LANGUAGE MODELS FOR HIERARCHICAL SUMMARIZATION

A Dissertation Presented

by

DAWN J. LAWRIE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2003

Computer Science

LANGUAGE MODELS FOR HIERARCHICAL SUMMARIZATION

A Dissertation Presented

by

DAWN J. LAWRIE

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

Donald Fisher, Member

Arnold Rosenberg, Member

W. Bruce Croft, Department Chair
Computer Science

EPIGRAPH

To my family, and especially my husband,

for supporting me throughout this endeavor.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my committee: Bruce Croft, James Allan, Donald Fisher, and Arnold Rosenberg for taking the time to evaluate my dissertation. In particular, I would like to thank my advisor, Bruce Croft, for his guidance during the development of this work, and without whose help this thesis would not have been possible. I would also like to thank Arny Rosenberg for his mentoring and support during my graduate career. I would like to thank James Allan for his advice and perspective both inside and outside of research. The contributions of all are very much appreciated.

I would also like to acknowledge the support and encouragement from my family. I would like to give special thanks to my mother, mother-in-law, brother, and husband for their contribution to my user study. I would also like to thank my husband for his superb job in editing this paper.

Thanks must be given, finally, to the CIIR faculty, students, and staff for their insightful discussions, contributions of software, and administrative support. I would especially like to thank Alvaro Bolivar for his help in developing the interface for the web demo.

This work was supported in part by the Center for Intelligent Information Retrieval, the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, in part by the National Science Foundation under grant number CCR-00-73401, in part by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling in Information Retrieval Research Program, contract number MDA904-00C-2106, in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235, in part by the United States Patent

and Trademark Office, and in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-18903. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor.

ABSTRACT

LANGUAGE MODELS FOR HIERARCHICAL SUMMARIZATION

SEPTEMBER 2003

DAWN J. LAWRIE

A.B., DARTMOUTH COLLEGE

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Hierarchies have long been used for organization, summarization, and access to information. In this dissertation we define summarization in terms of a probabilistic language model and use this definition to explore a new technique for automatically generating topic hierarchies. We use the language model to characterize the documents that will be summarized and then apply a graph-theoretic algorithm to determine the best topic words for the hierarchical summary. This work is very different from previous attempts to generate topic hierarchies because it relies on statistical analysis and language modeling to identify descriptive words for a document and organize the words in a hierarchical structure.

We compare our new technique to previous methods proposed for constructing topic hierarchies, including subsumption and lexical hierarchies. We also compare the words chosen to be part of the hierarchy to the top ranked words using TF.IDF in terms of how well each summarizes the document set. Our results show that the language modeling approach

performs as well as or better than these other techniques in non user-based evaluations. We also show that the hierarchies provide better access to the documents described in the summary than does a ranked list using one of the non-user based evaluations we have developed. In a user study that compares the ability of users to find relevant instances using both the hierarchy and a ranked list to using the ranked list alone, we find that users like the information provided by the hierarchy and after some practice can use it as effectively as they can a ranked list.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

In this chapter we introduce word-based summaries, the topic of this dissertation. We first present the motivation for automatically generating hierarchical word-based summaries in Section 1.1. We then describe the benefits of constructing a hierarchy in Section 1.2. In Section 1.3 we introduce our approach to generating summaries with a discussion of how a person might approach the task. In Section 1.4 we provide motivation for the use of language models and relative entropy as a basis for creating the summaries. In Section 1.5 we explain how language modeling and relative entropy are used to produce hierarchical word-based summaries. Finally, we conclude with the research contributions of this thesis in Section 1.6 and an organization of the remainder of the thesis in Section 1.7.

1.1 Motivation

A large amount of text is readily available in electronic form in today's world. However, simply because a user has access to this information does not mean that it is easily found. In fact, it seems that the larger the volume of information available, the smaller the chance that a given person will be able to find the particular information that she is looking for in the deluge of irrelevant data. Therefore, a better way to access this information is needed.

State-of-the-art retrieval engines, such as those available on the World Wide Web, organize the documents retrieved from a query into a ranked list, which provides users with linear access to the documents. This method of providing results works well when the search engine does a good job of interpreting a user's information need. However, when relevant documents are not part of the top ranked documents, a user should still be able to

quickly determine if the information is present in the retrieved set and if it is, find where it is located.

One way to solve this problem is to develop a way of automatically generating a concise summary of the set of retrieved documents. Such a summary would enable the user to quickly determine if relevant information is present anywhere in the document set. The presentation of the summary should also allow the user to quickly access the documents that appear to be most relevant. We envision people using this summary to find the particular data sources that are most likely to be of interest. Thus the automatically generated summary specifically addresses the deficiencies of the ranked list.

In order to summarize a retrieved set, we intend to construct a summary that is composed of single words and phrases, which we refer to as a *word-based summary*. This type of summary differs from other state-of-the-art summaries because these single words and phrases, which we refer to as *words* hereafter, will not be organized into a grammatical framework. Instead the user will interact with lists of words that are related in a hierarchical structure. The user will discern topics of the documents from the list of words included in the summary. A word-based summary is particularly well suited to this kind of task because a user will be able to quickly absorb all of the information contained in the summary. This is in contrast to using a grammatical framework, also known as natural language summarization, which quickly becomes too long, especially when summarizing several hundred documents. Automatically generating natural language summaries will be discussed at greater length in Section 2.1.

Given a hierarchical approach to summarization, the top level of our summary will enumerate the main topics of the document set. These main topics will give a high-level, general description of all the topics in the document set. Underneath the main topics will be lists of subtopics related to a particular main topic. The user's ability to interpret the summary will depend on her understanding of the concept of a $hierarchy$ and her ability to interpret the words included in the summary as topics of the document set.

Although these summaries are particularly well suited to organizing the results of a search, they could also be applied to other tasks. One could use word-based summaries as a means of organizing e-mail or personal files. They would also be useful in a multi-lingual setting. Rather than translating all the foreign language documents returned in a search, the hierarchy could be translated, allowing the user to locate a small number of documents that seem most relevant. This would be especially helpful when translation resources are limited and decisions must be made about which documents should be translated.

1.2 Benefits of a Hierarchy

The hierarchy as a word-based summary is a description of a body of text which both summarizes the text and provides a method of navigating through it. The popularity of hierarchies as a method of organization indicates that this type of summary is relatively easy to understand. For example, the Yahoo hierarchies[73] and MeSH headings[40] have been used for many years. Topics of the hierarchy appear as the nodes in the hierarchy and should describe the documents that are found beneath them.

When searching for documents, this type of organization can give the user an alternative to the organization provided by a ranked list. Specifically, a hierarchy can be built from the full text of documents or even summaries generated by a search engine to describe the documents for the 200, 500, or even 1000 top ranked documents retrieved for a specific query. Unlike a ranked list, where a user is unlikely to find relevant documents not ranked within the top 50, a hierarchy can quickly group the 153^{rd}, 217^{th}, and 568^{th} document under a single topic, for example. If the topic seems relevant, the user will find documents that may have remained undiscovered in a ranked list.

The hierarchical summary has an advantage when compared to clustering algorithms, which are also used to organize a retrieved set (e.g. Scatter/Gather[14]). Clustering algorithms create groups of documents. The characteristics that unify a group are usually difficult to express to a user because they are polythetic. Polythetic clusters are formed

on the basis of several features, but no one feature has to be present in a document for that document to be considered a member of a particular cluster. This characteristic makes clusters very difficult to describe to a user. An example of this appears in the Scatter/Gather work. In one particular instance, three clusters entitled "the Gulf War", "oil sales and stock markets", and "East and West Germany" were created. Hidden within these clusters are documents about Pakistan, Trinidad, South Africa, and Liberia. The existence of these documents in the three clusters is surprising to a user and would most likely engender distrust of the system.

The hierarchical summary tries to avoid unintuitive classification of documents by creating monothetic groups of documents. Monothetic groups require that all the documents share at least one feature. Words that describe the feature(s) are the topic words for the group. There are times when documents may be misclassified, however. Let's consider a hypothetical situation where a document about mosquitoes is classified under the topic "U.S. military". This could happen if the document described research that the U.S. military was conducting on mosquitoes. Although one might not say that the topic of this particular document is the U.S. military, a user would be able to easily determine why the mosquito article was classified in that way. We expect that a user would be able to easily understand this kind of misclassification and would therefore not be surprised. We believe that a user will prefer behavior she can understand, and therefore the hierarchical summary has an advantage over clustering algorithms. Clustering algorithms will be discussed in greater detail in Section 2.2

1.3 Word Associations

The first step in building a word-based summary for a given set of documents is to determine what kinds of words will be most useful as part of the summary. To do this we consider how a human would approach the task of summarizing a topic. The person, Jane, begins with the words that describe the topic. Let's consider the example where the topic

is “fruit”. Jane begins to think about the circumstances where she has encountered the word “fruit” before. Figure 1.1 illustrates this process. First she may think of a number of different types of fruit – bananas, grapes, strawberries, etc – which suggests that “Types” could be the title of a subtopic. Then she thinks about how fruit is cultivated. She realizes that “Cultivation” must be a subtopic of fruit as well. Then she considers that fruit is grown on trees, vines, and bushes and so “Plants” are also part of the topic “fruit”. She then arrives at the topic model for “fruit” illustrated in Figure 1.2.

Figure 1.1. Jane is thinking about how to define the topic “fruit”.

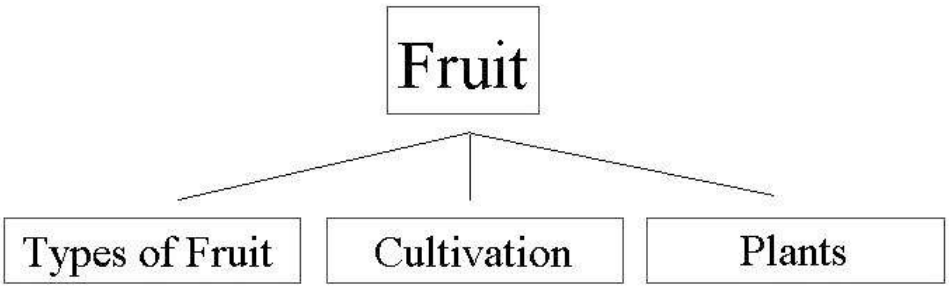


Figure 1.2. The model for “fruit” that Jane defined after considering her knowledge of the concept.

How did Jane produce this description of the topic of fruit? One could describe this process as playing a word association game. Jane tried to think of other words that were highly associated with the word "fruit". In some sense she uses a personal model of "fruit" that tells her which other concepts are likely to occur in the context of "fruit".

Given such a perspective of how people might try to describe a topic, we will use this perspective to develop an algorithm for describing the topics of a document set. Our basic assumption is that we can use topic models as a method of identifying topics and subtopics.

Now we need to consider how one might manufacture a model of a topic from text rather than information stored in one's mind. Let's pretend that Jane has just read about Severe Acute Respiratory Syndrome (SARS). She decides to learn more about SARS, so she reads a few articles. These articles include the snippets in Figure 1.3. Based on her reading, she decides that the subtopics of SARS are pneumonia, Asia, and coronavirus. Jane identified these three concepts as subtopics because they were the important points in the articles and because they have special significance with respect to SARS. One could describe this process as building a personal model of the topic, where the key content-bearing words in the articles are identified as subtopics. The model would look like the one in Figure 1.4.

April 2003 – Health officials from the World Health Organization and the Centers for Disease Control and Prevention are tracking worldwide outbreaks of a contagious and potentially fatal form of pneumonia. Known as Severe Acute Respiratory Syndrome (SARS), the disease first emerged in the Guangdong province of China but has since spread with alarming speed. As of April 8, WHO officials report a total of 2,671 probable cases of SARS, 103 of them fatal, from 19 countries, including the United States. Within the United States, 148 suspected cases of SARS, so far none of them fatal, have been reported....[48]

...Computers at the British Columbia Cancer Agency in Vancouver completed work Saturday on the coronavirus, a new form of which is thought to cause SARS, the agency said....[9]

Figure 1.3. Parts of articles that Jane read about SARS. The boxed areas are the ones that she used to create her model of the topic "SARS".

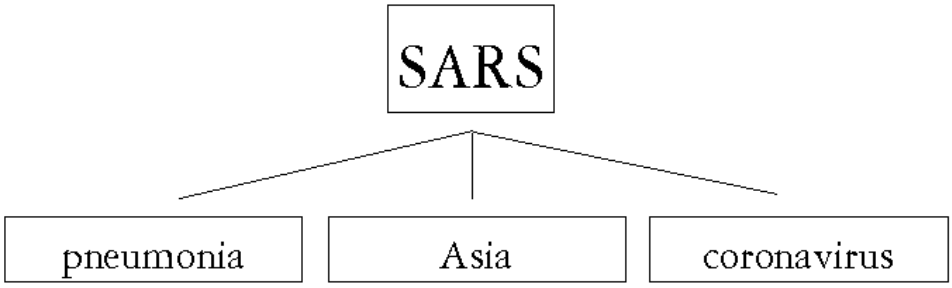


Figure 1.4. The model for "SARS" that Jane defined after reading articles about the topic.

1.4 Language Models and Relative Entropy

In order to automatically create topic models, we will need to develop a method for identifying the key content-bearing words of a document set. This means that we have to determine the features that differentiate content-bearing words from other words in a document. In order to do this, we intend to use a probabilistic approach. Probabilistic approaches have been used with great success in many language-oriented tasks, including speech recognition, part-of-speech tagging, word disambiguation, machine translation from one language to another, and search engines. The success of these probabilistic approaches might be explained by the theory that human cognition is probabilistic. This theory is based on the observation that people deal with uncertainty and incomplete information every day of their lives. In order to successfully interact with the world, they need to be able to cope with this type of information. Because language is an integral part of cognition, language is also most likely probabilistic[44].

The type of probabilistic approach we intend to use is commonly known as language modeling. In general a language model is a probability distribution on a finite feature set. These features vary depending on the type of application. In machine translation and speech recognition, word sequences are used as the feature set, but in part-of-speech tagging, the probabilities of part-of-speech sequences are the features in the model. The probability distribution may be estimated in a variety of ways depending on the information

available, and the way in which the model will be used. A language model describes the language generation process and is generally used as a method of generating text or some other desired output. However, in the context of this work, we use language models simply to express the information in a document set.

