes a probabilistic approach to collection selection by using Bayesian inference networks [42]. These networks are directed acyclic graphs (DAGs). Figure 2.3 shows an example of an inference network that contains of four types of nodes. The nodes are connected by edges, where an edge pointing from node p to node q is weighted with the probability $P(q|p)$ that q implies p.

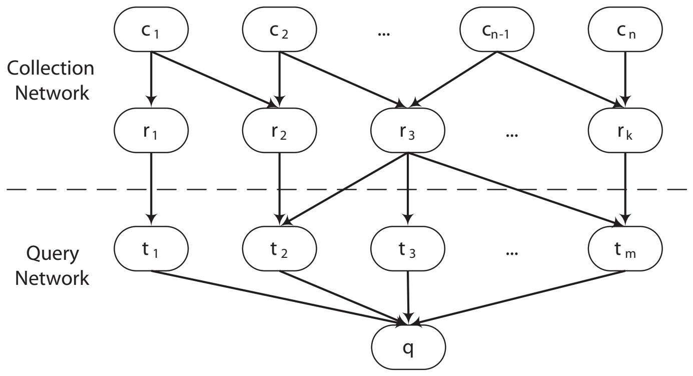

Figure 2.3: Example of an inference network.

The leaf nodes c_n are the collection nodes and correspond to the event that a collection is observed. There is a collection node present for every collection in the corpus. The representation nodes r_k correspond to the terms in the corpus. The collection nodes and representation nodes together form the collection network, which is built once for a corpus and does not change during query processing. The probabilities in this network are based on collection statistics. CORI uses document frequencies (*df*) and inverse collection frequencies (*icf*), that are calculated analogously to the common *tf* and *idf* values. This is possible because a collection is treated as a bag of words, just as a document is treated as a bag of words for calculating *tf* and *idf*. *df* is the number of documents containing a given term, *icf* is the number of collections containing the term.

The query network contains a single query node q which represents the user's query. The query nodes t_m correspond to the terms in the query. The query network is built for each query and is modified during query processing.

The collection ranking score for query q is the sum of the beliefs $p(t_m|c_i)$ in collection c_i due to observing term $t_m \in q$. This belief can be calculated using the following equations:

$$p(t_m|c_i) = d_b + (1 - d_b) \cdot T \cdot I \tag{2.4}$$

$$T = d_r + (1 - d_r) \cdot \frac{log(df + 0.5)}{log(max\_df + 1.0)} \tag{2.5}$$

$$I = \frac{log(\frac{|C|+0.5}{cf})}{log(|C| + 1.0)} \tag{2.6}$$

where

df	is the number of documents in c_i containing t_m.
max_df	is the number of documents containing the most frequent term inc c_i.
$\lvert C\rvert$	is the number of collections.
cf	is the number of collections containing term t_m.
d_r	is the minimum term frequency component when term t_m occurs in collection c_i. The default value is 0.4.
d_b	is the minimum belief component when term t_m occurs in collection c_i. The default value is 0.4.

The belief values are normalized to be between 0 and 1.

2.4.4 Indri

The retrieval model implemented in Indri [28, 27] uses a combination of the language model [31] and inference network approaches. Although it was not developed for collection selection it is possible to use it for that purpose by combining all documents of a collection into one single document [33]. A language model is constructed from every document in a collection. Given a query q, for every document the likeliness is estimated that the document's language model would generate the query q. Indri uses language modeling estimates rather than $df \cdot icf$ estimates for calculating the beliefs of the nodes in the inference network.

Instead of using Equation 2.4 for estimating the beliefs, Indri uses a probability based on the language model. This is probability is calculated by the equation

$$P(r|D) = \frac{tf_{r,D} + \alpha_r}{|D| + \alpha_r + \beta_r} \tag{2.7}$$

This is the belief at representation node r given a document D (in collection C). In this equation, α_r and β_r are smoothing parameters. Smoothing is a method used to overcome both the zero probability and data sparseness problem. The values for the smoothing parameters can be set in many ways. Dirichlet smoothing is widely used, which assumes that the likeliness of observing a representation concept is the same as the probability of observing it in collection C. The following values for the smoothing parameters are used by default [27]:

$$\alpha_r = \mu \cdot P(r|C) \tag{2.8}$$
$$\beta_r = \mu \cdot (1 - P(r|C)) \tag{2.9}$$

where μ is a tunable smoothing parameter which has a default value of 2500. Equation 2.7 can now be rewritten as

$$P(r|D) = \frac{tf_{r,D} + \mu \cdot P(r|C)}{|D| + \mu} \tag{2.10}$$

Indri provides a query language that can express complex concepts. Details on this query language can be found in [39].

2.4.5 Discussion

Prior research gives an overview of the performance of the collection selection algorithm described above. CORI proves to be one of the most consistently effective algorithms in various situations [30, 12]. One known weakness in this algorithm is that the results are worse in environments that contain both very small and very large databases [35]. CVV can be very accurate when used with complete resource descriptions, but the performance drops when query-based sampling is used [11].

Indri is a fairly new algorithm compared to the other mentioned algorithms. Not much research on comparing the performance to other algorithms has been conducted yet. There is research that shows promising results for specific retrieval tasks [46, 2], but the results are not consistently good.

We choose to use CORI as the collection selection in this research because of its performance, but also because it is implemented in the freely available Lemur toolkit for language modeling and information retrieval. It can be used out-of-the box which means we do not need to implement the algorithm ourselves.

2.5 Clustering

The objective of document clustering is to group documents together that share the same implicit topic. At the same time, the different clusters should have different topics. There are various motivations within the field of information retrieval to perform clustering. Using document clustering it is possible to automatically create browsable taxonomies like the Yahoo Directory[2]. Taxonomies are usually manually created, which takes a lot of

[2]http://dir.yahoo.com

effort to keep them up-to-date. A different goal of document clustering is to improve the efficiency of information retrieval systems. Searching clusters of documents together instead of all documents separately can decrease the number of calculations and therefore increase the speed of the search operation. Clustering can also be used to show the query results grouped by topic. This can give users a better overview of the different documents that were found and the relationships among them.

A good document clustering algorithm produces clusters that meet the following criteria:

- The intra-cluster similarity is high, which means documents in the same cluster are similar.
- The inter-cluster similarity is low, which means documents in different clusters are dissimilar.

2.5.1 Clustering types

There are two types of clustering methods: hierarchical and partitional methods. Hierarchical clustering methods generate trees of clusters, so called dendograms. The root of such a tree is a cluster that contains all documents and the leaves are individual documents. Hierarchical clustering algorithms can be either divisive where the dendogram is created by starting with one cluster containing all documents and recursively splitting the clusters into smaller ones, or agglomerative where at the start every document is considered a cluster which is recursively merged into a bigger cluster. Partitional clustering methods on the other hand create a one-level partitioning of the documents. This is typically done by selecting a number of initial clusters and assigning all documents to the closest cluster based on some measure of similarity.

2.5.2 K-means algorithm

The k-means algorithm [24, 26] is one of the most widely used clustering algorithms. It is a partitional algorithm that is based on the idea that a centroid can represent a cluster. Clustering is seen as a optimization problem in which an assignment of data vectors to clusters is desired, such that the sum of the similarities between the vectors and their cluster centroids is optimized. The document set containing n documents is denoted by $d_1, d_2, ..., d_n$. The objective is choose the number of clusters k and assign the documents to these clusters C_j in such a way that the function

$$\sum_{j=1}^{k} \sum_{d_i \in C_j} sim(d_i, \widehat{c_j}) \tag{2.11}$$

is either minimized or maximized, depending on the choice of the similarity function $sim(d_i, \widehat{c_j})$. This similarity function gives a value for the similarity between a document vector d_i and a cluster centroid $\widehat{c_j}$ which is a measure for the intra-cluster similarity. K-means does not take the inter-cluster similarity into account. Different similarity functions can be chosen, but most common is to use the Euclidean distance or cosine distance. The cluster centroid can be defined in different ways, but is often the median or the mean point of the cluster. The mean point of a cluster C_j that consist of the document set $d_1, d_2, ..., d_n$ is given by

$$\widehat{c_j} = \frac{1}{|C_j|} \sum_{d_i \in C_j} d_i \tag{2.12}$$

which is the vector obtained by averaging the weights of the various terms present in the cluster's documents.

Finding the best clustering, thus maximizing function 2.11, is know to be an NP-hard problem [26] and therefore a heuristic algorithm is generally used which gives an approximate solution. It maximizes the sum of the intra-cluster similarity values when an initial assignments of centroid is provided. The number of clusters k is fixed during the run of the algorithm and is chosen based on the problem and domain.

1. Select k points as the initial centroids. These points are selected randomly. A point is a vector representing a collection of documents or a single document.

2. Assign all points to the cluster with the closest centroid. Which is the closest centroid is determined by the similarity function.

3. Recompute the centroid of each cluster.

4. Repeat steps 2 and 3 until the centroids don't change.

The resulting clustering depends on the choice of the initial centroids. There is no guarantee that it will converge to a global optimum. It is common to run the algorithm multiple times with different initial centroids and select from the results the best clustering according to function 2.11.

Because the initial centroids are chosen in the first step, the number of collections, represented by k, must be specified prior to application. The choice of k therefore highly influences the results and must be carefully chosen. Another effect of choosing the centroids in the first step is that the variation in size of the resulting clusters may be large. An initial centroid which is central in the vector space may grow large, while an outlier may become a very small cluster.

The k-means clustering algorithm is very fast when the number of clusters is small. However when the number of clusters grows large, for example to a thousand of clusters, the efficiency decreases and the complexity approaches $O(n^2)$ where n is the number of documents [44].

2.5.3 Bisecting k-means algorithm

Bisecting k-means [38, 23] creates a hierarchical cluster tree using the k-means algorithm. It has two major advantages over the k-means algorithm. First, the complexity of the algorithm is linear to the number of documents, thus $O(n)$. Second, the number of clusters that is produced does not necessarily need to be known beforehand, as we will describe later in this section.

It is an agglomerative method so initially the whole collection set is considered one cluster. Recursively, a cluster is selected and split into two clusters using the k-means algorithm until a stopping criterion has been reached. The algorithm typically stops when the desired number of clusters is reached.

1. Select a cluster to split. There are several ways to do this. Most common is to select the largest cluster, the cluster with the least overall similarity or a combination of cluster size and similarity.
2. Divide the cluster into two clusters using the k-means algorithm. This means executing the algorithm using $k = 2$.
3. Repeat step 2 for a fixed number of times. Select the split with the highest overall similarity. The results of the k-means algorithm are dependent on the randomly selected initial clusters. By repeating this a number of times the quality of the resulting clusters can be improved.
4. Repeat steps 1-3 until the stopping criterion is reached, typically when a maximum number of clusters is created.

The complexity of the bisecting k-means clustering is linear to the number of documents [38]. This makes it more efficient than the k-means algorithm when the number of clusters is large. This is caused by the fact that there is no need to calculate the distance of every document to the centroid of each cluster since we consider only two centroids in the bisecting step.

Specifying cluster sizes The number of clusters and the sizes of the clusters are dependent on the stopping criterion in step 4. We can control the number of clusters that should be produced by stopping the algorithm when a certain number of clusters has been reached. We also have more control of the number of documents contained in each cluster. When a cluster has reached a size smaller than a given maximum size s_{max} we can decide not to split it any further and choose another cluster to split or stop the algorithm. We also want to be able to specify a minimum cluster size s_{min}. If a cluster c is split into clusters c_a and c_b, and the size of c_a or c_b is smaller than s_{min}, we discard the split and create another split for cluster c. This requires that $s_{max} \geq s_{min} \cdot 2 - 1$.

New collections can be added to an existing clustering by assigning them to the cluster with the closest centroid. We must keep in mind that the clusters do not grow bigger than s_{max}. When this happens, we split this cluster into two clusters.