In speech recognition, the sequence of words up to a particular point is used to predict the next word in the sequence. In this case, the occurrence of any given word is predicted by the composition of the previous sequence of words. Unfortunately, it is impossible to estimate the values for the most general probability function, $P(w_n|w_1, ..., w_{n-1})$, because there would need to be a huge number of estimations. Since it is impossible to estimate the probability of an arbitrarily long sequence, simplifying assumptions must be made. Language models differ in the simplifying assumptions that are made. The *Markov assumption* is one method of simplification; it assumes that only the prior local context – the last few words – affects the next word. The most common types of models that utilize this assumption are the unigram, the bigram, trigram, and four-gram models. (Actually, the unigram model employs no local context.) As a group, these are referred to as n-gram models for $n = 1, 2, 3, 4$. Table 1.1 illustrates the increase in the number of parameters that must be estimated for each model. The number of parameters is dependent on the size of the feature set. In this case we assume our feature set is the unique vocabulary of the King James Bible, about 12,500 words. Even with this small vocabulary, Table 1.1 shows how quickly the size of the model increases. To have good estimates of a four-gram model, an extremely large corpus would be required. For this reason, only unigram, bigram, and trigram models are used in most current systems.

The unigram model uses the simplest probability function, $P(w)$. This function is memoryless since no history is used to estimate the probabilities. An example of a unigram model with a feature set of size 2, features f_1 and f_2, is $P(x_i = f_1) = P(x_i = f_2) = 0.5$.

In the case of the bigram model, the parameters are the probabilities of the occurrence of a particular feature, given the occurrence of a prior feature. Possible values for such a

model using the same feature set of size 2 follow:

$$\begin{aligned} P(x_i = f_1 | x_{i-1} = f_1) &= 0.8 \\ P(x_i = f_1 | x_{i-1} = f_2) &= 0.2 \\ P(x_i = f_2 | x_{i-1} = f_2) &= 0.8 \\ P(x_i = f_2 | x_{i-1} = f_1) &= 0.2 \end{aligned}$$

These probabilities would be estimated by using a large amount of generic text in order to learn which words tend to follow other words. In this work, we use the unigram model and a form of the bigram model for summarization.

We intend to use language models in the process of locating content-bearing words, which we determined in Section 1.3 to likely be words that people could interpret as topics and subtopics of text. We hypothesize that words associated with a given topic word occur very frequently in conjunction with this topic, but not particularly frequently with most other topics.

We believe language models provide a way to compare text at an abstract level. We intend to compare the topic model of the document set to some more general model of English, which will be referred to as "general English". Using this abstraction, we will be able to identify the differences between the two language models. We believe that the differences we find will occur specifically with topic and subtopic words.

The method that we plan to use for comparing language models is relative entropy, also known as Kullback-Leibler divergence[11]. KL divergence measures how different two

Model	n-grams	Parameters
0th order (unigram model)	$n = 1$	12,500
1st order (bigram model)	$n = 2$	$12,500 \times 12,499 = 156$ million
2nd order (trigram model):	$n = 3$	$12,500^2 \times 12,499 = 2$ trillion
3rd order (four-gram model):	$n = 4$	$12,500^3 \times 12,499 = 2.4 \times 10^{16}$

Table 1.1. Growth in number of parameters for n-gram models

probability distributions are over the same event space. In other words, it measures how well one probability distribution approximates another. For two probability mass functions, $p(x)$ and $q(x)$, KL divergence is expressed mathematically by the function:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

where we define $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = \infty$. We assign the topic model to $p(x)$ and the general English model to $q(x)$. If the calculation is broken down into the portions contributed by each word, the words that contribute the largest positive values are the ones for which general English is the poorest approximation. These same words are the ones we believe are the most useful for word-based summarization.

1.5 Creating Hierarchical Word-Based Summaries

We plan to use KL divergence to find the value of two different attributes of each word used in the vocabulary of the document set. We refer to these two attributes as "topicality" and "predictiveness".

The *topicality* attribute measures how well the word seems to function as a content-bearing word, which may be observed by looking at the distribution characteristics of the word. Specifically, our intuition is that content-bearing words should occur more frequently in the document set than they do in general English. To find these words, we will compare the language model of the document set to that of general English. Words with positive contributions to KL divergence are more frequent in the document set, and therefore will be considered likely content-bearing words.

The *predictiveness* attribute of a word is determined by measuring the number of subtopics that exist for that word. The existence of many subtopics is an essential characteristic of a topic word, because many aspects of the topic must be discussed in order to be classified as a topic within the context of the document set. These aspects are referred to as

subtopics. To identify words with many subtopics, we employ an approximation of relative entropy, which allows us to estimate the dependence between topic and subtopic words. We call this attribute "predictiveness" because the subtopic's occurrence will be dependent on the occurrence of the word we consider the topic word. In some sense, the topic word is predicting the subtopic word. In Chapter 3 we describe the details of how these attributes are calculated.

We also develop an algorithm in Chapter 3 that includes all main topics at the highest level of the hierarchy, even if only a few of the documents in the set are about a particular topic. This means that when the hierarchy is constructed, the topic words chosen by our algorithm will describe all the documents in the document set. Documents will be attached to the hierarchy because of the inclusion of the topic word somewhere in the text of the document. Because all documents attached to a particular node in the hierarchy will contain the topic word, these groups of documents will be monothetic clusters. Secondary and tertiary nodes of the hierarchy will also be linked to monothetic clusters, but these clusters will have more than one feature in common. Specifically, the documents will contain the topic word of the node they are attached to as well as all of the topic words for the ancestors of the node in the hierarchy. As a user explores deeper in the hierarchy, the groups of documents will become smaller because they will be subsets of the groups at the parent nodes. Figure 1.5 is a hierarchy built from the summaries of web documents. These documents were retrieved for the query "Endangered Species Mammals". There are two main topics: "Endangered Species" and "mammals". The "Endangered Species" topic includes 135 documents that contain the word "Endangered Species". In a subset of the 135 documents, the topic word "endangered" also occurs.

The words that are chosen to be part of the hierarchy should enable a user to determine what the documents are about. For example, in Figure 1.5 there is a topic "Endangered Species $\rightarrow$ endangered $\rightarrow$ marine". We expect that a user would infer that the 17 documents under this heading are about aquatic animals that are endangered species.

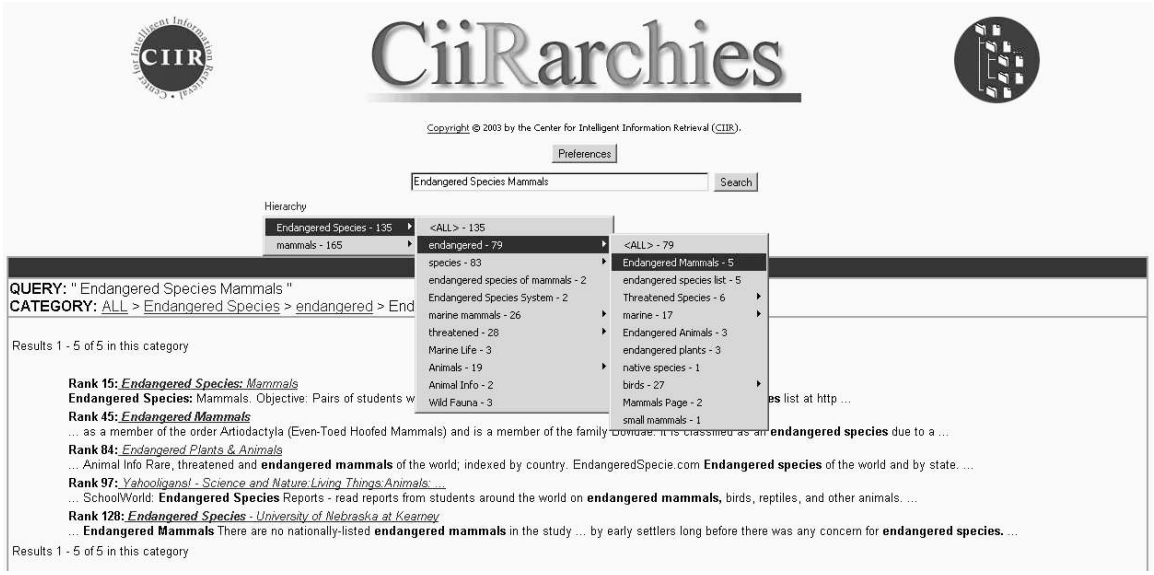


Figure 1.5. This is an example of a hierarchy generated using the probabilistic techniques in this paper. It summarizes 200 web documents retrieved for the query "Endangered Species Mammals".

We present two different types of hierarchies in this paper. The first type of hierarchy is created from the full text of the documents in the document set used to generate the hierarchy. In these hierarchies, the amount of text and the vocabulary considered is very large, which highlights the ability of our algorithm to choose appropriate words to be part of the hierarchy. The second type of hierarchy is created from text fragments, or snippets, of documents. These snippets are very short, usually only about 30 words, and are used as a summary of each document. Because the document set for these hierarchies is composed of snippets, the resultant hierarchies have very different properties. One of the main differences is there are very few words that can be used to describe any particular document in the snippet hierarchy, so the documents end up clumped in very large groups or dispersed in very small groups, with no intermediate size. This document grouping is more evident when the hierarchy is summarizing close to 1000 documents, rather than only 200 as is shown in Figure 1.5.

In addition to creating the hierarchies, we have developed a useful approach for evaluating them. This is a particularly challenging task because human user studies are time consuming and costly to perform. Because of the statistical nature of the approach, which leads to the necessity of testing for the best values of parameters, we required offline evaluations to facilitate incremental improvements to the hierarchy. We developed three such evaluations. One measures the quality of the hierarchy as a summary. The second determines how well the hierarchy provides access to the documents summarized by the hierarchy, and the third ascertains how quickly the hierarchy could give a user access to all the relevant documents. These evaluation are discussed in Chapter 6. Once we determined the best settings of the parameters, we conducted a user study which is described in Chapter 7.

1.6 Research Contributions

This research makes three general contributions:

- We provide a formal framework for creating hierarchical summaries.
- We develop a methodology for using language models as an abstraction of documents in order to create a hierarchical, word-based summary that can be used to navigate a collection of documents.
- We develop non-user-based evaluation measures for hierarchical summarization using large document collections. In order to develop human-usable hierarchical summaries, the current methods of evaluation are expanded to encompass a broader range of attributes. These measures allow us to verify the best parameter settings for the hierarchy without involving a user study.

The formal framework for creating hierarchical summaries relies a method for selection the best topic words by combining two attributes: topicality and predictiveness. Once again, the topicality of a word expresses the extent to which a word will convey information that a user is able to interpret as a topic. The predictiveness of a word refers to the existence

of subtopics. We believe a good topic word is one that has many subtopics, and that the more subtopics a given word has, the more important that topic is in the document set. We refer to this quality as predictiveness because we think of the topic as predicting its subtopics.

In this work, we estimate these two qualities using language models. The language models provide an abstraction of the document set that we can manipulate probabilistically, and the ability to make comparisons between language models to identify interesting features.

We define the best hierarchy as one whose words predict the vocabulary in the document set well, provide good access to the individual documents in the set used to create the hierarchy, and provide a user with the ability to quickly locate the information for which she is searching. We have developed non-user-based evaluations (which we refer to as "offline evaluations") in order to compare hierarchies and determine which are the best word-based summaries. Of course, offline evaluations are not a substitute for user testing; they merely facilitate the development of the summaries.

Determining the most intuitive and useful visualization of the hierarchy is another interesting research topic. In this work, we use the standard popup menu paradigm because it provides the user with a very fast mechanism for browsing the hierarchical summary. Since she can view a level of the hierarchy almost instantaneously, there is little time investment when deciding to view the subtopics of a topic. Unfortunately, the popup menu has limitations. We are able to display only a small amount of information about the topic, specifically the topic word and number of documents that contain the topic word. Other useful information includes

- a visualization of the document set to determine which particular documents are grouped within the topic by their location in the ranking

- the ability to use the interface to find multiple locations of a particular document in the hierarchy. These other locations in the hierarchy might lead to the discovery of other interesting documents

- the ability to find all the occurrences of a particular topic word in the hierarchy.

Although these offer an interesting avenue to explore that is pertinent to the research of hierarchical word-based summaries, this dissertation does not address it.

1.7 Organization of the Dissertation

The remainder of this dissertation is organized in the following manner. In Chapter 2 related work is discussed. We include a discussion of summarization, clustering, and hierarchies. In Chapter 3 we develop a methodology for selecting the words that will appear in the hierarchical summary. Chapter 4 provides examples of hierarchies created using different document sets. In Chapter 5 we discuss the system that was built to construct hierarchical summaries including the web-based user interface. In Chapter 6 we introduce the offline evaluations and use them to compare different hierarchies. We explain the experimental design of the user study and provide the results in Chapter 7. Finally, in Chapter 8 we conclude with a discussion of future work.

CHAPTER 2

RELATED WORK

In this chapter discuss methods of organizing and summarizing documents. Both of organization and summarization attempt to assist a user in finding the information for which she is looking. First, in Section 2.1 we discuss automatic methods of summarization. In Section 2.2 we describe automatic methods of clustering documents. In Section 2.3 we discuss the use of hierarchies within the context of finding information. This includes a description of human generated hierarchies and their use in automatic categorization as well as techniques for automatically generating hierarchies. Finally, in Section 2.4 we conclude with a brief discussion of how the hierarchical, word-based summaries relate to the work presented in this chapter.

2.1 Summarization

Titles, keywords, tables-of-contents, indexes, and document abstracts can all be thought of as forms of summarization. Luhn began working on the problem of automatic summarization in the 50's[41]. Research on the topic has intensified in the past decade with the advent of the World Wide Web. The resulting body of work includes the automatic construction of headline type summaries, cluster-based summaries that can be used as tables-of contents, and document abstract type summaries. These automatic methods of summarization have used three main approaches: linguistic [45, 47], statistical [6, 7], and combinations of the two approaches [19, 31, 60].

Most of the summarization research focuses on single document summaries created by extracting parts of the documents such as sentences or sentence fragments, and then creat-

ing natural language, grammatical summaries. There are also a few examples of generating natural language summaries without using extraction. We will discuss both of these in greater detail in Section 2.1.1. Additionally, there has been exploration into the creation of natural language summaries for multiple documents, most of which has been developed in the last two to three years. This work will be discussed in Section 2.1.2. Finally, there is other research that focuses on word-based summarization. Examples of such work be discussed in Section 2.1.3.

2.1.1 Single Document Natural Language Summarization

In this section we discuss examples of single-document summarization that uses natural language. A majority of this work has focused on extractive methods. Here we highlight research using statistical and linguistic techniques [19, 31], machine learning approaches [34, 10], and purely statistical approaches [57, 77]. We also include an example of a generative summary [6].

Goldstein *et al.*[19] extract sentences based on statistical and linguistic information to form single document summaries. They produce generic summaries which generally summarize, as well as document and query-relevant summaries. This technique uses a variety of statistical information including term weighting, pseudo-relevance feedback, query expansion, thesaurus expansion, and Maximal Marginal Relevance. Linguistic information is used to assign different document genres weights which reflect their individual linguistic features.

While most summarization work focuses on summarizing short news articles or web pages, Kan and McKeown[31] summarize long journalist-style documents containing at least 1500 words in a domain independent fashion. To do this, they first determine the foci of the document, which makes use of term weighting to determine the importance of a portion of text. Each focus is assigned a specific type, and then the system determines which sentences to extract by finding the text that answers questions according to the type

of focus. These sentences are parsed to find grammatical relationships, which should be in the summary.

Kupiec *et al.*[34] use machine learning to create single document summaries. Hand-selected document extracts train a classification function that estimates the probability that a given sentence is included in a summary. Sentences are ranked according to this probability, and a user-specified number of top-ranked sentences are selected to be included in the summary.

Conroy and O'Leary[10] use hidden Markov models to determine which sentences in a document should be extracted for a single document summary. They report that the sentences extracted have more agreement with human extracted sentences than other methods. This technique uses natural language processing to identify terms, which consist of multiple words. The hidden Markov model uses the position of the sentence in the document, the number of terms in the sentence, and the probability of the terms in the document to decide which sentences should be extracted.

Salton *et al.*[57] use a statistical approach to single document summarization. Intra-document links are generated between passages of a document. An algorithm analyzes the linkage pattern to characterize the structure of the text. This is used to perform passage extraction, which form the summaries.

Zha[77] has a unique approach to document summarization which uses a graph theoretic approach. The text documents are modeled using weighted-undirected and weighted-bipartite graphs. A spectral graph clustering algorithm uses the weighted undirected graph to partition sentences of the documents into topical groups. In this graph, vertices represent sentences, and there is an edge between two sentences if there are common terms in the two sentences. The edge is weighted to indicate the similarity between the two sentences. The weight is increased if the two sentences are adjacent within the context of the document. The weighted bipartite graph is used to score individual terms and sentences. In this graph, vertices represent terms and sentences. An edge is present when a sentence contains a spe-

cific term. These edges are weighted based on the frequency of the term in the sentence. Sentences that include many high scoring terms score highly. Both graphs exploit term co-occurrence relationships in textual documents. Zha explores the idea of building a hierarchy of summaries for documents by capturing different levels of granularity. Levels of granularity are recognized by a hierarchical clustering algorithm. When summarizing a non-root node of the hierarchy, terms used in the summarization must remain the same for all nodes, and terms used in summarization at a particular node do not include the top terms from the summarization of its parent nodes. This second point is used to distinguish finer structures of the document.

An example of generating summaries rather than extraction sentences is Berger and Mittal[6]. They create a language model of the document, select terms that should occur in a summary, and then combine the terms using a trigram language model to generate readable summaries. Because Berger and Mittal focus on summarizing web documents rather than news articles, they use a translation approach to create a more concise representation of the web document.

2.1.2 Natural Language Multi-Document Summarization

Obviously, techniques to summarize single documents can be applied to the multi-document summarization task by creating single document summaries and concatenating them together to form a multi-document summary. However, this would create an extremely long multi-document summary. Recently, techniques have been developed to specifically address multi-document summarization.

There are several different approaches to the task of multi-document summarization. Many of them focus on the similarity among a group of documents[7, 46, 61, 38]. In 2001 the first annual Document Understanding Conference (DUC) was held[25]. The conference has greatly influenced the type of multi-document summarization research being conducted. Since DUC creates test sets consisting of about 10 news articles per event,

most multi-document summarization systems try to produce good summaries for such a set. There are a few notable exceptions to this trend. We include one example of summarizing dissimilar information[43]. There are also examples of work that focus on a particular type of multi-document summarization such as providing a biographical summary[60] and summarizing multiple topics[64]. There is also an example of summarizing retrieved sets and user-profile compiled sets[24]. We discuss such forms of multi-document summarization in this section.

Carbonell *et al.*[7] create a multi-document summary by first finding passage similarity using Maximum Minimal Relevance for multiple documents on the same topic. For the summary, they organize the top-ranked sentences in their original order within the documents. This work is extended in Goldstein *et al.*[20] which develops a domain-independent approach to multi-document summarization which relies on fast, statistical processing, and a metric for reducing redundancy and maximizing diversity in selected passages. Goldstein *et al.* makes use of a modular framework to allow easy parameterization for different genres, corpora, and user requirements. They believe that a multi-document summary should contain the key shared relevant information among all the documents only once, plus other information unique to some of the individual documents that are directly relevant to the user's query. This system is unique because it explicitly uses a query. Maximal Marginal Relevance is used to determine the similarity of the query and the document and to insure that new passages included in the summary are different from passages already included. Extracted passages are organized in different ways based upon user requirements. This system was evaluated using 25 sets of 10 news wire articles.

McKeown *et al.*[46] introduces a system called MultiGen which first performs a shallow parsing of the documents and then extract features from the document set. It determines which themes are similar in order to identify phrases that should occur in the summary. Finally, a sentence generator is used, which orders the phrases by the earliest date they

occurred and employs a language generator with complete grammar rules to combine the phrases. MultiGen is designed to handle articles about the same event.

In order to cope with documents on topically related, but different events, DEMS was developed[61]. DEMS produces summaries by extracting the top-ranked sentences until the desired length is met. In order to rank sentences, features are assigned values. The features measure the inherent importance and interest of the segment. The summarizer combines these features using established techniques, such as increasing the weight of sentences near the beginning of the article.

Lin and Hovy[38] introduces NeATS, which uses a six-step process for multi-document summarization. The system combines a number of techniques that had already been applied to single document summarization including sentence position, term frequency, topic signature[1], term clustering, Maximal Marginal Relevance, word filters, and time stamps. This system performed extremely well in DUC 2001 where it produced generic summaries of news documents that focused on a single topic group such as a natural disaster, a single event, multiple instances of a type of event, or a single person.

MEAD[54] is a multi-document summarizer that uses the centroid-based feature to identify sentences that are highly relevant to an entire cluster of related documents and selects these sentences to be part of the summary. Sentences are scored based on a linear combination of their centroid score, position, and overlap with the first sentence. Sentence similarity is computed using cosine similarity and ones that are too similar to a sentence already in the summary are not included.

RIPTIDES is another system for multi-document summarization, introduced in White *et al.*[68] and White and Cardie[67]. This system uses information extraction to generate templates containing the information in the documents. The information extraction is specifically focused on natural disasters. Heuristic-based clustering algorithms organize the

[1]Topic signatures associate a topic concept with a vector of related terms.

extracted concepts into templates specifically designed to support multi-document summarization. RIPTIDES also makes use of linguistic information compiled during information extraction. The summarizer merges the templates into an event-oriented structure. Surface-oriented clustering is used to group sentences from different documents. Sentences are then assigned scores based on a number of features including position in the document, document recency, presence of quotes, average sentence overlap, headline overlap, size of the cluster, size of the semantic groups, specificity of the numeric estimates, and whether these estimates are thought to be current. Iterative random search is used to select the sentences. These sentences are designed to be a high-level textual overview. Each multi-document hypertext summary includes an overview; tables of all comparable numeric estimates, organized to highlight discrepancies; and targeted access to supporting information from the original articles.

Another system that relies on extracted templates in the GISTEXTER[23]. This system highlights differences in articles by making use of templates. In order to generate summaries regardless of the domain, the system generates an ad-hoc template when a trained template is unavailable for the domain of the particular group of documents to be summarized. The ad-hoc template is generated using statistical information with topical relationships mined from the WordNet lexical database. To populate the ad-hoc templates, GISTEXTER acquires extraction rules and derives possible relations between the salient facts that were extracted. Multi-document summaries are generated based on the order in which relevant text snippets appear in the original articles and include sentences or sentence fragments that contain the antecedents of each anaphoric expression from the relevant snippets.

Mani and Bloedorn[43] incorporate some of the differences between a pair of documents in their summaries. To do this they represent each document as a graph, where terms are nodes and edges correspond to semantic relationships between terms. They use spreading activation to find nodes that are semantically related to the topic. Activated graphs of

two documents are matched in order to find a graph corresponding to similarities and differences between the pairs. The system outputs the set of sentences containing the shared terms and the set of sentences covering the unique terms.

The first few examples focus on the similarity among documents. These systems have been used to create concise summaries of up to 25 documents. The problem with including only information that is similar across documents is that a related topic discussed in only a small subset of the document would be excluded. In Mani and Bloedorn[43] where differences are included, only two documents are used in the summary. By using templates, similarity is not as important, but this technique requires more knowledge understanding to extract the correct information. However, even these approaches to multi-document summarization are usually applied to very small document sets.

Utiyama and Hasida[64] is an example of research on summarizing a larger set of documents. They develop a system to summarize multiple topics within an event such as a war or hostage situation. This approach expects that documents have been tagged with Global Document Annotation. Such annotation is produced with a semi-automatic process. It adds linguistic information so that a machine can infer the semantic and pragmatic structures underlying the documents. Humans are needed to verify that the annotation is correct and to resolve any problems. A tree-type graph is created using the document set. The structure of the tree is such that each individual document is a child of the root. Documents are further divided by paragraphs, sentences, and sub-sentence elements. Cross-reference links are added to the tree to reflect the annotation information. A spreading algorithm is used to determine which documents are important and which sentences within those documents should be include in the summary. In Utiyama and Hasida the system is tested using a set of 50 documents about the Peru hostage incident from December 1996 to April 1997.

Schiffman *et al.*[60] summarizes information about people described in the news. Rather than focusing on similarity or dissimilarity between documents, they focus on coherence, accuracy, and non-redundancy using corpus statistics along with linguistic knowl-

edge to select and merge descriptions. The system identifies all sentences in a corpus that mention the person by name. They present examples of summaries that used 24, 503 and 607 sentences. The 24 sentences were drawn from 73 documents while the other sentences were drawn from 1300 documents.

Hardy *et al.*[24] introduces XDoX, a multi-document summarizer for 50-500 documents obtained from a text database, the internet, or a newswire according to a query or user-defined profile. To create a single summary, they created an n-gram based clustering algorithm for passages and build an extracted-summary by selecting a representative passage from each cluster. Generally, they consider a paragraph to be a passage. They weight terms occurring in a larger n-gram more than ones occurring in a smaller n-gram when computing the similarity metric for clustering. Although XDoX is designed to handle fairly large document sets, the only examples in Hardy *et al.*[24] are from DUC where document sets contain about 10 documents. It is unclear how long a summary of 200 documents would be, especially because paragraphs are extracted to form the summary.

Within the work on summarization, there is also the problem of summarizes new news events as they occur in a news stream. Allan *et al.*[2] develop a method of ranking sentences with respect to their usefulness and novelty. Their goal is to generate a summary that summarizes new events. They propose a solution to this problem by defining precision and recall with respect to novel and useful sentences, which they use to rank sentences. When studying summary length, they determine that the best summaries, which include some top ranked sentences, would include 10% of the original sentences.

2.1.3 Word-Based Summarization

Producing summaries of a specific type, such as biographies, makes it easier to include more documents in the summary and still end up with a concise summary of a handful of sentences. However, in order to summarize the similarities and differences of many more documents about a wide variety of topics, a term-based summary is advantageous because

it can remain concise while covering a diversity of topics. The research discussed so far focuses on generating some sort of natural language summary that acts in the same way as an abstract.

Witbrock and Mittal[70] produce headline style summaries which is similar to term-based summaries because headline summaries are extremely short. One of the strengths of this system is its ability to generate words that are not found in the document. They accomplish the task of summarization using a translation model between a document and its summary. The summary structure model is first order Markov, and Viterbi beam search is used to efficiently find a near-optimal summary.

Term-based summaries also exist. Exemplifying a single document term-based summary are the lists of keywords that appear at the beginning of articles. Kea [71] is an example of such a single document summarizer. It uses machine learning to identify features that are characteristic of keywords, and the trained system selects keywords for documents where the author did not give any. An example of multi-document word summaries is a word-based hierarchy which is discussed later in this chapter.

2.2 Clustering

Clustering is a method of organization that has been practiced for many years and applied to many fields. Within the field of information retrieval, two types of clustering have been developed: the clustering of documents and the clustering of terms. Document clusters are created on the basis of common terms among the documents. Term clusters are based on the documents in which the terms co-occur[69]. Clustering has been used within two different environments—browsing and retrieval. Scatter/Gather [14] is an example of using document clustering for browsing where users are presented with clusters in order to locate relevant documents. Document clusters have been used for retrieval in the SMART environment[56], where queries are first compared to document cluster centroids and then with the individual documents of the clusters whose centroid similarity is sufficiently large.

Sparck Jones[62] and Crouch *et al.*[13] are examples of using term clusters for retrieval. Sparck Jones[62] uses term clusters to find similarities among keywords in order to identify terms that could be substituted in a query. In Crouch *et al.*[13] documents are first clustered using the content term vector, which is a vector of the terms in the original query. The cluster that is most similar to the content term vector is used to extend the query with non-content-term sub-vectors from the documents in the selected cluster.

Since this thesis is mainly concerned with the organization of information rather than the retrieval of information, this section surveys research that uses the formation of clusters as a means of organizing documents. As mentioned above, Scatter/Gather is among the most prominent work of this type[14]. It clusters documents into five groups and labels the clusters with the top eight terms. A user is expected to pick one or two clusters which are thought to contain relevant documents. The selected groups are reclustered and again labeled. The user continues this process of drilling down until a satisfactory group of documents is gathered. Yang *et al.*[74] extends Scatter/Gather to the task of topic detection and tracking. They cluster the entire corpus of a few thousand documents and use the top five ranked terms to describe the clusters. They envision that a complete system would allow the user to drill down to the particular level of granularity that met the user's need and then use a single document natural language summarizer to learn about the specific contents of the documents.