2.6 Cluster-based retrieval

Some research on cluster-based retrieval has already been performed. Xu and Croft [45] describe three methods for optimization of distributed information retrieval by clustering the resources. Using the first two methods called global clustering and local clustering, the documents are physically partitioned into collections. This requires besides a global coordinating system the cooperation of the collections. The third method, multiple-topic representation, does not require physical partitioning and avoids the cost of re-indexing. Each subsystem summarizes its collection as a number of topic models. With this method, a collection corresponds to several topics. The INQUERY retrieval system [7] is used for indexing and searching collections. The steps to search a set of a distributed collections for a query are (1) rank the collections against the query, (2) retrieve 30 documents from each of the best n collections, (3) merge the retrieval results based on the document scores.

SETS [4] is an algorithm for efficient distributed search in P2P networks. Participating sites are categorized by topic. Topically focused sites are connected by short-distance links and clustered into segments. These segments are connected by long-distance links. Queries are sent only to the topically closest segments.

The algorithm described in [34] automatically categorizes specialty search engines into a hierarchical structure based on the textual content of the documents. The taxonomy from the DMOZ Open Directory Project[3] is used during the research. Collections are classified into taxonomy categories by using probe queries. This is very similar to the work described by Ipeirotis at al. [20, 18]. First a hierarchical classification scheme or taxonomy is defined. For each category in this taxonomy a number of probe queries are generated. These are queries that return a set of results that is relevant to the category. The query *Jordan AND Bulls* for example will retrieve mostly documents in the sports category. Instead of retrieving the actual documents, only the number of results is counted. From the number of matches for each probe query it is possible to make an estimation about the topics covered by a collection and categorize the collection in the taxonomy. In more recent work Ipeirotis at al. [19] describe an algorithm for collection selection for a

[3]http://dmoz.org

given query. From the top of the hierarchy at each level the best category is selected using existing algorithms as GlOSS or CORI. This process proceeds recursively until the number of collections under the selected category drops below a certain value.

2.7 Summary

This chapter gave an overview of the relevant literature about the topics related to the research described in this thesis. A explanation of Zipf's law is given, as well as a description of query-based sampling to give some background information on the cause of the problem of the incompleteness of the collection descriptions.

Four collection selection algorithms are discussed. We chose to use CORI for the experiments in this research because of its performance and because it is included in the Lemur toolkit for language modelling and information retrieval.

We propose to use clustering to improve the collection selection performance. The k-means algorithm and the bisecting k-means algorithm are described in Section 2.5. We will run experiments using both algorithms to find out if there is difference in performance when used in combination with collection selection.

Chapter 3

Research

In this research, we use a prototype of a distributed information retrieval system. An overview of the system is shown in Figure 3.1. The system is divided into two parts. The indexing on the left side is initiated by the server and produces two indexes that are needed to perform the querying. The querying, shown on the right, is initiated by the users. It involves selecting relevant collections from the index and ranking them according to their supposed relevance to the user's query. This chapter will describe in detail how the different parts of the system are implemented. Further, it will describe how the evaluation of the system is performed.

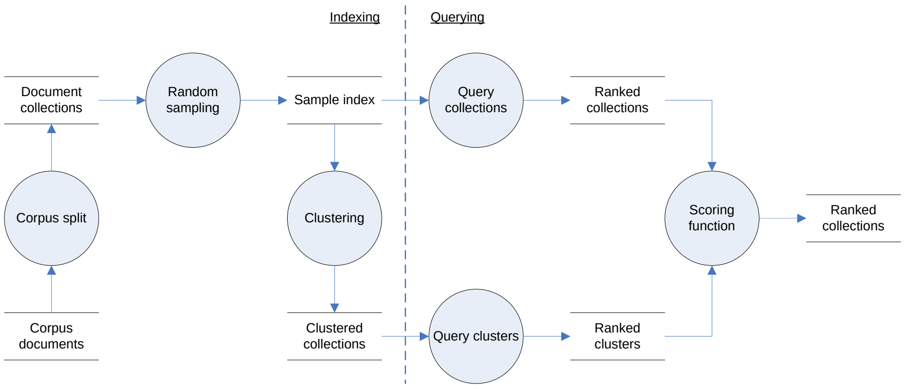

Figure 3.1: Experimental setup.

3.1 The WT10g corpus

The dataset used in our experiments is the WT10g corpus, used in the TREC-9 and TREC 2001 Web Tracks. It is constructed from the 100GB Very Large Corpus 2 (VLC2) and contains about 1.69 million documents with a total size of about 10GB. From this corpus, binary and non-English documents were removed, as well as duplicate and redundant data. Details about the construction of the WT10g corpus can be found in [1]. Some statistics on the WT10g corpus as taken from the TREC website[1] are:

- 1,692,096 documents
- 11,680 servers
- an average of 144 documents per server
- a minimum of 5 documents per server

A number of measurements on the WT10g corpus is performed by Soboroff [37]. From this measurements the conclusion is drawn that the corpus retains the properties of the web, and is therefore a good representation of the web for research purposes.

There are two sets of topics with relevance judgements that can be used with the WT10g corpus. Topics 451-500 include a number of misspelled words, topics 501-550 do not. The relevance judgements tell whether a document is considered to be relevant to a topic or not. These relevance judgements are made by humans and classify documents as irrelevant, relevant or highly relevant.

Splitting the corpus The WT10g corpus was not created for distributed information retrieval. Compared to previous TREC corpora, WT10g has better support for distributed information retrieval. However, it is not possible to use it for distributed information retrieval experiments without any preprocessing. In order to represent a distributed environment consisting of a lot of different search engines, the corpus is split based on the server IP address. By doing so, we create 11,512 separate data collections. This is a little less than the 11,680 servers that are present in the WT10g corpus, which means that a few servers share the same IP address.

We counted the number of documents in each collection. The smallest collection contains 5 documents and the largest collection contains 26,505 documents. The average collection size is 147. This is a little more than the 144 documents per server in the corpus, which is again caused by the fact that we have a little less collections than there are servers in the corpus. Table 3.1 shows the number of collections with a given maximum size. The collections are fairly small, only 11.37% of the collections contains more than 200 documents.

A split created this way is representative for a scenario where small search engines index single websites. In this scenario, there is no overlap between the indexes of the search engines. Documents are present in just one index. We

[1]http://ir.dcs.gla.ac.uk/test_collections/wt10g.html

Table 3.1: Collection sizes

max size	count	percentage
200	10203	88.63%
100	9228	80.16%
50	7725	67.10%
20	4977	43.23%
10	2547	22.12%
5	443	3.85%
1	0	0.00%

expect that there will be coherence in the topics of the documents contained by these collections. This means we expect that two documents on a single server are more likely to share the same topic than two documents that reside on different servers.

Documents and collections A collection contains all documents that originate from a server with the same IP address. The data from these documents is merged into a single file and assigned a unique identifier for the collection. This is possible because CORI treats collections as bags of words, as if they were large single documents. No statistics are used about the individual documents in the collections.

3.2 Random sampling

In practice all random sampling techniques are biased. In this research we do not want this to affect our results. It is also not necessary to use a technique like query based sampling, because we have the full corpus at our disposal. By randomly selecting documents from a known collection, we simulate the perfectly random sampling method. The selected samples are a subset of the full collection data and are indexed as the sample index.

Creating indexes In order to evaluate the effect of the number of samples used in the random sampling procedure, we create different indexes for different amounts of samples. From every collection in the corpus, we take n randomly selected documents and store these as a partial collection. This way a new index can be constructed where every collection contains at most n documents. If a collection contains less than or exactly n documents, this means the full collection will be retrieved. We create indexes for $n \in \{1, 5, 10, 20, 50, 100, 200\}$. For comparison, an index is created that contains all documents.

3.3 Clustering

After the random sampling indexes have been created we cluster the sampled collections. We use both the k-means and the bisecting k-means algorithm. When the k-means algorithm is used we have to give the number of clusters that is to be produced as a parameter. Using bisecting k-means we specify the upper and lower bounds of the cluster sizes.

3.3.1 Reducing the calculations complexity

We make some compromises in the way we execute the clustering algorithm in order to reduce the number of calculations needed during the clustering process. These compromises are similar to those described in [44].

First of all, the feature vectors that represent the collections are reduced to the 25 most frequently occurring terms in the collection. Stop words are removed from the feature vectors first. This highly reduces the number of calculations needed to calculate the distance between collections and cluster centroids.

Second, we reduce the number of iterations in the splitting step. In the original k-means algorithm all collections are reassigned to the closest centroid until the centroids do not change. It may take a long time before the clusters converge to a situation like that while the clusters do not change much. For the k-means algorithm we conduct experiments with different numbers of iterations to test the influence of this on the collection selection results. For the bisecting k-means algorithm we limit the number of iterations to a maximum of three.

In the bisecting k-means algorithm, the splitting of the clusters is performed multiple times after which the best split is selected. This is done to reduce the effect of the random selection of the initial centroids during the execution of the k-means algorithm. We execute this step only two times to reduce the number of calculations.

3.3.2 K-means

Number of clusters The k-means algorithm requires a value for the parameter k, the number of clusters that is to be produced. We run collection selection experiments using clusterings of 384, 767, 1535 and 3837 clusters. These clusters have an average size of respectively 30, 15, 7.5 and 3 collections. From the results we evaluate the effects of the number of clusters on the collection selection results. The sizes of the clusters also depend on the number of clusters. For this reason we also analyze the sizes of the produced clusters.

Table 3.2: Cluster sizes for bisecting k-means

cluster size		# clusters	
min	max	min	max
5	20	576	2303
2	20	576	5756
1	20	576	11512
2	10	1152	5756
1	10	1152	11512
1	1	11512	11512

Cluster quality As explained in the previous section, one of the means to reduce the complexity of the calculations is to limit the number of iterations for reassigning the collections to the clusters. This affects the quality of the clusterings. The intra-cluster similarities of the clusters will be lower. We run experiments where the number of iterations is limited to 1, 5 and 25. We evaluate the effects of the number of clusters on the collection selection performance.

3.3.3 Bisecting k-means

We adjusted the bisecting k-means algorithm in such a way that it allows us to specify the sizes of the clusters that are produced by the algorithm. The size of the clusters may affect the performance of the final collection selection step. In order to tell what the effects of the cluster sizes are, we produce a number of clusterings with different cluster sizes. Our implementation of the bisecting k-means algorithm does not allow us to define the exact size of the clusters but we can set a minimum and maximum size. Clusters that have a size larger the the maximum size are split. If one of the clusters that is produced by splitting a larger cluster is smaller than the minimum size, the split is considered invalid and splitting is performed again until the sizes of both clusters are in the desired range. Setting a minimum size may favor clusters with a certain size over clusters with high intra-cluster similarity, which leads to clusterings with lower quality. For that reason we keep the minimum cluster size low.

The minimum and maximum sizes of the different clusterings that are produced are given in table 3.2. It also shows the minimum and maximum number of clusters that will result from running the algorithm. When no clustering is performed, the result is equal to a clustering with exactly one collection per cluster. This situation is added for comparison.

3.3.4 Indexing clusters

All collections that are in a single cluster are merged together as a single collection. Since a collection is represented by CORI the same way as a document, a cluster is also represented as a document. This way it can be indexed by a general information retrieval system without any adjustments.

3.4 Ranking collections

Upon query time, the system generates a ranked list of collections that are relevant to the query. This list is created in three steps, that are visualized in the right part of Figure 3.1.

1. A ranked list is created by ranking the collections from the index where no clustering was applied. This assigns a ranking score to each individual collection.
2. The clusters are ranked using the index that contains the clustered data. Each cluster is assigned a ranking score.
3. A scoring function combines the scores from step 1 and step 2 to a combined score for each collection. The final ranking is created by rearranging the collections based on the combined scores.

We use the CORI algorithm as it is implemented in the Lemur toolkit to perform the actual collection selection steps (1 and 2). Collections and clusters are indexed and queried as if they were documents.

For this research the scoring function is kept simple. It calculates the combined scores by adding the cluster scores and collection scores. Because both scores are calculated using CORI, no normalization is needed. Other scoring functions are possible, for example to let the cluster score account more to the combined score.