Athena[1] is a system developed to organize e-mail into a hierarchical structure using classification and clustering. The system is designed as an interactive tool so the clustering must be accomplished quickly and the classification accurately. The researchers developed a linear time algorithm in the number of documents for clustering and improved the accuracy of the standard method of using Bayesian classifiers by 30 to 60%. The system uses classification for hierarchy reorganization, document routing, and identification of misfiled documents. The clustering component is used for topic discovery. The topic discovery is done interactively with the user, so descriptions of a possible topic are supplied to the

user for acceptance. They compute similarity between documents in the same manner as Scatter/Gather. They tested the ability to discover topics by trying to recover the original classification of email from among 6 different folders. They were able to show that their method of clustering did a better job of recovering the original topics than both k-means and agglomerative clustering. Classification was used to assign documents to the discovered topics. They simulated a user reassigning a small number of incorrectly clustered documents and showed that this significantly improves the results.

Hofmann[28] uses a different clustering algorithm based on an annealed version of the Expectation-Maximization algorithm to produce a hierarchical probabilistic clustering of the documents. As with Scatter/Gather and Yang *et al.*, documents are expected to fit into a single place in the hierarchy, which assumes that documents are about a single topic. Because of the probabilistic model used in clustering, the model can be used to summarize a particular cluster rather than using the most frequent terms for the summary. Hofmann argues that these terms are more discriminating than the most frequent terms, but this description still suffers from the general problem observed in polythetic clustering where topics appear at lower levels of the hierarchy but are not even hinted at in higher levels.

Another example of using clustering for organization is Zamir *et al.*[75], which applies intersection-based clustering algorithms to web snippets as a means of organizing the retrieval results. This clustering algorithm is based on the traditional hierarchical agglomerative clustering algorithm, so it is also a polythetic clustering algorithm. The goal of Zamir *et al.* is to create closely related clusters. They use a global quality function to accomplish this. At each step the union of clusters that increases the global quality the most is formed. The algorithm stops when there is no union that increases the quality. This type of clustering was integrated with a Meta-search engine using Grouper[76]. Clusters are described to the user using a list of shared phrases and sample titles. Because the phrases will not appear in all documents, a percentage figure is used to indicate how frequently a specific

phrase occurs in the documents. At most six phrases and five single words are displayed. It is not mentioned how likely it is that all documents in the cluster contain at least one of the descriptive words, which would give one an indication of how well the terms describe the set of documents.

Lighthouse[37] is another application that uses clustering to organize documents. This work translates the similarity between documents into a graphical representation. This work hypothesis that relevant documents are similar, and therefore once a user finds a relevant document, she should look at more documents within the same region of space. Although Lighthouse uses a polythetic clustering algorithm, it avoids the problem of describing the clusters, by displaying them graphically.

In previous work[36] we used document clustering to create more homogeneous groups of documents and then created hierarchical summaries of the clusters. There are two predominant types of document clustering: non-hierarchic and hierarchic. Non-hierarchic clustering methods partition a document set into groups where similar documents are found in the same cluster and are separated from dissimilar documents. The only way to be assured of achieving an optimal partition in terms of similarity in the non-hierarchic instance is to compare all possible partitions and choose the best. Because this task is infeasible for any realistic document set, heuristics have been developed that produce sub-optimal partitions. Generally this is done by partitioning a set of N objects into a specified number of C clusters while minimizing the total within-cluster sum of squares for each cluster. One example of this technique is the k-means algorithm [29]. Hierarchic clustering methods incrementally divide or combine clusters. This creates small clusters of very similar documents within larger clusters. One method is hierarchic agglomerative clustering, where document clusters begin as singleton clusters and join in N-1 operations to form a single cluster [65].

Clustering has also been used in solutions to a variety of other information-related research. In Section 2.1 there are a number of examples of using sentence or passage based

clustering to determine which points need to be summarized. In Section 2.3 clustering is used as a means for classification and generating a hierarchy.

2.3 Hierarchies

There is a long history of creating hierarchies as a means of organizing information. The most widely used hierarchy today is the Dewey Decimal Classification System[50]. It is used by libraries in more than 135 countries and has been translated into over thirty languages. The classification system was developed by Melvill Dewey in 1873 at Amherst College and was first published in 1876. It was based on an even older hierarchy – Sir Francis Bacon's structure of knowledge[5] which was published in *Of the Proficience and Advancement of Learning* in 1605. One of the reasons for Dewey Decimal's continued importance is that it is continuously updated by the editorial office that is located within the Decimal Classification Division of the Library of Congress. This division categorizes over 110,000 works a year. This enables the editors to detect trends in the literature that must be incorporated into the classification system. The classification is updated every two weeks after the editors prepare proposed schedule revisions and expansions, and forward the proposals to the Decimal Classification Editorial Policy Committee for review and recommended action.

Given the resources that are devoted to maintaining the Dewey Decimal Classification System, it is evident the number of human generated hierarchies will be very small, and that it is unlikely the resources will be made available to categorize all the electronic data that is currently available. When considering information access to this electronic data, there are two ways to hierarchically organizing documents automatically. One is to automatically categorize documents using human generated hierarchies. Such work will be discussed in Section 2.3.1. The alternative is to automatically generate the hierarchies. Examples of this work will be presented in Section 2.3.2.

2.3.1 Classification and Categorization

Many researchers have explored using automatic methods to classify documents into manually constructed hierarchies. In this section we highlight a few of the systems. They use a variety of types of information from the documents including the full text of the documents, text found adjacent to HTML links, and summaries of documents.

Fuhr *et al.*[16] develop a system for indexing abstracts of physical science papers written in English. In order to accomplish the indexing task, they use a dictionary or thesaurus of single words and noun phrases to map the text onto a set of descriptors given by the controlled vocabulary. The controlled vocabulary is what is used to classify the abstracts. Because the dictionary did not previously exist, they explored automatic methods for generating the dictionary. They found that using a large set of documents that were manually indexed with a controlled vocabulary produced a satisfactory dictionary. The dictionary is an estimate of the probability that a particular descriptor s is used for a document when a term t is present in the document. When testing the system, they automatically assigned a total of 12 descriptors to each of the 20,000 documents. These were then checked manually. The automatic indexing of 19% of the documents was considered good; however, the indexers accepted the automatic indexing of 63% of the documents as being useful in spite of some corrections needed. In the remaining 18% of the documents, there were too many problems and so substantial corrections were required.

Koller and Sahami[32] use Bayesian classifiers to categorize documents in a pre-existing hierarchy. Because they expect so many categories in the hierarchy, they break the problem up into smaller steps by creating a classifier for each node in the hierarchy. This allows them to use a small number of features in any one decision because the features used by one classifier do not have to be the same as the features used by another classifier. In this case the features are words. In order to evaluate the classification system, they use the Reuters dataset and construct two hierarchically classified document collections. Although this dataset does not have a predefined hierarchy, they construct a hierarchy by using the

manually assigned labels and then determining which labels tended to subsume other labels. These hierarchies have two or three top-level topics and each top-level topic has two second level topics. The experiments show an improvement in classification error when the hierarchical classifiers are used instead a flat classifier.

Grobelnik and Mladenić[22] classified documents into the Yahoo hierarchy. They assume that the hierarchy captures most topics found on the Web. Like Koller and Sahami, they use a naive Bayesian classifier and the same break down of the decision problem, but their features include sequences of words as well as single words. Their focus is on fast categorization, which they accomplish by reducing the number of categories involved in classification. This is accomplished by using an inverted index where each feature lists the categories it appears in with a probability greater than 0. During classification, only those classifiers that share a required number of highly scored features with the document are considered. When testing the system, experiments show that considering fewer categories improves the classification.

Attardi *et al.*[4] develop a system to categorize a web document based on the text surrounding HTML links to the document. The categorization relies on a base hierarchy to determine where to categorize a particular document. To accomplish this task, they first extract contextual information about the documents by analyzing the structure of the web documents that refer to them. All the contextual information about a document is associated with its URL. Nodes in the hierarchy are described by a title which consists of a single word or phrase. The system uses the title to decide if a particular document should be associated with the category by assigning a confidence value that the document belongs in the category. If the confidence is above a pre-defined threshold, the document will be categorized at the node. The base hierarchy used in the experiments was Arianna, which is an online Italian hierarchy that the authors translated into English. They then used documents that had been categorized by the Yahoo! hierarchy and categorized them in the translated version of Arianna. The authors report that is most cases their system chooses the most

appropriate category. When they attempted to re-categorize the Yahoo! documents based on the Yahoo! hierarchy, they reported that in most cases their system selected the same category. However, in a few instances, they believe their system chose a better location for the document. They also purpose that the system should be able to partition a category that has become to large, perhaps by suggesting subcategories to a user, although there are no experiments that evaluate this proposal.

Chen and Dumais[8] use a pre-existing hierarchy to automatically classify arbitrary search results on-the-fly. They experiment with the LookSmart Web Directory, which has 13 top-level categories, 150 second-level categories, and over 17,000 categories in total. Because they intended to classify web pages based on snippets, the system was trained using summaries of the training documents that were generated using the title, the keyword tag, and either the description tag or the first 40 words. Support Vector Machines are used as the classifiers. They used 13,352 pre-classified web pages to train the system for the 13 top-level categories, and between 1,985 and 10,431 examples of each top-level category to train the second-level categories. When classifying new documents, thresholds are used to determine if a document should be a member of a particular category. Pages are classified at the second level only on demand. They report that the categorization agrees with human-assignment almost 70% of the time. A user study was performed in which participants used either the categories or a ranked list to find a single relevant page from among 100 pages. In this study they showed that participants using a ranked list needed 50% more time to find a page than when using the categories.

A drawback of using predefined hierarchies is that they are not particularly adaptable to varying interests or to changes in the document collection. It would be of great value to develop a way to create hierarchies that would both fully communicate to people the contents of a document set, rather than be predefined.

2.3.2 Automatically Generated Hierarchies

When automatically constructing a hierarchy, hierarchies can be created that are document-oriented or term-oriented, just as in clustering. A document-oriented hierarchy is one in which the documents are divided at each level in the hierarchy. A term-oriented hierarchy uses terms within the documents to form a hierarchical structure. Documents are attached to nodes in some predetermined fashion.

Document clusters such as those created using agglomerative clustering have been used to try to communicate information about document sets. Scatter/Gather[14] is one such example where users search for relevant documents by selecting multiple high level clusters. At each level in the hierarchy, fewer documents remain to be re-clustered. The problem with this approach is that the polythetic clusters used have, by definition, many terms in common but no specific term is required for membership. Because of this, it is difficult to communicate the contents of a cluster.

Term-oriented hierarchies are much more common and make up a large portion of the manually created hierarchies such as MeSH (Medical Subject Headings) and Yahoo!, and they seem better suited to the task of communicating the contents of a document set. In the following sections, we discuss three types of word-based hierarchies – ontologies in Section 2.3.2.1, subsumption hierarchies in Section 2.3.2.2, and lexical hierarchies 2.3.2.3.

There is at least one other automatic method for constructing a hierarchy. It is developed in Glover *et al.*[18]. This work identifies terms associated with a cluster. Specifically, the research is interested in self terms which describe a cluster, parent terms which describe more general concepts, and child terms which describe specializations of the cluster. Terms are identified by using the frequency of a term in a set of documents compared to the frequency in the entire collection. Self terms are those terms that are very frequent in the document set but not in the collection. A term that is common in both the set and the collection is a parent term, and a term that is somewhat common in the set and rare in the collection is a child term. This approach is tested using the Open Directory Project.

Document clusters consist of 500 documents. Forty-one sets are used in experimentation. First, the hypothesis is tested by computing all the frequencies of the original labels in the hierarchy. This is used to assign cutoffs to distinguish the three types of terms. In about 80% of the cases, the top self terms are the actual labels in the hierarchy. In all 41 cases, the parent term assigned by the hierarchy is ranked in the top five by the algorithm used to assign terms. Children terms are harder to judge because acceptable children are more numerous, so no evaluation of such terms appears in Glover *et al.*.

A more detailed description of subsumption and lexical hierarchies follows because in Chapter 6 we use them in a comparison with the methodology developed in this paper.

2.3.2.1 Automatic Ontology Construction

The main difference between ontologies and other types of word-based hierarchies is that ontologies are generally constructed in order to be used as a knowledge base that is preserved and used for other tasks. Other word-based hierarchies can be tailored to a specific document set and do not necessarily reflect any static relationship between the topics that are described.

One example of the automatic construction of an ontology is Kosovac *et al.*[33], which use automatic methods for collecting terms relevant to a specific field and updating an existing thesaurus. To accomplish this, they gather electronic documents that can be used as the source for extracting the terms. These documents are collected using five search engines to gather 75 documents. During experimentation, the authors tested the quality of the words and found that there is no significant difference between words extracted from the retrieved document and words extracted from a set of high quality documents. The extracted words are also compared to the existing thesaurus and over half of the words that were extracted multiple times appeared in the thesaurus. The authors assert that the majority of mismatches reflect the out-datedness of the thesaurus rather than irrelevance of the terms. Inserting the terms into the thesaurus is expected to be done manually although

the authors believe some information about the terms could be provided automatically and would assist with this step, such as inter-term relationships based on co-occurrence and syntax information.

Maedche *et al.*[42] also use web documents to automate the ontology engineering process. This work uses a lexical approach to identify the concepts and then gathers statistical data about the concepts from the web documents to increase the accuracy of automatically augmenting the ontology with new concepts. The statistical data include co-occurrences of words that occurred 3 words before and after the word. They developed a tree ascending algorithm to locate the best location for a new concept. In order to test the approach, the ontology GETESS was used. Each concept in the ontology was described by a single word or phrase. When adding new concepts to the ontology, each node is also associated with its child concepts up to three levels deep. In order to test the performance of the approach, they used all but one of the concepts in training and than tested to see how close to the original location the final concept was placed in the ontology. Unfortunately, they found that standard clustering techniques were better able to find the correct location of the new concept than the tree ascending algorithm which made use of the structure of the ontology, which means that this algorithm does choose the same relationships as a human would.

2.3.2.2 Subsumption Hierarchies

One way to create a hierarchy of terms is using the notion of subsumption. Given a set of documents, some terms will frequently occur among the documents, while others will occur in only a few documents. Some of the frequently occurring terms will be ones that provide a lot of information about the topics within the documents. In fact, there are some terms that broadly define the topics, and others that co-occur with a general term and explain aspects of a topic. Subsumption attempts to harness the power of these words.

A subsumption hierarchy as described in [59] has the following characteristics:

- a means of associating terms so that it reflects the topics covered within the documents,
- within the association, a parent term is more general than its child,
- a term subsumes all of its descendants so that transitivity holds,
- a child may have more than one parent.

These characteristics are achieved by defining subsumption as the conditional probability:

$$P(x|y) \geq 0.8^2 \text{ and } P(y|x) < P(x|y). \tag{2.1}$$

Thus *x* subsumes *y* if the documents in which *y* occurs are a subset or nearly a subset of the documents in which *x* occurs. The second rule ensures that if both terms occur together more than 80% of the time, the more frequently occurring term will be chosen as the parent. If the terms co-occur exactly the same number of times, the two terms are combined into a single unit. This definition is designed to identify high quality relationships between words.

Once one has defined a notion of subsumption, the candidate terms must be selected. Sanderson and Croft[59] intended subsumption hierarchies to be used after retrieving documents using a query. In this case, terms can be selected from both the document and the query. Query terms and terms generated from automatic expansion are used to focus the hierarchy, since they should describe the interest of the user.

Document terms are selected by comparing a term's frequency of occurrence in the set of retrieved documents with its occurrence in the collection as a whole. Terms that are 'unusually frequent' in the retrieved set compared to their frequency in the collection are selected. This list of terms is sorted based on their scores, and the top N terms are designated for use in the subsumption hierarchy.

[2]The threshold 0.8 was determined empirically.

Subsumption relationships are found using O(n^2) comparisons where n is the size of the vocabulary. Finally, extraneous relationships are removed. Because of the transitive nature of subsumption, there will be some terms where *a* subsumes *b*, *a* subsumes *c*, and *b* subsumes *c*. The relation *a* subsumes *c* can be eliminated because it is redundant. Relations are also eliminated if the terms co-occur together in fewer than two documents.

2.3.2.3 Lexical Hierarchies

Another way to create a hierarchy is by using the hierarchical structure of phrases that appear frequently. Creating such a hierarchy has been explored by many people including Nevill-Manning *et al.* [49] and Anick and Tipirneni[3]. Both of these cases rely on frequently occurring words within phrases or noun compounds of a document set to expose the topics of that document set. Anick and Tipirneni introduces the *lexical dispersion hypothesis* which states that "a word's *lexical dispersion* – the number of *different* compounds that a word appears in within a given document set – can be used as a diagnostic for automatically identifying key concepts of that document set."

A $\rightarrow$ B (pcfgs)
B $\rightarrow$ probabilistic C
C $\rightarrow$ D grammars
D $\rightarrow$ context free

Figure 2.1. Production rules for "probabilistic context free grammars (pcfgs)".

There are many ways of selecting the phrases that will be used for candidates in the lexical hierarchy. Nevill-Manning *et al.* breaks all sequences into hierarchies in such a way that each branch refers to a rule in a context free grammar. The highest level of the sequence generates the entire sequence, which consists of unique sequences in the sentence and other rules that must occur at least twice in the collection. For example, if the phrase "probabilistic context free grammars (pcfgs)" appeared as a sequence, the rules to generate this sequence appear in Figure 2.1. In this case, A is the highest level

sequence and "(pcfgs)" is the unique portion of the rule. For the corpus that was tested in Nevill-Manning *et al.*, sequences averaged 9.6 non-terminals.

Another way to locate sequences is to match the pattern {*?adjective noun+*} of two to four terms in length, as was done in Anick and Tipirneni.

Once the phrases are chosen, they are divided into groups based on the terms that appear in the phrases. The lexical dispersion of each term can then be calculated. Anick and Tipirneni studied the effects of ranking the candidate terms based on lexical dispersion and found that in order to study the dispersion of a term throughout the document collection, it is also necessary to examine the number of documents that involve phrases using a particular term. Otherwise, a long document that uses the term a large number of times could make that term seem like a much better candidate than it actually is. As a rule, Anick and Tipirneni ranked terms based on the number of documents that contributed at least one phrase if the dispersion level exceeded five phrases. The remainder were ranked by dispersion.

2.4 Conclusion

The work described in this thesis is a word-based summary, which is fairly unique among the summarization work. As a clustering algorithm, this work takes a different approach then most other clustering research because it uses monothetic clusters, rather than polythetic clusters. Among the work on automatically generating hierarchies, this work is unique because it uses a probabilistic approach in which language models determine which words should be part of the hierarchy.

CHAPTER 3

PROBABILISTIC FRAMEWORK FOR SUMMARIZATION

In this chapter we develop a probabilistic framework of a word-based summary. To do this we pursue a methodology for using language models and relative entropy in order to identify words that people will be able to associate with topics and subtopics. In Section 3.1 we present a mathematical model that will use topicality and predictiveness to find the best set of words for a word-based summary. In Section 3.2 we discuss a method for estimating topicality. We then discuss estimating predictiveness in Sections 3.3 to 3.5. In Section 3.6 we propose an algorithm that implements our model. In Section 3.7 we demonstrate how the algorithm works, and conclude in Section 3.8.

3.1 The Model for Word-Based Summarization

By studying subsumption and lexical hierarchies and development of the probabilistic hierarchy, we have identified two important qualities for the words which are part of a word-based summary. The first is topicality, which indicates whether a word is a content-bearing word in the document set. We believe that a summary word should be content-bearing because a user is more likely to be able to interpret it as a topic of the document set than a non-content-bearing word. The second quality is predictiveness, which measures the number of subtopics of a given word. We hypothesize that the more subtopics a word has, the more likely the word is to describe a topic in the document set. This leads to our model for finding the best set of topic words, which we refer to as our Term Selection Formula:

$$T* = \arg\max_{T \in W} \sum_{i=1}^{|T|} \mathcal{P}(\mathit{Topical}(w_i), \mathit{Predictive}(w_i) | w_1 ... w_{i-1}) \tag{3.1}$$

where T is the set of chosen topics and W is all subsets of the vocabulary. Expression 3.1 finds the set of topics that maximizes the joint probability of topicality and predictiveness, where the probability is dependent on previously chosen words.

$Topical$ is a binary random variable and should be true when a word is a content-bearing. A topical word is not necessarily the most frequent word in the document set, since it may only be mentioned once in a particular document set; however, the information conveyed to the reader by content-bearing words is crucial to understanding the document.

The second part of the joint probability in Expression 3.1 is predictiveness. $Predictive$ is also a binary random variable, and such a word can be thought of as a precondition for many other words. This quality takes into account the fact that subtopics occur in the presence of the main topic. Previous work on topic hierarchies[59] has shown that this is an important aspect of topic words, because these are the words that are frequently used to discuss the topic at different levels of generality. Specifically, predictive words are those which occur with a distinct set of vocabulary and without which it would be highly unlikely that other words would occur. Included in this set of words are most stopwords[1], which due to their frequent occurrence in a document make most words dependent on them. General words related to a topic are also in this set. For example, in a retrieved set about Endangered Mammals, "endangered species" is likely to be a predictive word because many other words occur only when it is present.

By combining these two random variables, one arrives at a set of words that will maximize a user's understanding of the information contained in the documents. In order to compute the joint probability of the two random variables, we assume the two probabilities are independent, and so

$$\mathcal{P}(\textit{Topical}(w_i), \textit{Predictive}(w_i) | w1 \ldots w_{i-1}) =$$
$$\mathcal{P}(\textit{Topical}(w_i) | w1 \ldots w_{i-1}) \mathcal{P}(\textit{Predictive}(w_i) | w1 \ldots w_{i-1}).$$

[1]The most frequent words used in the English language including "the", "is", "said", etc.

This independence assumption is a common assumption in information retrieval with regard to words. One example of assuming words are independent is Relevance Models[55]. Although the independence assumption for words is clearly untrue, it has been shown that the dependence is a second order effect[65]. Although the assumption that topicality and predictiveness are independent is different than the assumption that words are independent, we do have some evidence that this assumption may be valid. Specifically, knowing whether a word is topical does not necessarily lead to the conclusion that the word is predictive, and conversely knowing that a word is predictive does not necessarily lead to the conclusion that a word is topical. If one considers the set of high frequency words such as stopwords, they are always predictive but usually not topical. Also, if one considers the set of low frequency words, they are generally topical but rarely predictive. The words in the mid range can be topical and predictive depending on their usage.

Once the independence assumption is made, we need to develop a means for estimating topicality and predictiveness. In the following sections we will describe our estimation techniques. We begin with our estimation of topicality in Section 3.2. In Section 3.3 we develop the methodology for incorporating the conditional part of Equation 3.1 for the predictive estimate. We then present a relative entropy approach in Section 3.4, but find insurmountable problems with this approach. We then develop an approximation to relative entropy in Section 3.5, which is the method we use in this paper for estimating predictiveness.

3.2 Estimating Topicality

When creating the hierarchies, it is necessary to estimate the probability that a particular word is in fact a topic word. Cronen-Townsend and Croft have shown in their work with the Clarity measure that words with high contributions to the Kullback-Leibler divergence score are likely to be about the topic of the document, while words that are not ranked

highly are less likely to be about the topic[12]. This is why we choose to use the Clarity measure by estimating topicality using KL divergence.

In this context, KL divergence is a measure of relative entropy between the language model of the text used to create the hierarchy and the language model of general English. We are interested in each word's contribution to KL divergence, so we calculate:

$$\text{KL contribution}(w) = \mathcal{P}_{hier}(w) \log_2 \frac{\mathcal{P}_{hier}(w)}{\mathcal{P}_{GE}(w)},$$

where $\mathcal{P}_{hier}(w)$ is the probability of a word in the document set and $\mathcal{P}_{GE}(w)$ is the probability of a word in general English. A particular word w has a contribution of zero when $\mathcal{P}_{hier}(w) = \mathcal{P}_{GE}(w)$. It has a positive contribution when $\mathcal{P}_{hier}(w) > \mathcal{P}_{GE}(w)$, and a negative contribution when $\mathcal{P}_{hier}(w) < \mathcal{P}_{GE}(w)$. The most topical words will be the words where $\mathcal{P}_{hier}(w) \gg \mathcal{P}_{GE}(w)$ because they are the words that occur much more frequently in the document set than would be expected in general English.

Now, $\mathcal{P}_{hier}(w)$ and $\mathcal{P}_{GE}(w)$ must be estimated. The most straightforward approach is to estimate the unigram language model for the hierarchy where

$$\mathcal{P}_{hier}(w) = \frac{\#\text{occurrences}(w)}{\#\text{total words}}.$$

The language model of general English can be estimated in a similar way by using the frequency of words in a suitably large collection.

Theoretically speaking, there exists a single model of general English that describes all of the English language. This means that we should be able to use one model of general English for all possible document sets. During our experimentation, we found that sometimes this assumption did not hold. Initially, we estimated our model from a very large collection of newspaper and government documents. When we began building hierarchies of web documents, standard web terminology would appear in the hierarchy. The phrase "Click on me" is a good example. Although it is a common phrase for web documents, it

is not a common phrase for newspaper documents. If one observed this phrase in a news document, it would most likely be highly related to the topic of the document. On the web this is not the case. Such a phrase is not a good candidate for a topic word, and so our model of general English should reflect that.

There are two ways of addressing this problem. One method is to bias $\mathcal{P}_{hier}(w)$ towards the query. This method can only be used when the hierarchy is being constructed for a group of documents retrieved for a query. Given a query Q, we estimate the probability of w using the formulation in Cronen-Townsend and Croft[12]:

$$\mathcal{P}_{hier_Q}(w) = \mathcal{P}(w|Q) = \sum_{D \in hier} \mathcal{P}(w|D)\mathcal{P}(D|Q),$$

where $hier$ refers to the set of documents used to create the hierarchy and

$$\mathcal{P}(D|Q) = \prod_{q \in Q} \mathcal{P}(q|D).$$

This method demotes words such as "Click on me" because they are irrelevant to the query. A query bias provides a better estimate of topicality; however, it does not provide a universal approach – hierarchies can be constructed for groups of documents that are not related by a common query.

The second method assumes the algorithm receives as input a document's sub-collection or document type. For a set of documents S, $S = \{S_1, S_2, ...S_n\}$, there are n different sub-collections from which the documents originated. Given a particular S_i, there exists a language model SC_i that models the type of language used by a particular sub-collection. This means that the KL divergence contribution of a word can be calculated for each sub-collection separately, and then combined to get a single KL divergence contribution in the following manner:

$$\text{KL contribution}(w) = \sum_{S_i \in S} \mathcal{P}(S_i)\mathcal{P}_{S_i}(w) \log_2 \frac{\mathcal{P}_{S_i}(w)}{\mathcal{P}_{SC_i}(w)},$$

where

$$\begin{aligned}
\mathcal{P}(S_i) &= \frac{|S_i|}{|S|}, \\
\mathcal{P}_{Si}(w) &= \frac{\#\text{occurrences}_{S_i}(w)}{\#\text{total words in } S_i}, \text{ and} \\
\mathcal{P}_{SCi}(w) &= \frac{\#\text{occurrences}_{SC_i}(w)}{\#\text{total words in } SC_i}.
\end{aligned}$$

This method also gives a better estimation of a word's topicality, but does not bias the hierarchy with respect to the query.

Both methods were used to generating hierarchical summaries. In Section 6.2.5 we show that using sub-collections significantly improves the hierarchies in some circumstances, and never harms the performance of the hierarchy.

Regardless of the method for estimating the KL contribution, we estimate the probability that a word is topical using the following equation:

$$\mathcal{P}(\text{Topical}(w_i)|w_1 \ldots w_{i-1}) = \frac{\text{KL contribution}(w) - \textit{minKL}}{\textit{maxKL} - \textit{minKL}} \tag{3.2}$$

where *minKL* is the smallest KL divergence value observed and *maxKL* is the largest KL divergence value observed. This method of estimation does not make use of previously chosen words, which means that the quality of topicality is independent of the other topic words in this estimation.

3.3 Using Breadth First Search to Discover Topic Words

Now that we have an estimate for topicality, we need to estimate predictiveness. The predictiveness of a word should increase as the number of subtopics associated with it increases. We assume the words concerning a particular topic will be found near each other. Our measure should choose the most general word to represent the topic. Each topic

word should concern a different general topic, so the estimation of predictiveness should take this into account.

In this section we explore how to ensure that our estimation technique can account for the fact that all general topics need to be described by the word-based summary. If the document set contains a single dominant topic and several minor topics, it is likely that most of the top n topic words, based on some method of estimating predictiveness, would relate only to this dominant topic, which means that there is no guarantee that these top n words will represent all the topics in the document set. In order to create a complete summary of the document set, all topics in the set must be covered, not just the dominant one(s).