Not all collections are ranked, because not all collections are found relevant by the collection selection algorithm. In this case the recall will not add up to one when all results are retrieved. This is corrected by adding the collections that were not ranked to the result list in a random order.

3.5 Evaluation

This section describes the metrics that are used for the evaluation of the test results. For the evaluation of the query results we use two metrics: recall and precision.

Collection merits The calculation of the recall and precision scores are based on relevance judgements. Relevance judgements are binary and tell whether a document is relevant or irrelevant to a query. The relevancy of a collection is given by the number of relevant documents contained by this collection. This number of relevant documents is called the merit of a collection.

Recall and precision In traditional information retrieval, precision and recall are the most widely used metrics for results evaluation. Similar metrics were defined for distributed information retrieval by Gravano et al. [13].

We know which documents are relevant to each query. From this information we calculate a merit score for each collection by simply counting the number of relevant documents in that collection. The merit of collection c_i for query q can be expressed as $merit(q, c_i)$.

In the ideal ranking, the collections with the highest merit score are ranked first. We use this ranking as our baseline ranking B. The merit of the i^{th} collection in this ranking is given by $B_i = merit(q, c_{b_i})$. For the ranking under evaluation, the merit is given by $E_i = merit(q, c_{e_i})$

The recall is defined by Gravano et al. as

$$R_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i} \tag{3.1}$$

This recall is a measure of how much of the available merit in the top n ranked collections has been accumulated by the ranking under evaluation.

A slightly different recall measure was defined by French et al. [12] as

$$\widehat{R_n} = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i} \quad \text{where} \quad n^* = max(k) \text{ such that } B_k \neq 0. \tag{3.2}$$

n^* is the breakpoint between the relevant and non-relevant collections. The denominator is the total merit of all collections that are relevant to the query. The recall $\widehat{R_n}$ gives the percentage of the total merit that has been accumulated in the top n collections in the ranking under evaluation.

R_n can be 1 from the beginning, when the ranking under evaluation is equal to the ideal ranking, while $\widehat{R_n}$ will gradually increase to 1. This also means that R_n can decrease as n increases. From the point where $n = n^*$, the values for R_n are equal to $\widehat{R_n}$. Because $\widehat{R_n}$ is always increasing, it is the most similar to the tradition recall metric.

Gravano et al. [13] also describe a precision metric that can be used in collection selection scenarios. It gives the fraction of the collections rc with a non-zero merit that occur in the Top_n retrieved results. A higher precision P_n is better because it reduces the number of database searches that will not give any relevant results.

$$P_n = \frac{|\{rc \in Top_n(S)|merit(q, rc) > 0\}|}{n} \tag{3.3}$$

Significance tests When two collection selection algorithms are compared and a difference is found, it is still a question whether this difference is significant or not. Some algorithms may do better for certain queries but worse for others. Significance test can be used to statistically prove that one algorithm performs truly better than another.

In information retrieval research, commonly used tests are the Student's paired t-test, the Wilcoxon signed rank test and the sign test. Smucker et al. [36] showed that the Wilcoxon and sign test have the potential to lead to false detections of significance. We will use the Student's paired t-test.

We run the significance test on two paired sets of results. These are the recall or precision values per query for both algorithms after n collections are retrieved. We assume that these sets of results are samples taken from normally distributed population. The null hypothesis is that there is no difference between the two algorithms.

A significance level is calculated from the paired result sets. This significance level gives the probability that the results could have occurred under the null hypothesis. A high value means that there is a high probability that there is no difference between the two algorithms. This significance level is also known as the *p-value*. We define that a *p-value* under the threshold of 0.05 disproves the null hypothesis. In this case we can say with a certainty of 95% that there is a difference between the two algorithms and therefore we consider the difference significant for n.

3.6 Summary

A setup was described for conducting collection selection experiments using clustering of collections. The experiments consist of the following steps:

1. Create a split of the WT10g corpus based on the IP address of the web server from where the documents were retrieved.
2. Perform random sampling of the corpus split. We do not use query-based sampling but take random samples from the full collections that are accessible.
3. Cluster the collections. We conduct experiments using the k-means and the bisecting k-means algorithm.
4. Run the queries and create ranked lists of collections.
5. Evaluate the ranked lists. We use recall and precision measures that are similar to the measures that are widely used in document retrieval.

The next chapter shows the results of the experiments.

Chapter 4

Results

This section gives an overview of the results of the conducted experiments. There are a few points of attention one must keep in mind before looking at the graphs in the following sections:

a) The recall and precision graphs show only the results for the first 1,000 retrieved collections. It is highly unlikely that in practice more collections will be addressed for a query. The collections that are ranked at the top are considered to be the most important, which is why the horizontal axes are on logarithmic scale. The recall is calculated using Equation 3.1, for calculating the precision Equation 3.3 is used.

b) It is also important to notice that the vertical axis of the precision graphs ranges from 0 to 0.5 while the range is 0 to 1 for the recall graphs.

c) In the precision graphs, some lines seem to be missing. This happens when two lines are exactly the same so they overlap.

d) Only the graphs that are considered to be the most important are shown. More graphs can be found in the appendices.

4.1 Query-based sampling

In order to measure the effects of query-based sampling with different numbers of samples, retrieval experiments are run without using clustering. Figure 4.1 shows the average recall over TREC topics 501-550 for the different number of samples. The graph clearly shows that on general the performance increases with the number of samples.

Figure 4.2 shows the precision of the same experiments. Again we can say that in general the performance is better when more samples are used. The difference between the precision scores for subsequent numbers of samples is small. At some points the precision is even better when a smaller number

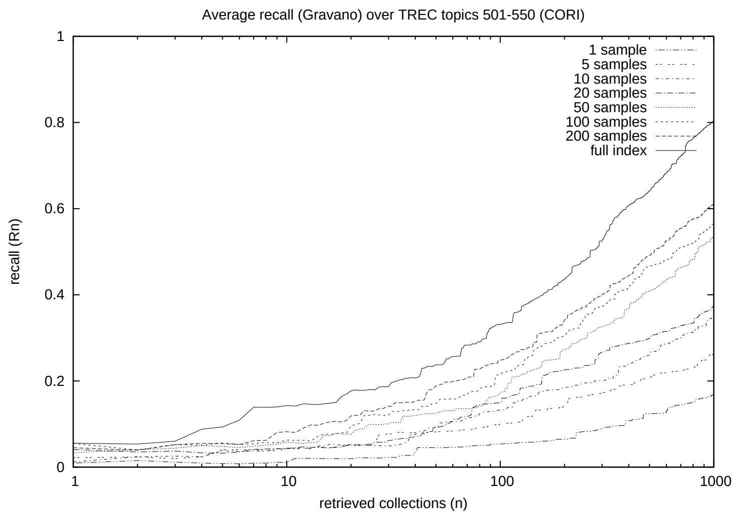

Figure 4.1: Average recall for different numbers of samples.

of samples is used. The explanation for this effect is the randomness of the samples that are taken. A smaller set of samples may contain documents that make a collection give a high rank while a larger set of samples may contain only documents that make the collection seem irrelevant and give it a low score. This effect is visible in the graphs because the random sampling phase was executed only once for every number of samples.

Although the full collection was indexed for more than 88% of the collections when 200 samples are taken, the difference in performance with the situation where the full index was used for all collections is still significant for both recall and precision at most points in the graphs. The exact recall and precision values and details about the significance of the results can be found in Tables D.1 and D.2 in Appendix D.

The results of these experiments are used as the baseline for the experiments that use clustering of collections. They were produced in a general distributed information retrieval setting. The lines in the graphs can also be found in the result graphs of the experiments that where based on clustering of collections, where they were added for comparison.

4.2 Cluster sizes and the number of clusters

When using k-means as a clustering algorithm, the number of produced clusters must be given as a parameter. The sizes of the clusters can vary. In Figure 4.3 we plot a histogram of the cluster sizes for 200 samples. The x-axis shows the sizes of the clusters, while the y-axis shows the percentage of clusters with this size. For all numbers of clusters, we see that there are

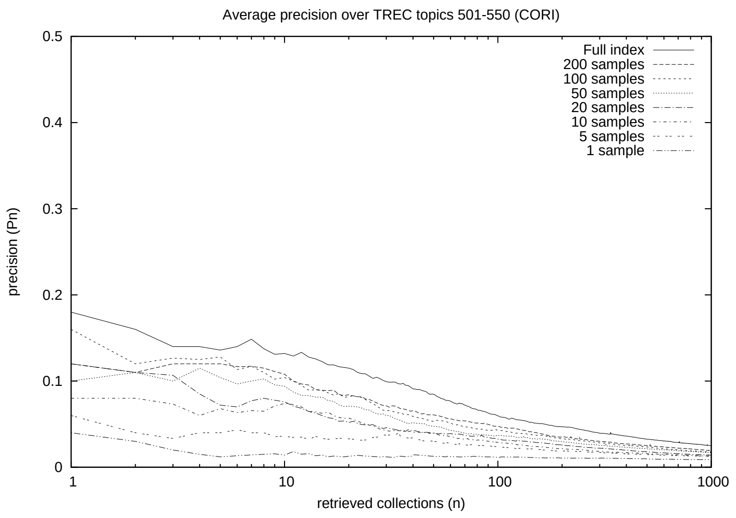

Figure 4.2: Average precision for different numbers of samples.

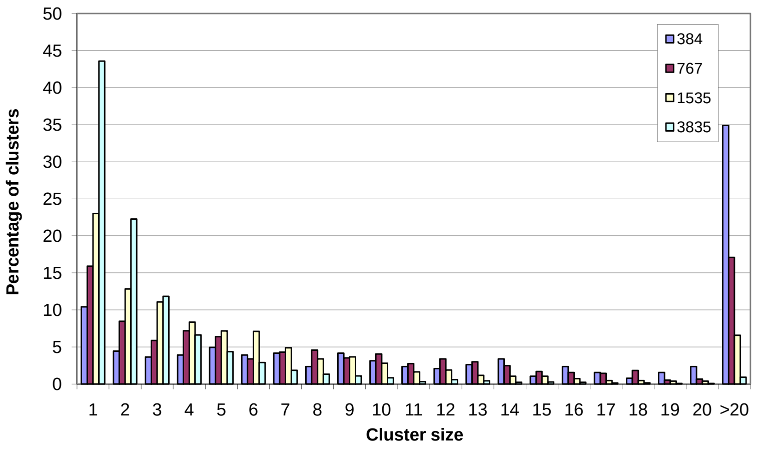

Figure 4.3: Percentage of clusters per cluster size (K-means, 200 samples).

relatively many clusters that contain only a single collection.

In our implementation of the bisecting k-means algorithm we define the size of the clusters that is produced. There are no hard cluster sizes used, but upper and lower bounds. Within these boundaries, the sizes of the clusters that are created may vary. The distribution of the cluster sizes for the clusterings based on 200 samples is plotted in Figure 4.4. In this figure, the y-axis shows the absolute number of clusters. In all 5 cases the number of clusters which size is equal to the minimum size is relatively large.

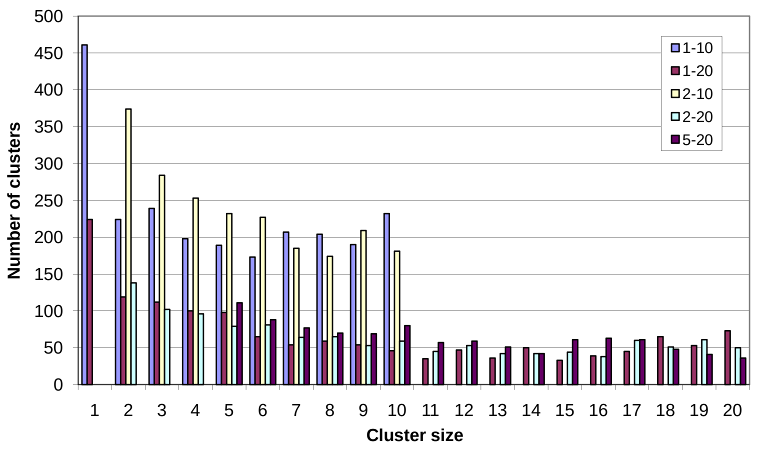

Figure 4.4: Number of clusters per cluster size (Bisecting k-means, 200 samples).

Collections that form a cluster by themselves can also be so called outliers. These collections are not similar to any other collections and therefore not grouped into a cluster with other collections.

The fact that there are relatively many small clusters might have a negative effect on the retrieval performance. A clustering with many small clusters is very similar to the situation where no clustering is performed at all.

Number of clusters for k-means In Figure 4.5 we compare the recall of the collection selection experiments using different numbers for k, the number of clusters that is produced by the k-means algorithm. The solid line is the graph of the experiment without using clustering. For the first 10 retrieved collections, we see that the recall is higher when clustering is used for all numbers of clusters. At this interval, the clusterings with 767 and 1,535 clusters perform best, while the clustering with 3,837 clusters follows almost the same line as the baseline graph without clustering. After 50 retrieved collections we see that a larger number of clusters leads to a higher recall.

The precision of the same experiments is shown in Figure 4.6. The dis-

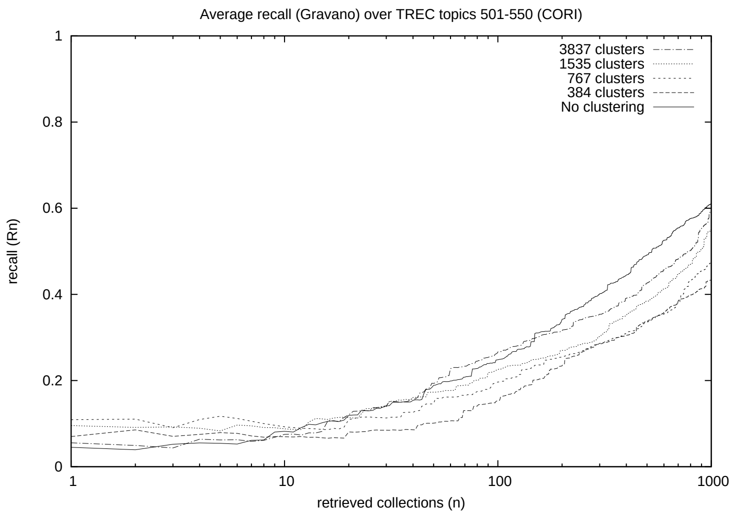

Figure 4.5: Average recall for different numbers of clusters (K-means, 200 samples, 5 iterations).

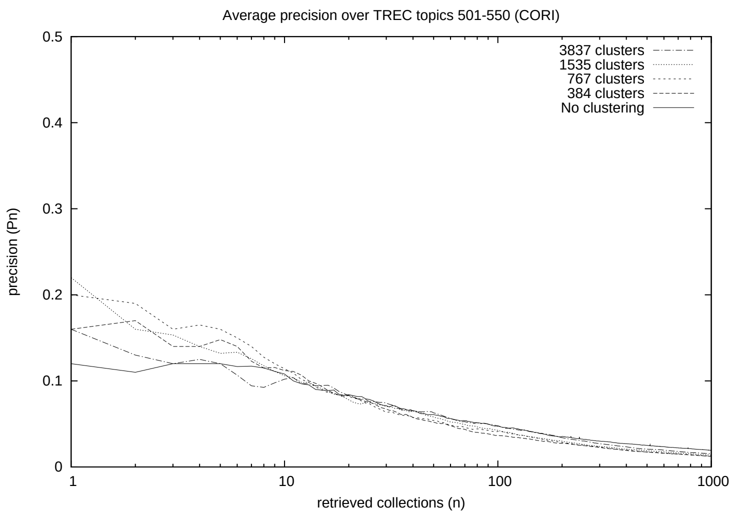

Figure 4.6: Average precision for different numbers of clusters (K-means, 200 samples, 5 iterations).

tinction between the different clusterings is less clear then for the recall graphs. For the first 10 retrieved collections the precision is higher than the baseline when clustering is used, except for the experiment with 3,837 clusters which only shows a small improvement for the first 4 retrieved collections. The clustering with 767 clusters has the best performance. The precision of the clustering with 3,837 clusters is close to the precision of the baseline so clustering with that many clusters has only a small influence on the precision.

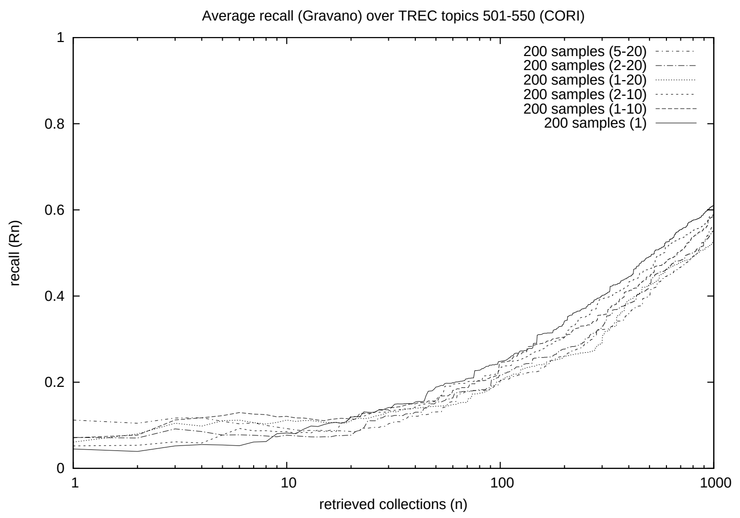

Figure 4.7: Average recall for different cluster sizes (Bisecting k-means, 200 samples).

Cluster sizes for bisecting k-means Figure 4.7 shows the recall of the experiments using clustering using bisecting k-means for 200 samples. The different graphs show the recall values for different boundaries of the cluster sizes. The solid line shows the recall when no clustering is used. This is the same graph as in Figure 4.1 and is used as the baseline.

The results show an increase in performance for the first 10 retrieved collections for the retrieval experiments where clustering is used. After 20 retrieved collections the recall for all cluster sizes drops below the baseline graph. The recall graphs for the different cluster sizes are close to each other and cross at some points. From this figure it is not possible to say which cluster size has the best performance.

The precision graphs of the same experiments are shown in Figure 4.8. The precision is higher when clustering is applied for the first 3 retrieved collections. After 10 collections are retrieved, the precision for all cluster sizes drops below the precision of the baseline. The lines of the different

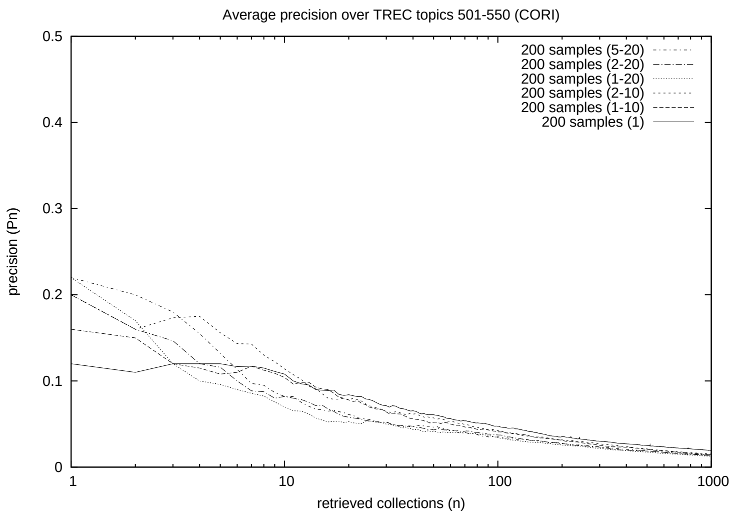

Figure 4.8: Average precision for different cluster sizes (Bisecting k-means, 200 samples).

cluster sizes cross a few times, so also from this figure it is not possible to determine a cluster size with the best performance.

For bisecting k-means there is no cluster size that produces consistently better results than other cluster sizes. We therefore conclude that the cluster size has no or only a small influence on the retrieval performance, at least for clusters with a minimum size of 1 and a maximum size of 20 collections.

4.3 Cluster quality

In section 3.3.1 we discussed some compromises we made to reduce the complexity of the calculations of the cluster algorithms. One of these compromises is a limit to the number of iterations for the k-means algorithm. We conducted experiments with different numbers of iterations to evaluate the effects of these compromises. Figure 4.9 shows the recall for the collection selection experiments using k-means clustering with different numbers of iterations.

For the experiments with 100 samples as well as the experiments with 200 samples we see that the results are just slightly different for different numbers of iterations. At some points in the graphs we see that the recall for the experiments with just one iteration is a little worse.

The precision graphs of the same experiments are shown in Figure 4.10. Although there is a little more variation in the precision for the first three retrieved collections, in general the results for different number of iterations are again really close to each other. Because the execution times of the

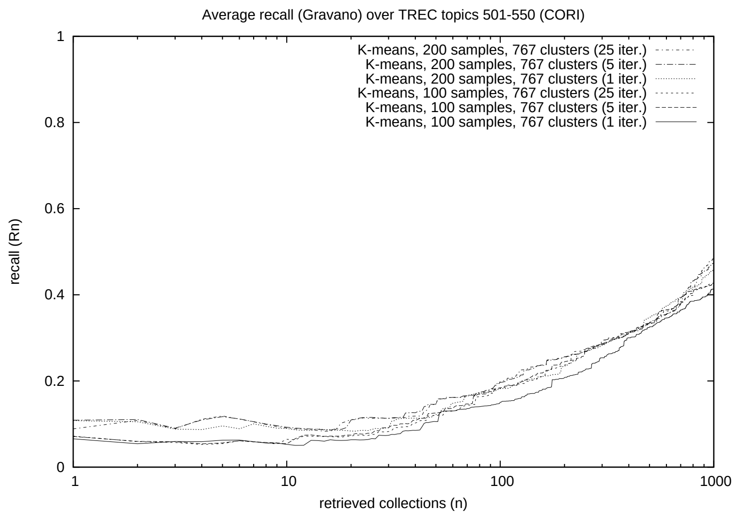

Figure 4.9: Average recall using k-means with different numbers of iterations.

cluster algorithm increases linearly with the number of iterations and we want to keep the necessary resources low, we use 5 iterations for the rest of the experiments. From these graphs we conclude that this will hardly affect the results.

4.4 Comparing cluster algorithms

This section compares the results of the k-means algorithm, the bisecting k-means algorithm and the baseline without clustering. The k-means graphs show the results of the experiment with 767 clusters and 5 iterations. We chose this number of clusters because it has the best performance as shown in the previous section. For the bisecting k-means algorithm there is no cluster size that performs best, so we include the graphs for the two cluster sizes that differ the most: 1-10 and 5-20.

The recall graphs for 200 samples are show in Figure 4.11. All graphs show an improvement in performance over the baseline for the first 10 retrieved collections. The recall improves significantly and increases 96% to 181% between 2 and 6 retrieved collections for all algorithms as can be seen in more detail in Table D.3 in Appendix D. The differences between the cluster algorithms is small. The recall graphs for 100 samples that can be found in Appendix C show less improvement, but improvement is still visible for the first 10 retrieved collections.

The precision graphs in Figure 4.12 also show some differences between the algorithms. The precision for bisecting k-means clustering with sizes between 5 and 10 is the highest for the first retrieved collection, but drops

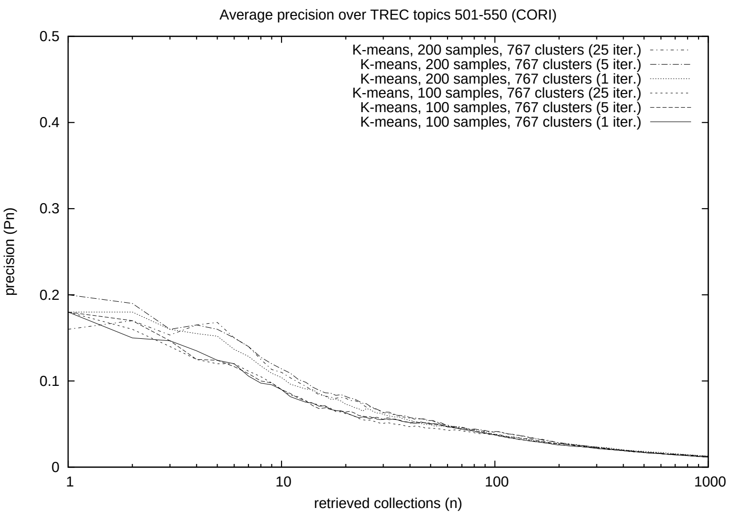

Figure 4.10: Average precision using k-means with different numbers of iterations.