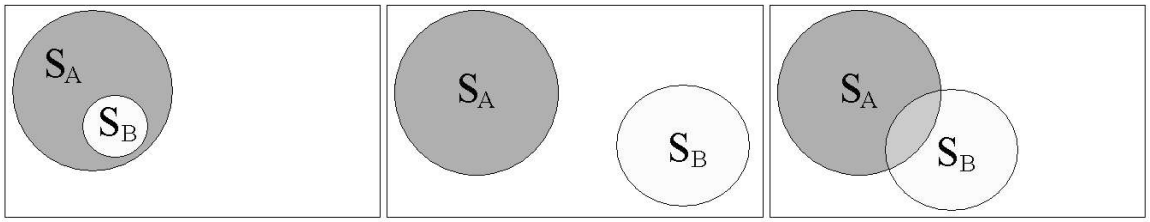
One way to describe all general topics is to divide the document set into topic groups using polythetic clustering, thereby creating multiple groups of similar documents. The assumption would be that similar documents are about the same topic. The problem with such an approach arises from our preference to describe each topic with a single word. With this constraint, we would have to choose the one best word to represent each group. Yet, it is unlikely that one word could actually represent the topic of the group as we discussed in Section 1.2 because it was created using polythetic clustering.

Instead of forming the groups and then labeling them, we will use the topic words as labels, and then assign the documents that fit each topic. This will create monothetic groups where all documents in the group include the topic word in their text. We will choose topic words in a step-by-step procedure to accomplish this monothetic clustering. First, we will identify the best topic word in the document set, which should represent one of the dominant topics. We then will determine exactly what the topic represents, so that the next topic chosen represents a different main topic.

There are two classic approaches that we can use for determining what the topic represents. One method is to identify a topic and then find all of its descendants (the subtopics, sub-subtopics, etc.) using a depth-first approach, so that the topic is fully specified. Once

this is done the topic should be clearly defined, and it will be easy to select the next topic from outside this description by eliminating those words used in the description of the first topic. The other approach is to use a breadth-first approach. In this case the coverage of a topic must be estimated so that subtopics are not mistaken as topics.

The depth-first approach has a number of flaws. The most important problem is that a particular word may be both a topic and a subtopic. Let's consider two topics A and B. Each topic has a set of subtopics: S_A are the subtopics of A and S_B are the subtopics of B. Figure 3.1 illustrates how the sets may be related. Either one set of subtopics is a subset of the other, as in Figure 3.1a, the two sets are completely distinct as in Figure 3.1b, or the two subsets overlap as in Figure 3.1c. Let's assume that we found B to be a subtopic of A. With a depth-first approach, B would be considered a subtopic of A before it is ascertained that it should not be a topic. If S_B is completely subsumed by the set S_A as is shown in Figure 3.1a, it is clear that B should be a subtopic of A. However, if the subtopics of A and B are only a subset of S_B as is shown in Figure 3.1c, then there must be a decision as to where the best place for B is in the context of the whole document set, i.e. as a subtopic of A or as a topic in its own right. Our solution must take this into account.



a: overlapping sets b: disjoint sets c: partially overlapping sets

Figure 3.1. Illustrates how different sets of subtopics may be related.

Another disadvantage of the depth-first approach is that the hierarchy would tend to be deep and most likely would not have many subtopics under any particular topic. Because we believe people often find it difficult to keep a large number of relationships in mind simultaneously, such a hierarchy would be hard to understand. A shallow hierarchy with

many branches is much more desirable because the number of relations between words can be reduced.

The greatest strength of the depth-first approach is that the topic is completely defined before selecting the next topic. However, this assumes that a particular subtopic can only be a subtopic for a single topic. This may not always be the case.

The alternative is to take a breadth-first approach. Rather than having the topic clearly defined, the coverage of the topic will be estimated. When developing a method of estimating predictiveness, we will have to ensure that there is some method for defining what a topic is related to. In this way we will be able to iteratively re-estimate the probability of predictiveness with respect to the words that have previously been identified as topic words.

3.4 Using Relative Entropy to Identify Predictive Words

We would also like to use a relative entropy approach to estimate predictiveness, which means that we need to find a way to identify subtopics using relative entropy. One way to do this would be to develop an algorithm that will identify topic words by considering each word in the document set in turn to determine the extent to which subtopics exist for that word. First, we will develop the language models needed for relative entropy in Section 3.4.1. Then we will determine a method for using the breadth-first approach for relative entropy in Section 3.4.2. In light of the requirements of the breadth-first approach, we will explain why using relative entropy is not an acceptable approach to estimating topicality in Section 3.4.3.

3.4.1 Creating a Topic Model

The first step when using relative entropy to identify predictive words involves transforming the document set into language models. Since our goal is to use the models to find topic words with many subtopics, we must develop a methodology for creating these mod-

els. We believe a subtopic must be a key point that is discussed within the context of the topic. In order to identify the context of a topic, we hypothesize that the words occurring near each other in the text have a strong dependence on one another. This means that the distance between words, which is calculated by counting the words that occur between two words, will be the determining factor for dependence. If we consider word a, which occurs at position i, and word b, which occurs at position j, the likelihood of dependence between them decreases as the distance ($|i - j|$) increases. Given this hypothesis, we can determine a maximum distance where associations between words are probable. This hypothesis is loosely based on Wittgenstein's *Use Theory of Meaning*, which states that the meaning of a word is defined by the circumstances of its use[44]. In light of this theory, some of the surrounding words should be generalities of the topic while others are subtopics of the topic.

Given this hypothesis concerning word associations, we will build *topic* language models using the sequences of text that include the topic word. This means that if we were to build a model of the topic word "SARS", we would first locate the instances where "SARS" occurs in our document set. From this we would build a unigram model of the text, by estimating the probability of each word in our vocabulary given the text segments where "SARS" is present. One possible method for defining a segment is to include the 200 words that occur before and after an occurrence of "SARS". All such segments containing "SARS" would be used for estimating the topic model. Our algorithm will construct such a topic model for each possible topic in the document set.

In order to construct topic models, we must identify which words should be the topics of the models. Of course, one could assume that all words in a document set are possible topics. However, by limiting the words to what we refer to as the candidate topics, we are able to bias the algorithm in certain ways and improve the ability to identify topic words. For instance, we can limit the words to the content-bearing words by using relative entropy in the manner discussed in Section 1.4. We believe the only words that can be

topics are those that are content-bearing words in the document set. To accomplish the identification of the candidate topics, the algorithm will use a unigram language model of the entire document set and compare it to a language model of general English using Kullback-Leibler divergence. The particular words which contribute a large positive value to KL divergence are the words that are most likely to be the content-bearing words. The reason that the words have a positive KL divergence contribution is that their probability is greater in the document set than in general English. Biasing the candidate topics to words that make a positive contribution KL divergence is tested in Section 6.2.1. We hypothesize that the greater a word's contribution to KL divergence, the more likely a word is to be a topic word.

Now that we have a set of possible topic words and the ability to construct topic models for them, we intend to use the models to determine which words are the best topic words. Our expectation is that sections of text related to each other due to the existence of a topic word will be less similar to a model of general English than sections of text that are related because of the existence of a non-topic word. This expectation arises from the differences in the kinds of language we expect to co-occur with topic words and non-topic words. Let's consider two different words: "SARS" and "conclude". If we consider all the segments of text where "SARS" occurs, co-occurring words should concern health, medical research, and places where the disease has had an impact. In particular, these words will have a probability in the topic model that is much greater than in general English, and are therefore subtopic words. It then follows that "SARS" is a topic word. On the other hand, let's consider the word "conclude". This word is likely to come up in segments relating to all different kinds of topics. Because there is no standard type of vocabulary that we expect to see with this word, the language model of its segments should be very similar to the language model of general English. We can use this to identify "conclude" as a non-topic word. This means that a non-topic word's language model resembles general English be-

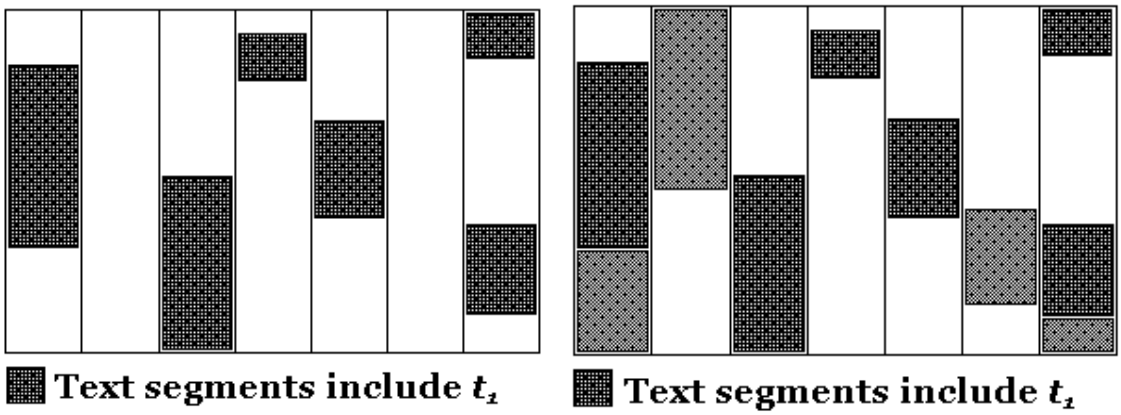
cause it lacks subtopics. In a sense, the reason that general English is a bad approximation for models of true topic words is because topic words have subtopics.

3.4.2 Using a Breadth-First Approach

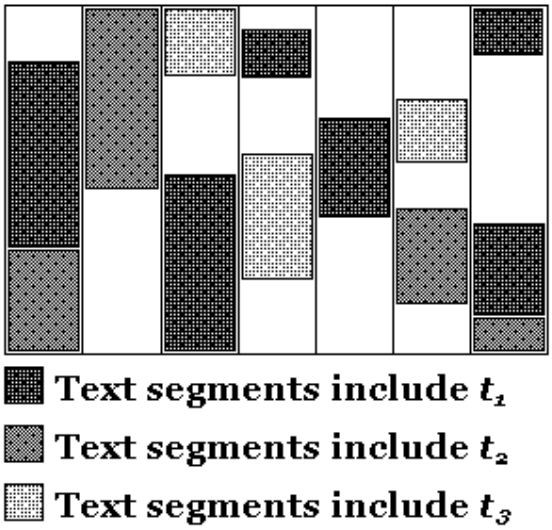
In order to use a breadth first approach, we will develop an algorithm that selects the best topic, t_1, according the comparison of the language models. Now we want to change the topic models of remaining candidate topic words so that we can select the next best topic word that it is unrelated to t_1. In Section 3.4.1 we described how segments of text are associated with each candidate topic when constructing the language model. We hypothesize that if we remove the text associated with t_1 from the document set, as is illustrated in Figure 3.2a, and rebuild the topic models of the remaining topic candidates, the new models will reflect the text without the dominant topic. With the text of t_1 removed, we can select the next best topic by again comparing the topic models to the general English model and determining which is most different. Because of the way the models are created, the topics that are chosen are only the topics for the particular segments of text that were used to build the model. After those segments have been eliminated, we can determine the next best topic from the remaining text. The topic models will be estimated using increasingly smaller amounts of text as is shown in Figure 3.2.

From this process, we determine the top-level topics of the document set. Because we have a way to estimate the coverage of a topic, we will use the breath-first approach for finding topic words.

Thus far we have only determined the top-level topics. In order to complete the summarization task, we must associate subtopics with the topics in such a way that the subtopics explain the information covered by the topic. We assume a limited space which may not allow us to simply list all the subtopics. This means that we must determine which subtopics are the best at explaining each topic. We will reuse the approach that was used to discover topics by computing language models within the context of two words, rather than just



a: Segments t_1 b: Segments t_1 and t_2



c: Segments t_1, t_2, and t_3

Figure 3.2. Illustrates how the language model will change as topics are chosen. Initially all the text is used to estimate the language model. Once t_1 is chosen as the first topic, the topic models for all candidate topics are re-estimated without using the text associated with t_1 as indicated in Part a. From these new language models, t_2 is chosen as the best topic word, and the text associated with it as shown in Part b is removed from all the other topic models. Then t_3 is selected as a topic word. This process will continue until the maximum number of topics is reached or all the text is associated with a topic.

one. This is done by identifying the segments where both words are present. By asking the question "What are the subtopics for the two words?", we can find the best subtopics of the topic from among all the subtopics. Using this definition, we have a recursive algorithm for defining the hierarchy.

3.4.3 Problems with Relative Entropy for Estimating Predictiveness

As we argued above, relative entropy is well suited for our task because it enables us to compare language models. Let's consider an example to explain exactly what relative entropy will mean under these circumstances. Our plan is to compare the model, M_t, for the topic word t to the model of the entire document set. M_t will be estimated using the segments of the document set where the word t occurs.

As a simplifying assumption, the following discussion assumes that no smoothing is used when generating the language models. Smoothing language models is a standard practice because it makes the model more accurate by accounting for the possibility that a word that did not occur might have occurred in some text about the topic. Smoothing is generally used because language models are generative models, meaning that they should predict the text they are modeling, and to cope with poor estimates due to small sample size. For a topic model to be generative, the model should be able to generate segments of text that are about the topic even if they were not used to estimate the model. This means the model should be able to generate all the words in another piece of text, which may include words that were not present in the segments used to estimate the topic model. In order to do this, there needs to be some probability for these previously unseen words occurring. The other problem with smoothing in this particular instance is that it is unclear what model should be used to smooth the topic model since we are very interested in the relationships under the particular circumstances that occur in the document set.

We now compute $D(p||q)$, the KL divergence, where p is the probability distribution of the model M_t and q is the probability distribution of the document set. In this case,

the document set is representing general English. Because the words of p with non-zero probabilities are a subset of the words in q, we will never encounter the situation where

$$p \log \frac{p}{0} = \infty$$

and thus the total value of KL divergence will never be infinity.

Now let's consider the contribution of an individual word to the KL divergence value. To estimate the probability of an individual word, $p(w)$, we use the standard approach, which is to divide the number of occurrences of the word by the total number of words that occur in the model. Let's say there are y total words in the segments used to estimate M_t and z total words in the document set. Now let's consider the word a. Assume there are x_a total occurrences of a in the document set, and that every occurrence of a falls in a segment associated with M_t. This would mean that probability of a in model M_t is

$$p(a) = \frac{x_a}{y},$$

and the probability of a in the model of the document set is

$$q(a) = \frac{x_a}{z}.$$

Now, recall from Section 1.4 that the function for calculating KL divergence is:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}.$$

If we consider just the log portion of the KL divergence calculation, then

$$\log \frac{p(a)}{q(a)} = \log \frac{z}{y}.$$

Since y and z are simply the number of total words in the estimate, this part of the KL divergence is solely dependent on the size of the segments used in the estimate. Given the

relation between y and z, the smaller the total size of the segments used to estimate M_t, the larger the contribution to KL divergence.