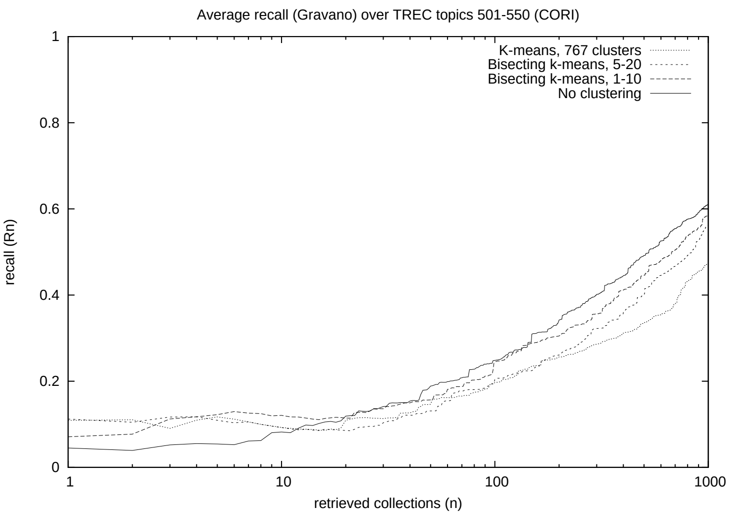

Figure 4.11: Average recall using clustering (200 samples).

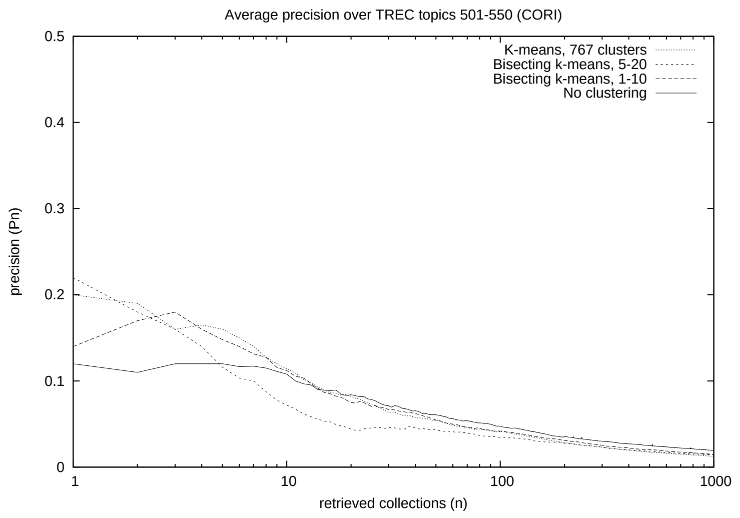

Figure 4.12: Average precision using clustering (200 samples).

quickly when more collections are retrieved. When more than 5 collections are retrieved, the precision score is lower than for the situation without clustering. The bisecting k-means clustering with cluster sizes in the range from 1 to 20 has a low precision for the first retrieved collection, but the precision increases for the next few collections. The k-means algorithm has a high precision and it does not increase or decrease as much as the precision for the bisecting k-means algorithm. The results for the k-means algorithm are more stable. For all algorithms there is only significant improvement in precision for the first 4 retrieved collections.

4.5 Query sets

The experiments for which the results are described so far are all conducted using the TREC10 topics 501-550. To check the validity of the results, the same experiments are conducted using a different query set, the TREC9 topics 451-500. As described in section 3.1 the main difference between the two query sets is that the latter contains spelling errors. The experiments with this query set are only conducted for the bisecting k-means algorithm.

The precision of the retrieval experiments results for different numbers of samples without clustering is shown in Figure 4.13. The precision graphs for the different numbers of samples are closer to each other than the graphs for topics 501-550. The precision for the experiments using the full index and 200 samples drops dramatically, but the precision for 1 and 5 samples increases. Also note that the precision for the first retrieved collection is in some cases better when an incomplete index was used. The explanation

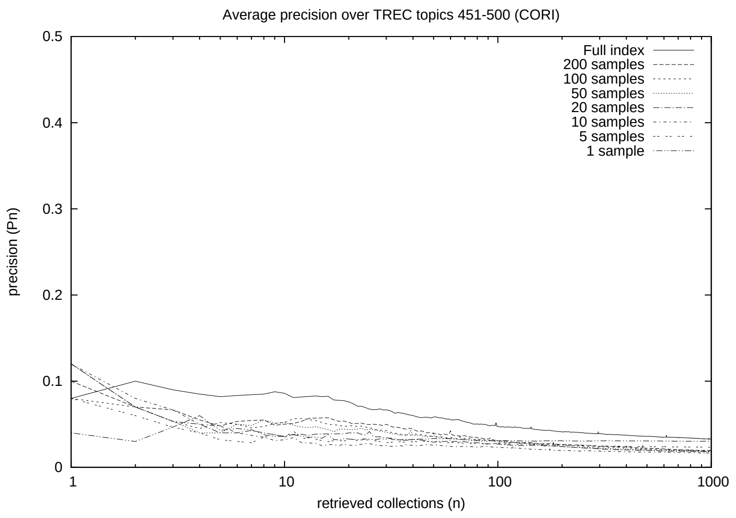

Figure 4.13: Average precision for different numbers of samples for TREC topics 451-500.

of these effect is that the collection selection algorithm does not have any methods implemented to detect and handle spelling errors.

4.6 Summary

This chapter described the various collection selection experiments that were conducted. The focus of these experiments is query-based sampling and clustering.

Query-based sampling has a direct influence on the collection selection performance. When incomplete indexes are used as the effect of query-based sampling, the performance drops significantly. The results also show that on general, the more samples are taken, the higher the performance.

For both k-means as bisecting k-means, analysis of the distribution of the cluster sizes shows that there are relatively many clusters that are exactly the size of the lower boundary.

The results of the experiments with different numbers of clusters for k-means show that for all used cluster sizes both recall and precision increases. The clusterings with 767 and 1535 clusters produce the best results for collection selection. For bisecting k-means there is no cluster sizes that produces consistently better results than other cluster sizes. When this algorithm is used, the cluster size seems to have no or only a small influence on the retrieval performance.

Clustering of collections can have a positive effect on the collection selection results. In most experiments we see an increase of the recall for the

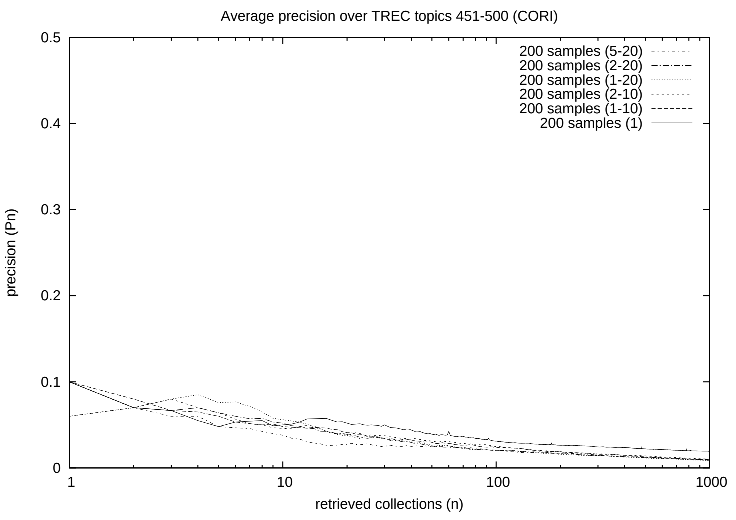

Figure 4.14: Average precision over TREC topics 451-500 using clustering (200 samples).

first 10 to 20 retrieved collections. For the collections that are retrieved after that, the recall is higher when no clustering is applied. The precision graphs also show improvement when clustering is used, but in general only for the first 4 to 10 retrieved collections.

The difference between the results for k-means and bisecting k-means are small. However, the scores for bisecting k-means tend to change a lot when the number of retrieved collections increases, in particular the precision. The scores for k-means are more stable.

The collection selection experiments with bisecting k-means are conducted a second time with a different query set to check the validity of the results. This query set contains spelling errors, causing both recall and precision scores to be considerably lower. However, the positive effect of clustering is still perceptible.

Chapter 5

Conclusion

This research focuses on using collection clustering to improve the performance of collection selection in distributed information retrieval systems. In the system that is used for this research the collections are non-cooperative, which means that the resource descriptions can be created using the default search interfaces. We simulated query-based sampling and conducted experiments to research the effect of incomplete resource descriptions. We clustered document collections using two different cluster algorithms and used a well known collection selection algorithm to conduct retrieval experiments. This chapter describes the conclusions that can be drawn from the results of the experiments by answering the research questions.

5.1 Collection selection algorithms

Research question 1: Which collection selection algorithm can be used in combination with collection clustering for this research?

A number of collection selection algorithms were described for use in the experiments that are conducted during this research. These algorithms are GlOSS, CORI, CVV and Indri. We chose to use CORI for various reasons. It is a well researched algorithm which proved to perform well in various situations. Further it is possible to use the algorithm for collection selection based on clustering because it represents collections as large documents and can handle clusters of collections the same way. This means that no modifications are necessary to make the algorithm work for clusters instead of collections. CORI is implemented in the Lemur toolkit for information retrieval so there was no need for a custom implementation.

5.2 Incomplete resource descriptions

Research question 2: How big is the problem of incomplete resource descriptions?

Collection selection experiments were conducted using different numbers of samples for query-based sampling. The results of these experiments show that there is a clear relation between the number of samples and the collection selection performance. The more samples are used for creating the resource descriptions, the better the performance. Even when 88% of the collections are fully indexed, performance is still significantly worse than when all collections are fully indexed.

In practice it is both difficult and inefficient to create resource descriptions that are based on the full content of the collections when query-based sampling is used. The collection selection performance for incomplete resource descriptions must be improved to reach a performance comparable to the performance for complete resource descriptions.

5.3 Measuring collection selection performance

Research question 3: How can the performance of different collection selection algorithms be measured and compared?

In this research a number of collection selection experiments are conducted. The corpus that was available for testing was the WT10G corpus containing 1.69 million documents with a total size of 10GB. This corpus was not created for distributed information retrieval and needed some preprocessing for this purpose. The corpus was split into collections based on the servers IP addresses, creating 11,512 separate data collections. Two sets of queries with relevance judgements are available for this dataset.

Two measures are selected for the evaluation of the results. These measures are similar to the recall and precision measures that are commonly used in traditional information retrieval. The recall measure that is used in this research is a measure of how much of the available merit in the top n ranked collections has been accumulated by the ranking under evaluation. The precision measure gives the fraction of the retrieved collections that has a non-zero merit.

5.4 Cluster methods

Research question 4: What is the best technique for collection clustering?

In this research the collections needed to be clustered based on the topical content of the documents in the collections.

We used two cluster algorithms to do this: the widely used k-means algorithm and the bisecting k-means algorithm. Bisecting k-means is a hierarchical cluster algorithm which is based on the k-means algorithm. When using k-means, the number of produced clusters must be known when clustering starts. There is no control over the sizes of the clusters. We adjusted the bisecting k-means algorithm which makes it possible to specify the lower and upper bound of the size of the produced clusters. When this algorithm is used, the number of clusters is not known until clustering is finished.

We conducted experiments with different numbers of clusters and different cluster sizes. For k-means the best results are achieved when 767 and 1535 clusters are created. In these cases the average cluster sizes are respectively 15 and 7.5. For bisecting k-means the results are inconclusive. From the results it is not possible to tell what is the optimal cluster size.

From the experiments we conducted we conclude that k-means is the most reliable cluster algorithm, but the difference with bisecting k-means is small. Our implementation of the bisecting k-means algorithm has the advantage that the number of clusters must not be specified as a parameter. The computational complexity of bisecting k-means is lower which makes it more scalable than k-means.

In the implementation of the clustering algorithms some compromises are made to reduce the complexity. The most important compromise is limiting the number of iterations for k-means. Using a small number of iterations results in a poor quality of clustering. The results show that this limit has only a small influence on the collection selection performance. Setting the limit higher than 5 increases the performance only marginally.

5.5 Using clustering to improve collection selection

Research question 5: What are the effects of clustering on the collection selection performance?

The collection selection experiments show an improvement when clustering of collections is applied, but only for the first few retrieved collections. The recall generally improves for the first 10 to 20 retrieved collections, the precision improves for the first 4 to 10 retrieved collections. After retrieving more collections the performance drops below the performance of collection selection without clustering.

In practice a distributed information retrieval system will only send the query to the collections that were ranked highest in the collection selection process. The top ranked collections are therefore the most important. The improvements we show affect these top ranked collections.

From the results we conclude that the cluster sizes have no or only a minor effect on the retrieval performance. There is no cluster size that performs consistently better and the lines in the graphs often cross. This holds at least for cluster that contain between 1 and 20 collections.

5.6 Summary

Main research question: Can the clustering of collection improve the performance of collection selection methods for distributed web search?

We have shown that clustering of collections can improve the collection selection performance. The results show an increase of precision for the first 4 to 10 retrieved collections, and and increase of recall for the first 10 to 20 retrieved collections. Using the k-means algorithm results in a slightly higher performance than using the bisecting k-means algorithm, but because of the time complexity of this algorithm it does not scale to a large number of collections. For that reason it is better to use bisecting k-means for which the collection selection results are almost as good.

We proposed the collection cluster hypothesis which we defined as:

Closely associated collections tend to contain documents that are relevant to the same requests.

We clustered closely associated collections so that relations between collections are taken into account during collection selection. Collections can get a higher rank because of their relations to other collections. The results of the collection selection experiments prove that this leads to higher performance, and therefore we conclude that the collection cluster hypothesis holds.

Chapter 6

Future work

This research shows that collection selection based on resource clustering can improve the retrieval performance. More research is needed to prove the applicability of the described collection selection method in other situations. The increases in retrieval performance that are shown are clear, but minimal. We expect that performance can increase further when better cluster algorithms are used.

6.1 Replacing CORI

In the experimental setup in this research we chose to use CORI as the underlying collection selection algorithm. Previous research [46, 33] has shown that more recently proposed algorithms can outperform CORI. More research is needed to evaluate the effects of clustering on other collection selection algorithms. We expect that combining clustering with other collection selection algorithms will also increase the performance.

6.2 Conducting experiments using different corpora

All experiments are conducted using the WT10G corpus. This corpus was not created for distributed information retrieval. Splitting the corpus into collection by server IP address allows to simulate a distributed information retrieval environment. In the corpus split we used there is no overlap between the collections. Every document occurs in only one collection. In other scenarios overlap between collections may occur. Many of the produced collections are small, causing whole collections to be sampled during the query-based sampling step, even when only a small number of samples is taken. Conducting the same experiments on other corpora can give a better view of the influence of the number of collections and the collection sizes. Using different methods to create corpus splits can show the effects of overlap between collections.

6.3 Improving performance

In most of the experiments we saw an increase in collection selection performance for the first 3 to 20 retrieved collections. When more collections are retrieved, the performance drops below the performance without using clustering. Increasing the performance for a higher number of retrieved collections is highly desirable.

The scoring function we use to calculate a ranking score from the cluster scores and collection scores is very simple. We did not experiment with using different functions. It is possible to weigh the cluster scores more or less than the collection scores. By changing this function it may be possible to create better rankings.

Bibliography

[1] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments, 2001.

[2] Protima Banerjee and Hyoil Han. Drexel at trec 2007: Question answering. In *TREC*, 2007.

[3] Ziv Bar-Yossef and Maxim Gurevich. Random sampling from a search engine's index. *Proc. 15th WWW*, pages 367–376, May 2006.

[4] Mayank Bawa, Gurmeet Singh Manku, and Prabhakar Raghavan. Sets : Search enhanced by topic segmentation. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 306–313, 2003.

[5] Krishna Bharat and Andrei Broder. Estimating the relative size and overlap of public web search engines. In *7th International World Wide Web Conference (WWW7*, pages 512–523. Springer, 1998.

[6] S. Bockting. Collection selection for distributed web search : using highly discriminative keys, query-driven indexing and colrank, February 2009.

[7] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The inquery retrieval system. In *In Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83. Springer-Verlag, 1992.

[8] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, 19(2):97–130, April 2001.

[9] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. *SIGMOD Rec.*, 28(2):479–490, 1999.

[10] Jamie Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collection with inference networks. *Proceedings of the 18th annual in-*

ternational ACM SIGIR conference on Research and development in information retrieval, pages 21–28, 1995.

[11] Jamie Callan, Allison L. Powell, James C. French, and Margaret Connell. The effects of query-based sampling on automatic database selection algorithms. *Technical Report CMU-LTI-00-162, Language Technologies Institute, School of Computer Science, Carnegie Mellon University.*, 2000.

[12] James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the performance of database selection algorithms. *SIGIR'99*, 1999.

[13] Luis Gravano and Héctor García-Molina. Generalizing gioss to vector-space databases and broker hierarchies. *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 78–89, 1995.

[14] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems*, 24(2):229–264, June 1999.

[15] Luis Gravano, Panagiotis G. Ipeirotis, and Mehran Sahami. Qprober: A system for automatic classification of hidden-web databases. *ACM Transactions on Information Systems*, 21(1):1–41, January 2003.

[16] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. *Proc. 14th WWW (Posters)*, pages 902–903, 2005.

[17] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Communications of the ACM*, 50(5):94–101, May 2007.

[18] Panagiotis G. Ipeirotis and Luis Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. *Proceedings of the 28th VLDB Conference*, 2002.

[19] Panagiotis G. Ipeirotis and Luis Gravano. Classification-aware hidden-web text database selection. *ACM Transactions on Information Systems (TOIS)*, 26(2), March 2008.

[20] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hiddenweb databases. *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 67–78, 2001.

[21] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5):217 – 240, 1971.

[22] Narayanan Kulathuramaiyer and Wolf-Tilo Balke. Restricting the view and connecting the dots - dangers of a web search engine monopoly. *Journal of Universal Computer Science*, 12(12):1731–1740, 2007.

[23] Yanjun Li and Soon Myoung Chung. Parallel bisecting k-means with prediction clustering algorithm. *The Journal of Supercomputing*, 39(1):19–37, 2007.

[24] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[25] Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys (CSUR)*, 34(1):48–89, March 2002.

[26] Peter Merz. An iterated local search approach for minimum sum-of-squares clustering. In *IDA*, pages 286–296, 2003.

[27] Donald Metzler. Indri retrieval model overview. *http://ciir.cs.umass.edu/m̃etzler/indriretmodel.html, retrieved on 20 May 2010*, July 2005.

[28] Donald Metzler and W. Bruce Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735 – 750, 2004. Bayesian Networks and Information Retrieval.

[29] Mark E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 47(5):323–351, September 2005.

[30] Henrik Nottelmann and Norbert Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[31] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. pages 275–281, 1998.

[32] R. Rousseau and Q. Zhang. Zipf's data on the frequency of chinese words revisited. *Scientometrics*, 24(2):201–220, 1992.

[33] Jangwon Seo and W. Bruce Croft. Umass at trec 2007 blog distillation task. In *TREC*, 2007.

[34] Jacky K. H. Shiu, Stephen C. F. Chan, and Korris F. L. Chung. Accessing hidden web documents by metasearching a directory of specialty search engines. *Databases in Networked Information Systems, Proceedings*, 2822:27–41, 2003.

[35] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. *Proceedings of the 26th International ACM SIGIR on Information Retrieval*, pages 298–305, 2003.

[36] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632, New York, NY, USA, 2007. ACM.

[37] Ian Soboroff. Does wt10g look like the web? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–424, New York, NY, USA, 2002. ACM.

[38] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD'00)*, pages 109–110, August 20–23 2000.

[39] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. Indri: a language-model based search engine for complex queries. Technical report, in Proceedings of the International Conference on Intelligent Analysis, 2005.

[40] Paul Thomas and David Hawking. Evaluating sampling methods for uncooperative collections. *Proceedings of the 30th International ACM SIGIR on Information Retrieval*, pages 503–510, 2007.

[41] A. S. Tigelaar and D. Hiemstra. Query-based sampling: Can we do better than random? Technical Report TR-CTIT-10-04, Enschede, February 2010.

[42] Howard Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991.

[43] C. J. van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann Ltd, 2nd edition, March 1981.

[44] Sholom Weiss, Brian White, and Chid Apte. Lightweight document clustering, 2000.

[45] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 254–261, 1999.

[46] Xing Yi and James Allan. Indri at trec 2007: Million query (1mq) track. 2007.

[47] Budi Yuwono and Dik Lun Lee. Server ranking for distributed text retrieval systems on the internet. *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, pages 41–50, April 1997.

[48] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

Appendix A

Bisecting k-means recall and precision

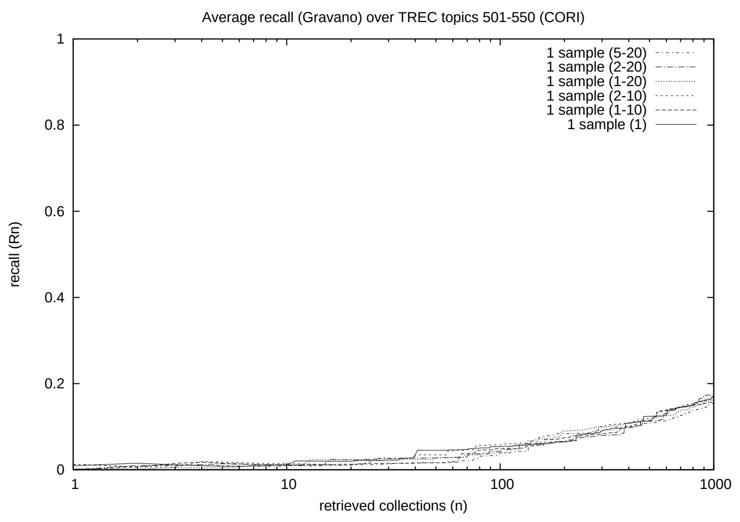

Figure A.1: Average recall using clustering (1 sample).

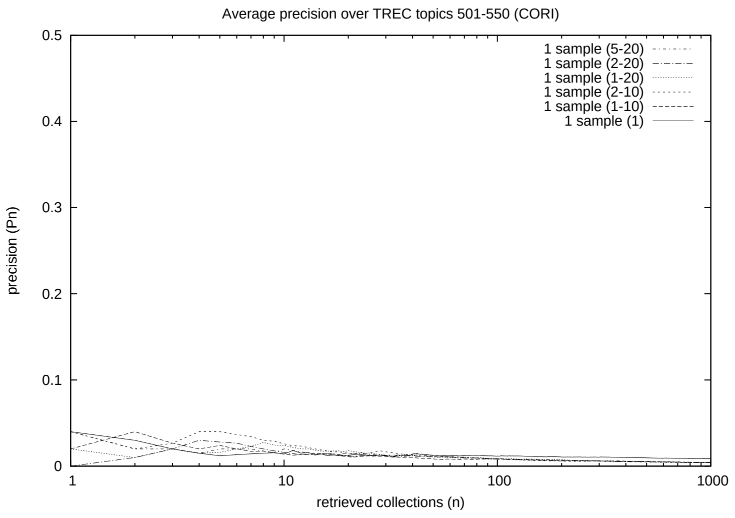

Figure A.2: Average precision using clustering (1 sample).