Now, let's consider word b. The word b occurs a total of x_b times in the document set. However, only a fraction of the occurrences of b occur in the segments associated with M_t (assume x_{b_t} times), then

$$p(b) = \frac{x_{b_t}}{y} \text{ and } q(b) = \frac{x_b}{z}.$$

In terms of the KL divergence value, the total segment size is weighted by the portion of occurrences,

$$\frac{x_{b_t}}{x_b}$$

of the particular word in the model, M_t, and thus

$$\log \frac{p(b)}{q(b)} = \log \left(\frac{x_{b_t}}{x_b} \times \frac{z}{y} \right).$$

Now we can observe two phenomena. One is that the size of text used to estimate the topic model is inversely proportional to the KL divergence value. The other is that the portion of occurrences of a particular word in the topic model is proportional to the KL divergence value. It can also be observed from this calculation that the $\log$ part of the KL divergence equation is maximized when all the occurrences of a word occur in the subset, and the size of the text used to estimate the model is small. Because the $\log$ value is not the only term in the KL divergence calculation, the discussion above is not complete. The probability of the word, $p(b)$, is as important as the $\log$ value in the KL divergence calculation.

The problem with this approach is that we are likely to encounter extremely sparse data when calculating the topic models, especially as text is removed to account for the selection of a topic word. Because there may be only a few occurrences of a particular word, it will be difficult to determine how well the topic model is approximated. Also, the size of the text used to estimate the model will become increasingly important. As was pointed out

above, the KL divergence values will be larger for topics where tiny amounts of text are used.

Similarly, it is problematic that the KL divergence contributions of individual words will also be partially dependent on the size of the text used to estimate their values. Because the size of the text segments plays such an important role, we will be unable to compare KL divergence contributions of words from two different models. This will lead to difficulties in selecting which particular subtopics should be associated with a topic word.

Due to these difficulties, we utilize another method. In the next section, we develop a more robust approach that does not suffer from the failings of unigram topic models for identifying predictive words.

3.5 Using an Approximation of Relative Entropy for Predictiveness

In this section we develop an alternative approach to finding candidate topic words, that approximates the relative entropy approach. Rather than using a unigram model of the topic, we use a *bigram* model.

In Section 1.4 we stated that the general probability function estimated in a bigram model is $\mathcal{P}(x_i = f_a | x_{i-1} = f_b)$. Such a model will not suit our purposes because we are interested in a model that preserves some of the information generated by calculating separate unigram models for each topic word. To do this we must devise a model that reflects the probability of a topic word occurring in the context of a subtopic, using the same kinds of segments that were developed for estimating the unigram topic model. In essence, our goal is to preserve the term relations present in the topic models, while using a bigram model. Such a model can be built using the probability function:

$$\mathcal{P}(x_i = f_a | \exists x_j \text{ s.t. } i - k \leq j \leq i + k \text{ where } x_j = f_b)$$

where k is the maximum distance from i, which is specified by the user when the hierarchy is created. We will call this expression $\mathcal{P}(t|w)$ where t refers to a topic word and w refers

to another word in the vocabulary. Although this is no longer a generative language model, since it incorporates information about what will occur in the text after the occurrence of the feature, the language model captures the information needed for summarization purposes. Another difference between this language model and more traditional models is that this model is not a probability mass function, so it cannot be used in the KL divergence calculation. The unigram topic model gives us a general description of the text found in the vicinity of the topic word, while this bigram model allows us to ascertain how dependent a word is on the occurrence of a topic word by calculating the probability that the topic word occurs with this other word w.

When estimating the bigram model, an individual probability, $\mathcal{P}(t|w)$, will be calculated by dividing the number of segments that contain t by the total number of segments that contain w. This means that $\mathcal{P}(t|w) = 1$ when t co-occurs with w in every segment of w. In the equivalent situation for the unigram model, all the instances of w occur in the context of t. KL divergence would then have identified w as a strong subtopic candidate.

We use this bigram model to determine which words are the best topics. We will still make use of our primary assumption about topics, which is that the best topic words have a relatively large number of subtopics. In the bigram model, we assume that the best subtopics will be the words that are most dependent on the topic, i.e., where $\mathcal{P}(t|w) \approx 1$. To find how topics compare to one another, we will sum over all the vocabulary, W, in the following manner:

$$\mathcal{P}(\textit{Predictive}(w_i)|w_1 ... w_{i-1}) = \frac{1}{|W|} \sum_{w \in W} P(w_i|w). \tag{3.3}$$

As in the approach using the unigram model, we cannot simply assume that the set of words with the highest probabilities of being topics should be accepted as the main topics of the document set. We must again develop a method to approximate the coverage of a topic in order to make use of the breath-first approach to choosing topics. One way to do this is to consider the relationship between topics and subtopics. We hypothesize that when all

subtopics are associated with at least one topic, then we have found all the main topics. In order to do this, we will develop a method to choose topics in such a manner that we simultaneously associate all subtopics with at least one topic.

One approach to finding the associations is to interpret the language model as a graph. This can be done by creating a *bipartite* graph where each word is represented by two vertices. One vertex represents the word as a topic, and the other vertex represents the word as a subtopic. If we assume a vocabulary of size n, this gives the possibility of n^2 edges in the graph, given that there will only be edges between a vertex in the topic set and a vertex in the subtopic set. This is exactly the number of possible entries in the bigram model. Edges are created between topic vertices and subtopic vertices when $\mathcal{P}(t_i|wj) > 0$. The graph will be a weighted graph where the edge weights are $\mathcal{P}(t_i|wj)$. Given that we can create a graph from the language model, we are able to use a graph theoretic algorithm to solve the problem of finding the main topics of the document set. The question that we would like the graph to answer is: "What is the smallest set of topic words that relates to all the subtopics in the document set?" The question is nearly identical to the question asked in the Dominating Set Problem (DSP): Given a graph $G = (V, E)$, find a subset of vertices $D \subseteq V$ so that for every $u \in V - D$, there is a $v \in D$ for which $\{u, v\} \in E$[17]. This means that a dominating set is a set of vertices where for every vertex, there exists an edge connecting it to an element of the set or the vertex is part of the set. The only difference between the question we would like to answer and DSP is that the only vertices that should be in the set D are those that represent a topic. A solution to DSP is illustrated in Figure 3.3.

The only issue with the Dominating Set Problem is that it is NP-hard[2], even in the bipartite case because the bipartite graph is merely a transformation of the original graph. Because of we cannot compute a solution in polynomial time, we develop a heuristic in

[2]The proof appears in Appendix A.

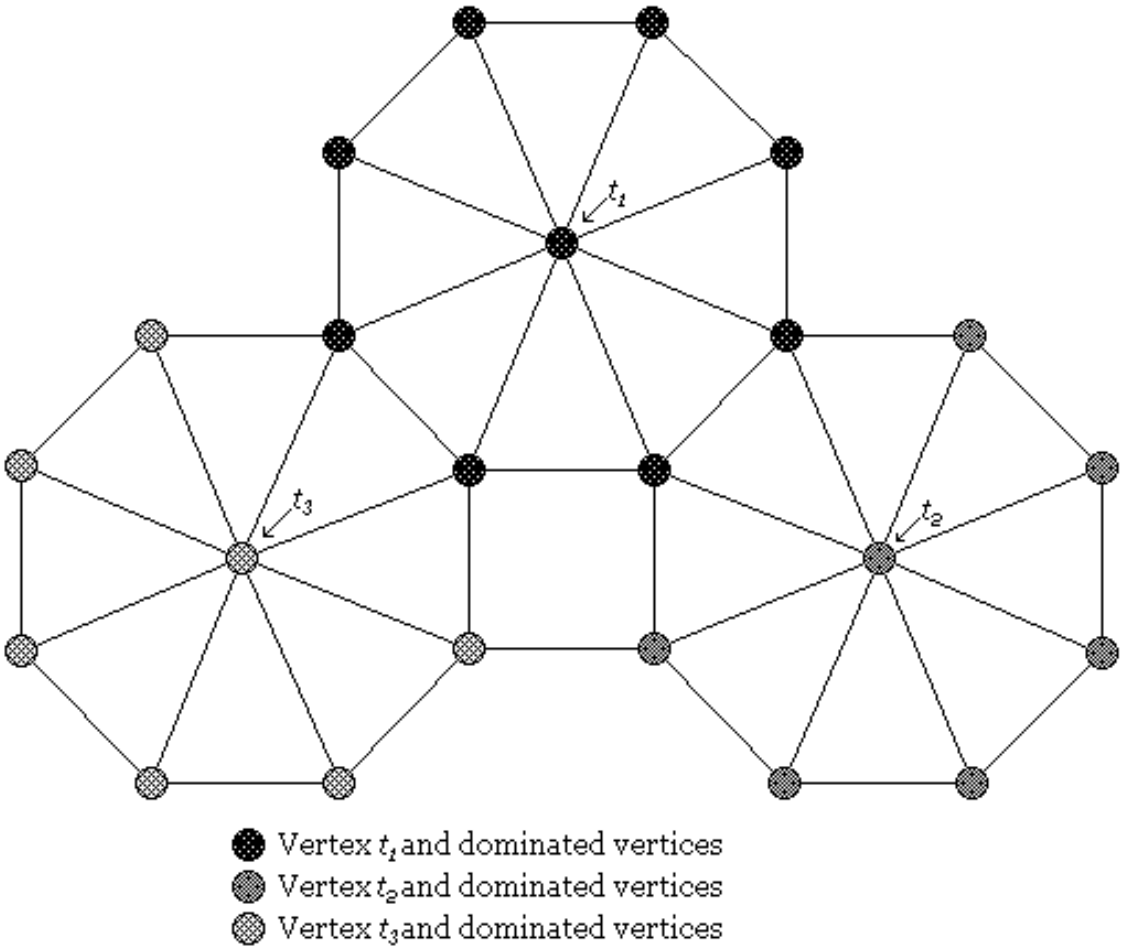


Figure 3.3. Illustrates the Dominating Set Problem. In the graph, there is a dominating set of size 3 consisting of nodes t_1, t_2, and t_3. This graph does not represent a graph consistent with the conversion of a bigram language model.

order to solve the problem. We first observe that each vertex represents either a topic or a subtopic because of the way we created the graph. The only vertices that we are really interested in selecting as topics are the vertices that represent the topic aspect from the bigram model. Because the graph is bipartite graph, the set of topic vertices is easily distinguishable from the set of subtopic vertices. We are not interested in dominating the topic vertices. The important part of the solution is that the each subtopic vertex is dominated by at least one particular topic vertex. Figure 3.4 reconstructs the graph in Figure 3.3 so that it conforms to the transformation of the bigram language model to the graph and demonstrates how the subtopic vertices are dominated while the topic vertices are not.

The heuristic to DSP makes use of Equation 3.3. In order to make our first selection of a topic, we find the weight of each vertex by summing over all the edges connected to that vertex. We then greedily choose the first topic vertex. By selecting the heaviest node as the first topic word, we are selecting the topic word that maximizes Equation 3.3. Once we have selected a topic, we use the graph to determine which subtopics are dominated by that topic. In the standard DSP graph, the edges are unweighted. In this case the edges do have weights. They can be used to determine which subtopics should be dominated by a topic. Since we are only interested in subtopics where there is a clear dependence between the topic and subtopic word, we can use the edge weight to determine which subtopics are dominated by a particular topic. Once a subtopic vertex is dominated, it will not need to be dominated by another topic. In essence, we will ensure summarization of the complete document set by ensuring that each individual subtopic vertex is dominated by at least one topic vertex.

Before choosing the second topic, the topic vertex weights are recalculated by summing over the edge weights of the remaining subtopics. We alter Equations 3.3 to

$$\mathcal{P}(\textit{Predictive}(w_i)|w_1 \ldots w_{i-1}) = \frac{1}{|W_i|} \sum_{w \in W_i} P(w_i|w), \tag{3.4}$$

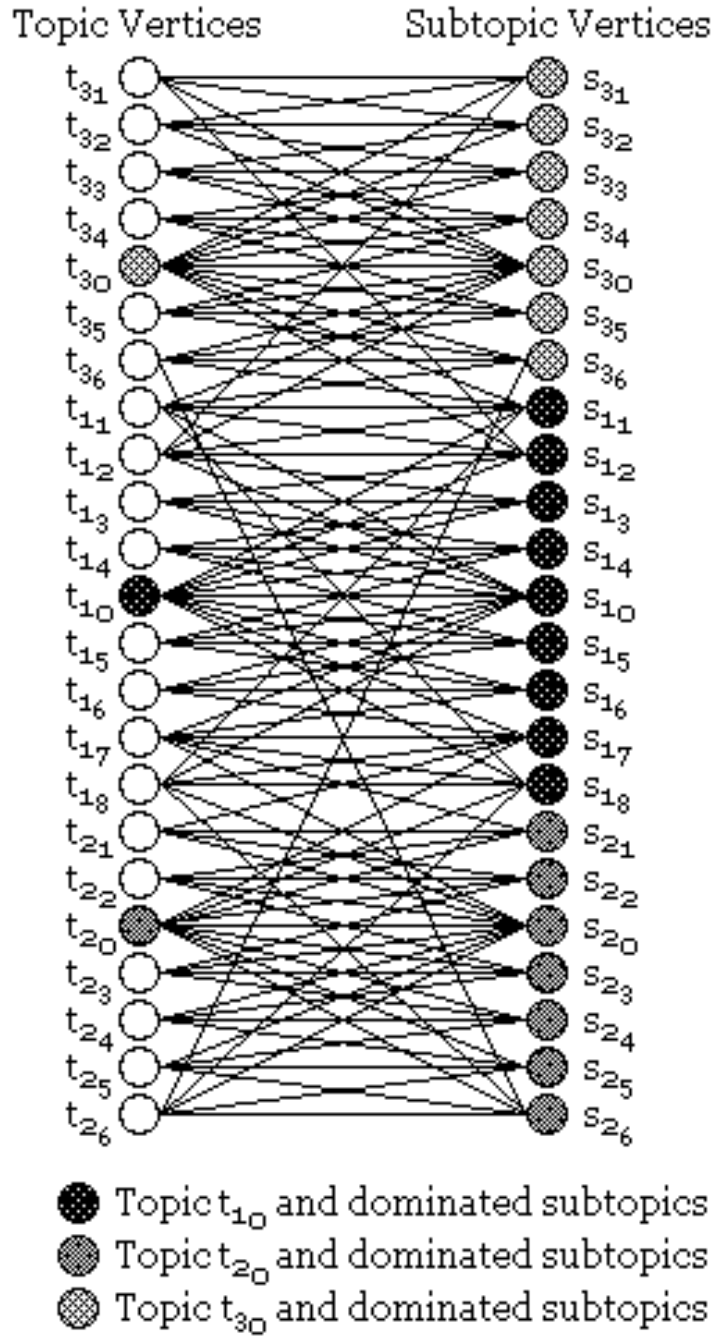


Figure 3.4. Illustrates the transformation of the bigram language model into a graph. In the graph, there is a subtopic dominating set of size 3 consisting of nodes t_{1_0}, t_{2_0}, and t_{3_0}. It demonstrates the bipartite nature of the graph and the fact that not all the nodes in the graph are dominated as would occur in the Dominating Set Problem.

where $W_i = W_{i-1} - S_{i-1}$ and $S_{i-1} = w_{i-1} \cup \textit{subtopics}_{w_{i-1}}$. In this case, the set $\textit{subtopics}_{t_1}$ are the dominated subtopics, which are removed from the vocabulary to create W_i. We again greedily choose the next topic word. We will stop choosing topics when all the subtopics are in the set W_i. In order to use Equation 3.4 for our general estimation, we define $W_0 = W$ and $\textit{subtopics}_0 = \emptyset$. This process can be visualized using Figure 3.5, which illustrates the set of subtopics and groups them according to the topics that dominate them.