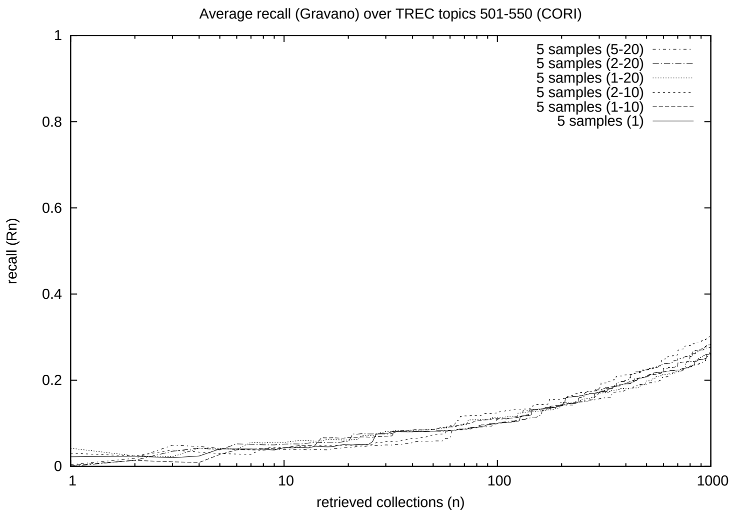

Figure A.3: Average recall using clustering (5 samples).

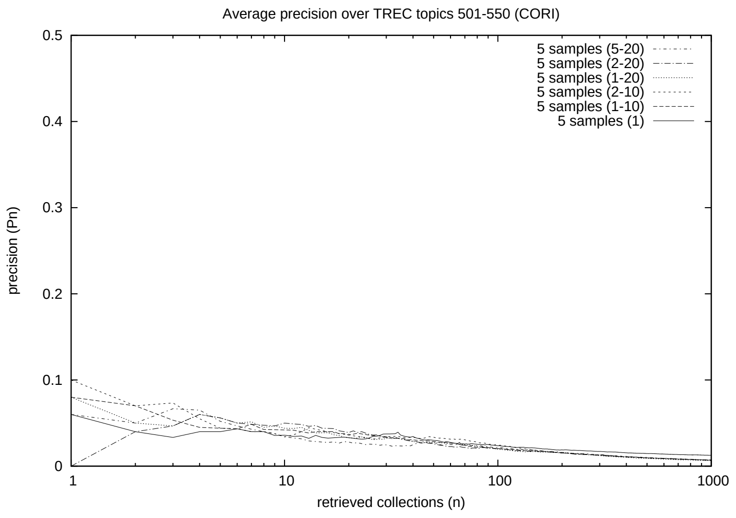

Figure A.4: Average precision using clustering (5 samples).

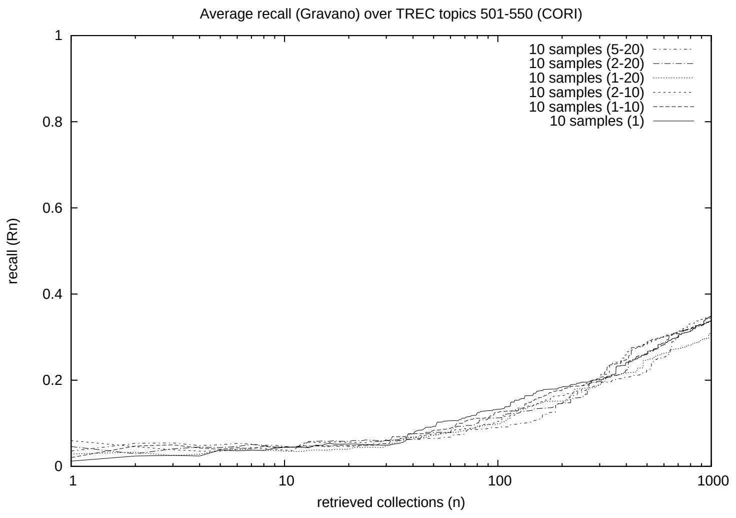

Figure A.5: Average recall using clustering (10 samples).

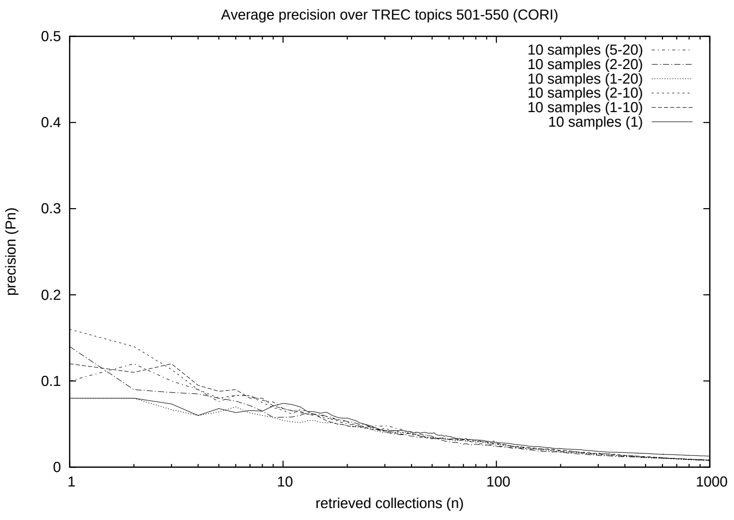

Figure A.6: Average precision using clustering (10 samples).

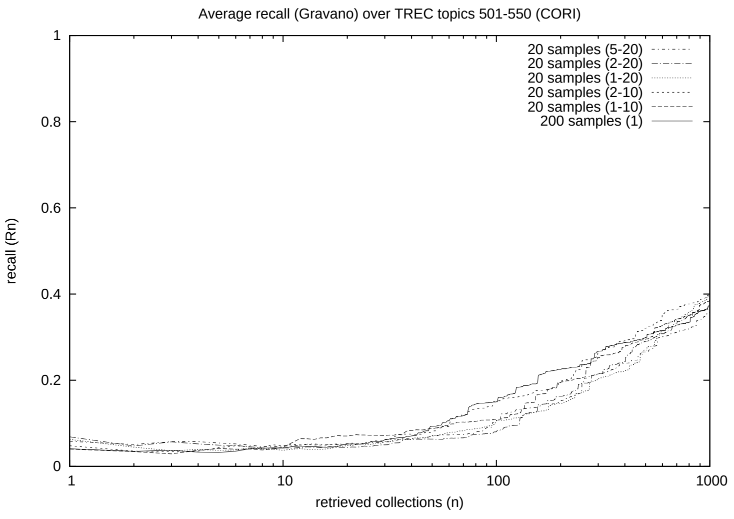

Figure A.7: Average recall using clustering (20 samples).

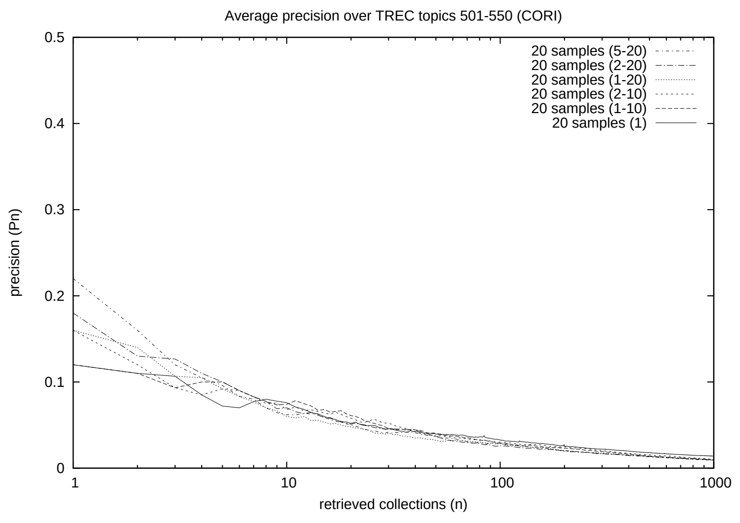

Figure A.8: Average precision using clustering (20 samples).

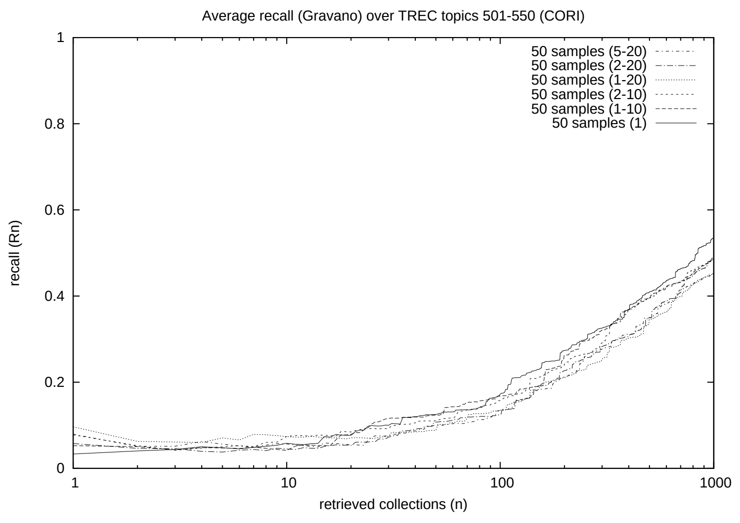

Figure A.9: Average recall using clustering (50 samples).

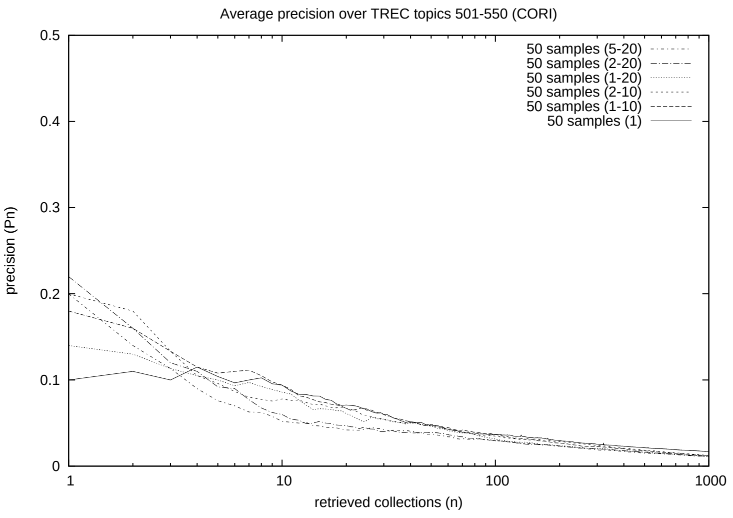

Figure A.10: Average precision using clustering (50 samples).

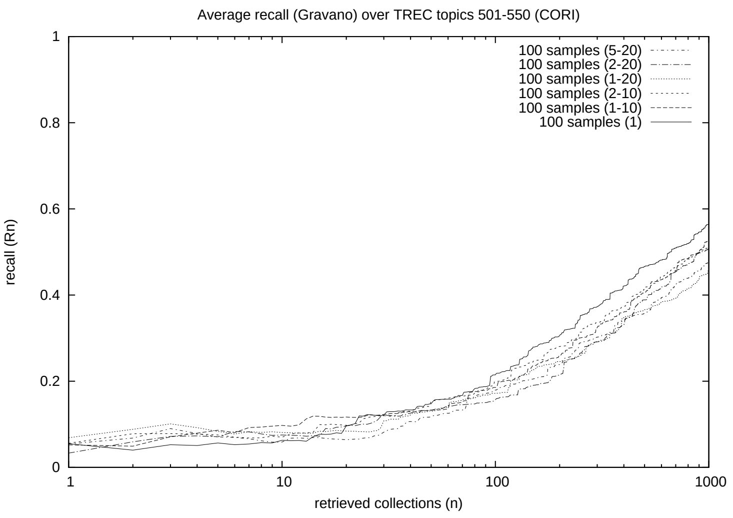

Figure A.11: Average recall using clustering (100 samples).

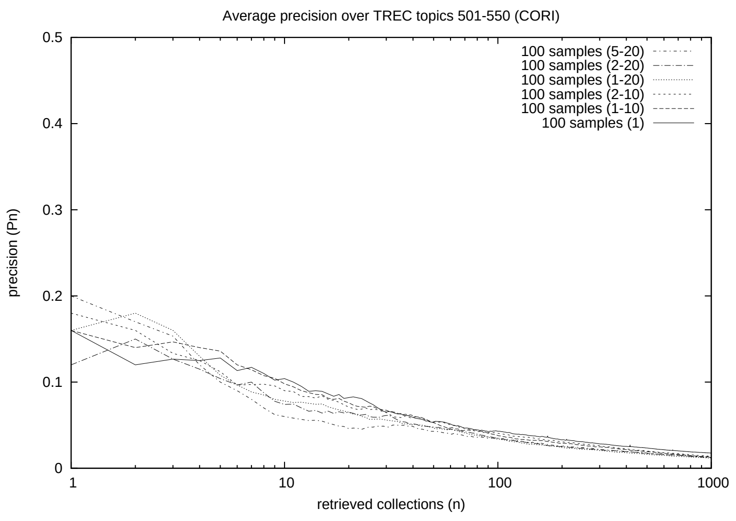

Figure A.12: Average precision using clustering (100 samples).

Appendix B

K-means recall and precision

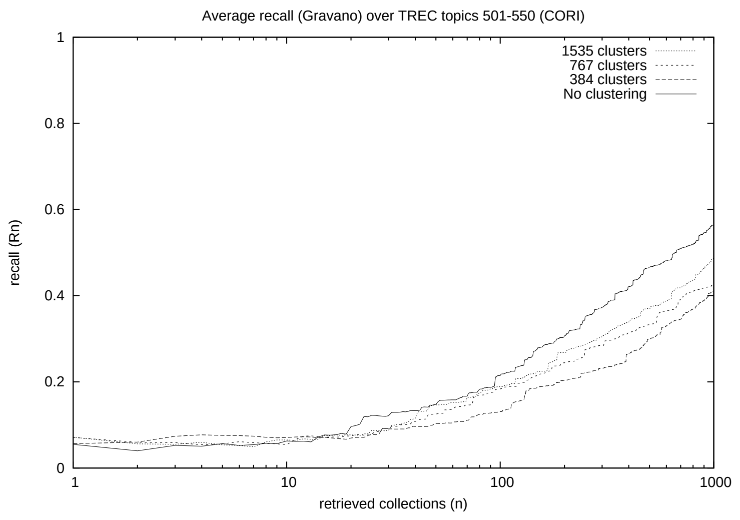

Figure B.1: Average recall for different numbers of clusters (K-means, 100 samples, 5 iterations).

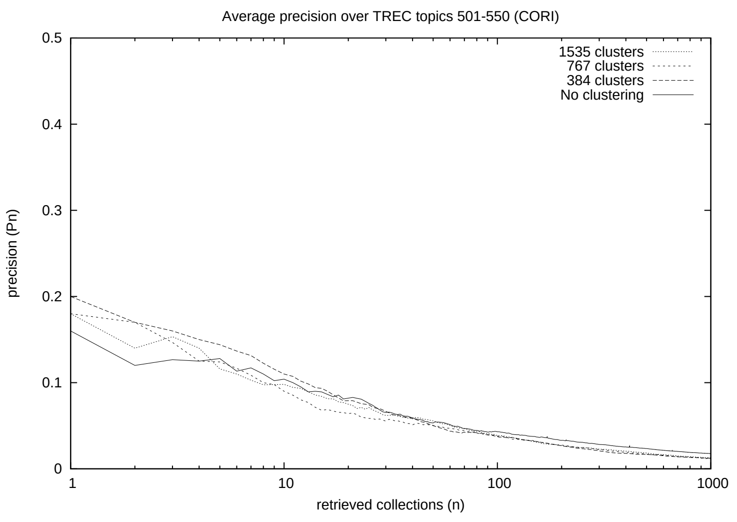

Figure B.2: Average precision for different numbers of clusters (K-means, 100 samples, 5 iterations).

Appendix C

K-means and bisecting k-means

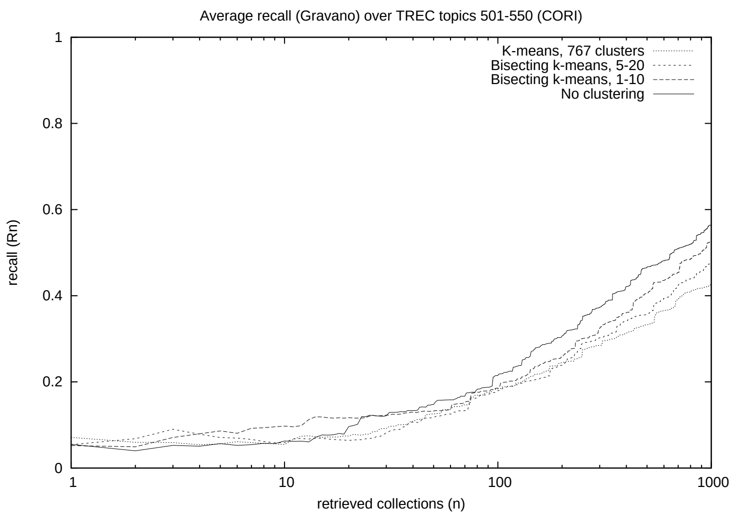

Figure C.1: Average recall using clustering (100 samples).

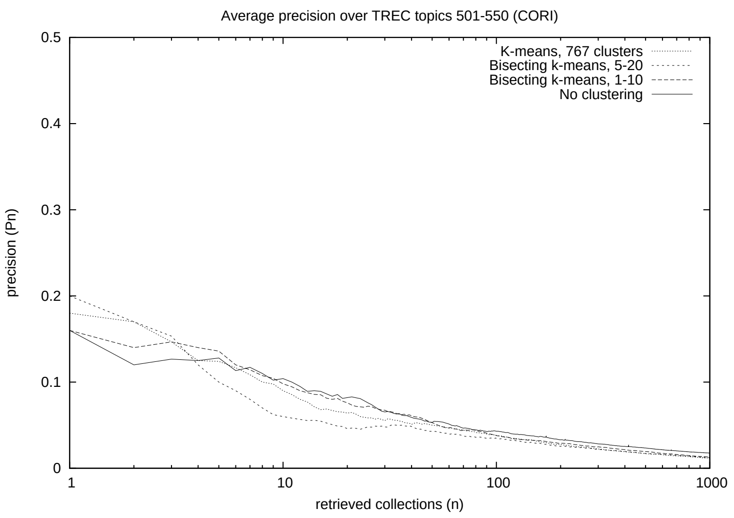

Figure C.2: Average precision using clustering (100 samples).

Appendix D

Recall and precision values

The tables in this appendix show a subset of the recall and precision values from which the graphs in Figures 4.1, 4.2, 4.11 and 4.12 were created. Values that are printed in bold are significance differences from the baseline algorithm. The significance is tested using the Student's paired t-test as described in Section 3.5.

Table D.1: Recall values for different numbers of samples.

Retrieved coll.	1	2	3	4	5	6	7	8	9	10	50	100
No clustering	0.056	0.053	0.060	0.088	0.093	0.109	0.139	0.139	0.140	0.143	0.237	0.331
Qbs200	*0.045*	*0.039*	*0.052*	***0.055***	***0.054***	***0.053***	***0.061***	***0.062***	***0.080***	***0.082***	***0.189***	***0.248***
Qbs100	*0.055*	*0.040*	*0.053*	***0.051***	***0.057***	***0.053***	***0.054***	***0.057***	***0.057***	***0.063***	***0.148***	***0.215***
Qbs50	***0.033***	*0.040*	*0.044*	***0.050***	***0.048***	***0.045***	***0.049***	***0.051***	***0.054***	***0.058***	***0.125***	***0.173***
Qbs20	*0.041*	*0.035*	*0.037*	***0.033***	***0.032***	***0.035***	***0.041***	***0.042***	***0.043***	***0.042***	***0.093***	***0.150***
Qbs10	***0.012***	*0.024*	***0.025***	***0.024***	***0.038***	***0.036***	***0.038***	***0.037***	***0.042***	***0.044***	***0.093***	***0.133***
Qbs5	***0.022***	***0.024***	***0.020***	***0.024***	***0.040***	***0.041***	***0.041***	***0.039***	***0.038***	***0.044***	***0.082***	***0.100***
Qbs1	***0.009***	***0.015***	***0.011***	***0.010***	***0.008***	***0.008***	***0.009***	***0.009***	***0.011***	***0.011***	***0.045***	***0.054***

Table D.2: Precision values for different numbers of samples.

Retrieved coll.	1	2	3	4	5	6	7	8	9	10	50	100
No clustering	0.180	0.160	0.140	0.140	0.136	0.140	0.149	0.138	0.131	0.132	0.085	0.060
Qbs200	***0.120***	***0.110***	*0.120*	*0.120*	*0.120*	*0.117*	***0.117***	***0.115***	***0.111***	***0.108***	***0.061***	***0.047***
Qbs100	*0.160*	*0.120*	*0.140*	*0.135*	*0.136*	*0.120*	***0.123***	***0.115***	***0.107***	***0.108***	***0.055***	***0.044***
Qbs50	*0.100*	*0.130*	*0.113*	*0.125*	*0.112*	***0.103***	***0.106***	***0.108***	***0.100***	***0.098***	***0.050***	***0.038***
Qbs20	*0.120*	*0.110*	*0.107*	***0.085***	***0.072***	***0.070***	***0.077***	***0.080***	***0.078***	***0.076***	***0.040***	***0.033***
Qbs10	***0.080***	***0.080***	***0.073***	***0.060***	***0.068***	***0.063***	***0.066***	***0.065***	***0.071***	***0.074***	***0.039***	***0.029***
Qbs5	***0.060***	***0.040***	***0.033***	***0.040***	***0.040***	***0.043***	***0.040***	***0.040***	***0.036***	***0.036***	***0.030***	***0.024***
Qbs1	***0.040***	***0.030***	***0.020***	***0.015***	***0.012***	***0.013***	***0.014***	***0.015***	***0.016***	***0.014***	***0.013***	***0.012***

Bold printed numbers are significant differences from the baseline algorithm.
Numbers in italic show a decrease in performance.

Table D.3: Recall values for different cluster algorithms, 200 samples.

Retrieved collections	1	2	3	4	5	6	7	8	9	10	50	100
No clustering	0.045	0.039	0.052	0.055	0.054	0.053	0.061	0.062	0.080	0.082	0.189	0.248
K-means, 767	**0.109**	**0.111**	**0.091**	**0.110**	**0.117**	**0.112**	**0.106**	**0.100**	0.096	0.093	***0.146***	***0.197***
Bisecting, 5-20	0.112	**0.105**	**0.117**	**0.117**	**0.111**	**0.103**	0.106	0.100	0.096	0.092	***0.131***	***0.205***
Bisecting, 1-10	0.071	**0.077**	**0.112**	**0.117**	**0.122**	**0.130**	**0.126**	**0.125**	**0.120**	**0.121**	***0.157***	*0.244*

Table D.4: Precision values for different cluster algorithms, 200 samples.

Retrieved collections	1	2	3	4	5	6	7	8	9	10	50	100
No clustering	0.120	0.110	0.120	0.120	0.120	0.117	0.117	0.115	0.111	0.108	0.061	0.047
K-means, 767	0.200	**0.190**	0.160	**0.165**	0.160	0.150	0.140	0.128	0.120	0.114	*0.054*	*0.041*
Bisecting, 5-20	**0.220**	**0.180**	0.160	0.140	*0.116*	*0.103*	*0.100*	*0.088*	***0.078***	***0.072***	***0.043***	***0.035***
Bisecting, 1-10	0.140	**0.170**	**0.180**	**0.160**	0.148	0.140	0.131	0.128	0.116	0.112	***0.055***	***0.042***

Bold printed numbers are significant differences from the baseline algorithm.
Numbers in italic show a decrease in performance.