To calculate predictiveness, the bigram language model described earlier in this section is constructed. A given entry in the model can be expressed as $\mathcal{P}(t|w)$ where t is a possible topic word and w is another word in the vocabulary. The function $\mathcal{P}(t|w)$ is estimated using the following formulation:

$$\mathcal{P}(t|w) = \frac{\text{segments}_x(t \cap w)}{\#\text{occurrences}(w)},$$

where $\text{segments}_x(t \cap w)$ refers to the number of segments of size x where t and w co-occur. The segment size should capture the maximum distance where associations between the words are probable. It is a parameter set when generating a hierarchy. In Section 6.2.8 we examine effects of different values for this parameter. After computing the bigram model, the probability of predictiveness is estimated using Equation 3.4.

By changing the language model and the set W_i for different levels on the hierarchy, we are able to identify subtopics of topics, and thus generate the hierarchy recursively. Initially, the language model is created using all the text that the hierarchy is summarizing. The top level words are selected based on their topicality and predictiveness. However, when finding subtopics of a topic, the language model is recalculated over the text that the parent topic word originally predicted, which is determined by the segment size. This biases the model to the type of text that occurs near the topic word. Again the most topical and predictive words are chosen as subtopics, but because of the bias of the language model and the vocabulary set, the words are only topics in the context of the parent topic.

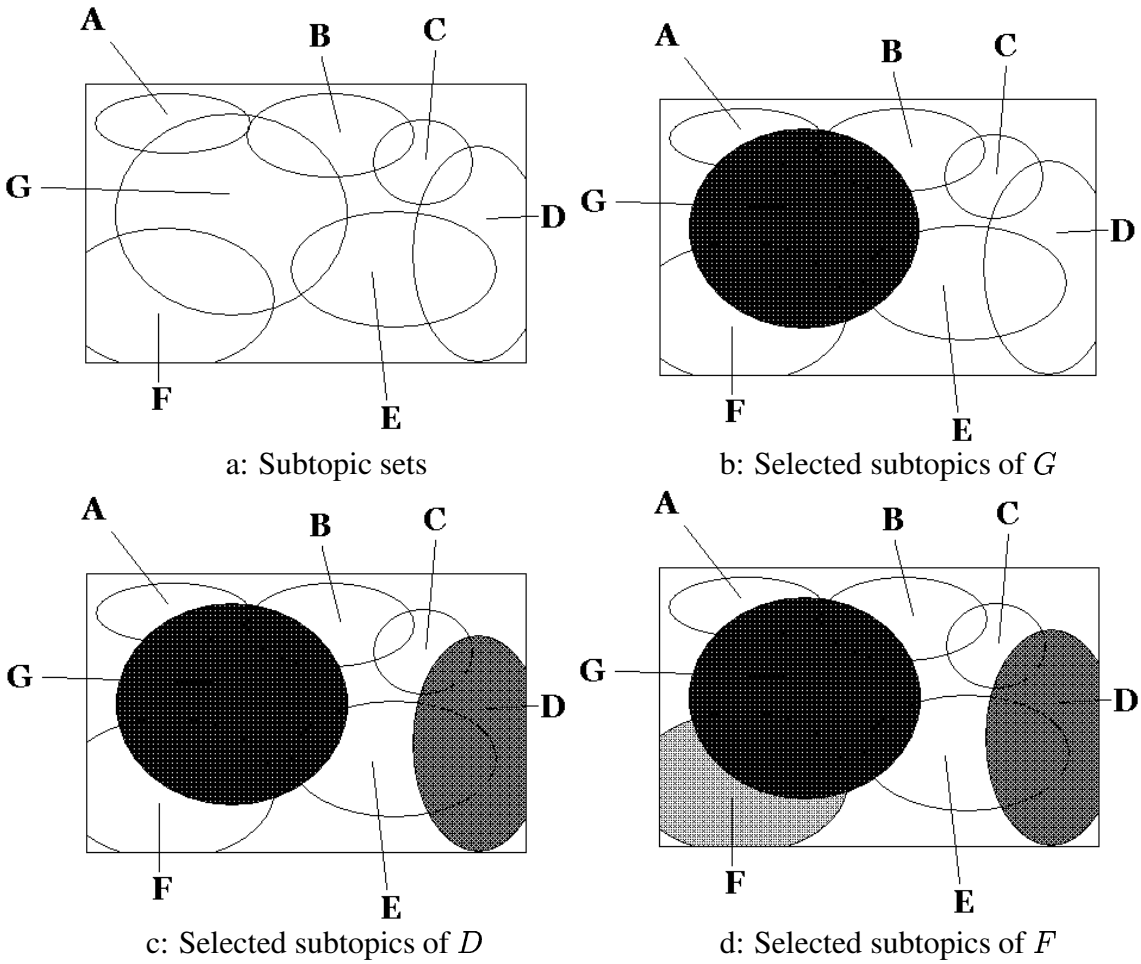


Figure 3.5. Illustrates how topics are selected using subtopic sets. First all subtopics are associated with topics as shown in Part a. After G is selected as the first topic word, all the subtopics are removed from the set of possible subtopics as shown in Part b. Then D is chosen as the second topic and its subtopics are removed as shown in Part c. Finally, F is selected as a topic as shown in Part d. Since some of its subtopics are associated with G, they were not used to decide that F is a topic, although some subtopics associated with G may turn out to be subtopics of F.

3.6 Implementing the Term Selection Formula

Now that we have developed the Term Selection Formula and methods for estimating the values, we need an algorithm that will implement it. In this section we introduce the algorithm **DSPapprox**, which is a greedy heuristic to the Dominating Set Problem for graphs. The algorithm appears in Figure 3.6.

$\mathbf{DSPapprox}(G, CT, P_{\mathcal{T}}, k)$
// Initialize variables
(1) $ST = V - CT$
(2) $T = \emptyset$
(3) $DominatedSubtopics = \emptyset$
(4) $thresh = \mathbf{mean}(w_e(CT, ST))$

(5) foreach $c \in CT$
(6) $\quad w_v(c) = P_{\mathcal{T}}(c) \times \frac{1}{|ST|} \sum_{s \in ST} w_e(c, s)$

(7) while ($DominatedSubtopics \neq ST$ and $|T| < k$)
// Find the best topic and the subtopics associated with it
(8) $\quad t = \arg\max_{c \in CT} w_v(c)$
(9) $\quad t_{subtopics} = t_s$ where $s \in t_s$ if $w_e(t, s) \geq thresh$
// Update the variables
(10) $T = T \cup t$
(11) $CT = CT - t$
(12) $DominatedSubtopics = DominatedSubtopics \cup t_{subtopics} \cup t$
(13) foreach $s \in t_{subtopics}$
(14) $\quad$ foreach $c \in CT$
(15) $\quad\quad w_v(c) = w_v(c) - (P_T(c) \times w_e(c, s))$

(16) return T

Figure 3.6. Dominating Set Approximation algorithm. It requires as inputs G (the graph), CT (the candidate topics), $P_{\mathcal{T}}$ (the topicality model), and k (the maximum number of topics desired). The algorithm returns the chosen set of topics, T, which will be a complete or partial dominating set of the subtopics.

This algorithm takes as inputs the graph, G, which consists of all vertices, edges, and edge weights; the candidate topics, CT, which are that portion of the vertices representing the candidate topics; the vector, $P_{\mathcal{T}}$, which are the topicality estimates of each candidate

topic; and a number k that provides a cut-off for the number of topics requested. The graph G is created using the transformation of the bigram model described in Section 3.5. The edge weights of the graph are the values in the bigram language model. The values in the vector $P_{\mathcal{T}}$ are the KL contributions calculated for topicality.

The first group of lines in the Figure 3.6 initialize the variables that will be used in the computation. In the first line, we identify the vertices that represent the subtopic. We then initialize T, the set that will hold the vertices chosen as topics, and $DominatedSubtopics$, the set that will hold all vertices that are connected to a vertex in T by an edge. Because we are trying to find true topics rather than just dominate the subtopics, the mere existence of an edge is not sufficient proof that the candidate is the topic for a particular subtopic. In order to be more certain that a word is a topic for a given subtopic, we test for the validity of the relationship by imposing a threshold. However, since we expect some document sets to be more closely related then others, we choose a document-set-dependent threshold. Here, we use the mean of all edge weights.

The second group of lines in Figure 3.6 calculates the vertex weights. These weights are estimates of the joint probability of topicality and predictiveness calculated using Equations 3.2 and 3.4 as shown in line six. Since subtopics will never be chosen as topics, it is unnecessary to calculate weights for subtopic vertices, and so the fifth line in Figure 3.6 limits the calculation of vertex weights to candidate topic vertices.

After the vertex weights are calculated, topics are selected. In line eight of Figure 3.6, we choose the heaviest vertex, t, from the set of candidate topics to be a member of the set T. We then determine the set of vertices adjacent to t that are dominated by the topic using the threshold that was calculated. Edges with weights that are less than the threshold are part of the overall weight of the vertex but are not used to determine which subtopics are dominated. This is because we expect topic words to have relationships with non-subtopic words as well as with subtopic words. This accumulation of infinitesimal weights

allows one to distinguish topics from the words they dominate by breaking the symmetries inherent in the measure.

In order to ensure that the second topic selected is related to different subtopics, we adjust the weights of the vertices by subtracting the edge weights and topicality of the subtopics dominated by t. The algorithm loops, picking the heaviest vertex each time. At each step, the heaviest vertex remaining is added to the set T. We continue to augment T until either all the subtopic vertices are in the set of dominated subtopics, or we accumulate k topics.

DSPapprox is an efficient algorithm. Given s subtopics, t candidate topics, and a goal of selecting k topics, the algorithm performs in $O(kts)$ time. Since we actually have the same number of topics and subtopics, the performance time can be simplified to $O(kt^2)$. Although we use the same words as topics and subtopics, it would be possible for the candidate topics to be merely a subset of the subtopics. In fact, one way to improve the performance is to carefully choose the candidate topics by allowing only those words that have the qualities of a topic or subtopic to be part of the candidate topics.

3.7 Putting It All Together

This section gives a step-by-step account of how a document set is used to produce a hierarchical word-based summary. Figure 3.7 illustrates the main steps in this process. For the purposes of this discussion, we use a set of 50 documents retrieved for the query "Endangered Species – Mammals". This document set contains one news article about spotted owls, several documents by the National Wildlife Service discussing the merits of listing particular species as endangered, a few documents that contain information about permits allowing the harassment of marine mammals for scientific research, others describing rules that fisherman have to follow when dealing with marine mammals, and debates in Congress on the renewal of the Marine Mammal Protection Act.

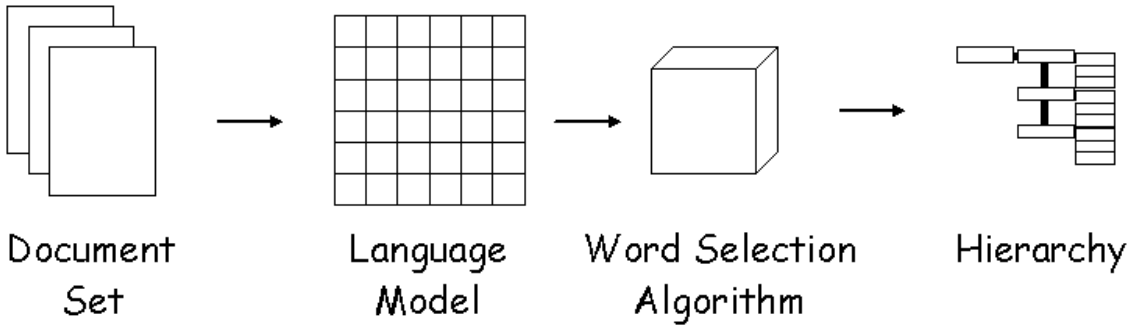


Figure 3.7. Illustrates the process of generating a hierarchy. From the original document set, language models are constructed. The language models are used by the word selection algorithm to find topic and subtopic words. The algorithm outputs a hierarchy.

Given a document set, the next step is to construct the unigram language model for the set, so that the topicality model can be estimated. Using the Endangered Species document set, Figure 3.8 illustrates what the unigram model looks like. Since stopwords have been disqualified as candidate topic words, they are absent from the unigram language model. A unigram model of general English is also constructed. For this document set, we estimate the general English language model from TREC volumes four and five, which consist of about 555,000 news and government documents – just over 2 gigabytes of information.

The unigram language model for the document set is compared to the unigram language model for general English as is shown in Figure 3.9. This figure demonstrates that the words whose probability of occurring differs the most from that of general English are the words that are ranked highest by KL divergence. The KL divergence score is used to estimate each word's probability of being a topical word in the topicality model. The probability that a given word is a topical word is set to 1 for the word with the highest contribution to KL divergence, while the word with the lowest KL divergence contribution is assigned a probability of 0. All other words have some probability between 0 and 1, calculated by 3.2. Because phrase occurrence statistics are not present in the database that we use to estimate the language model for general English, we must treat phrases differently than single words

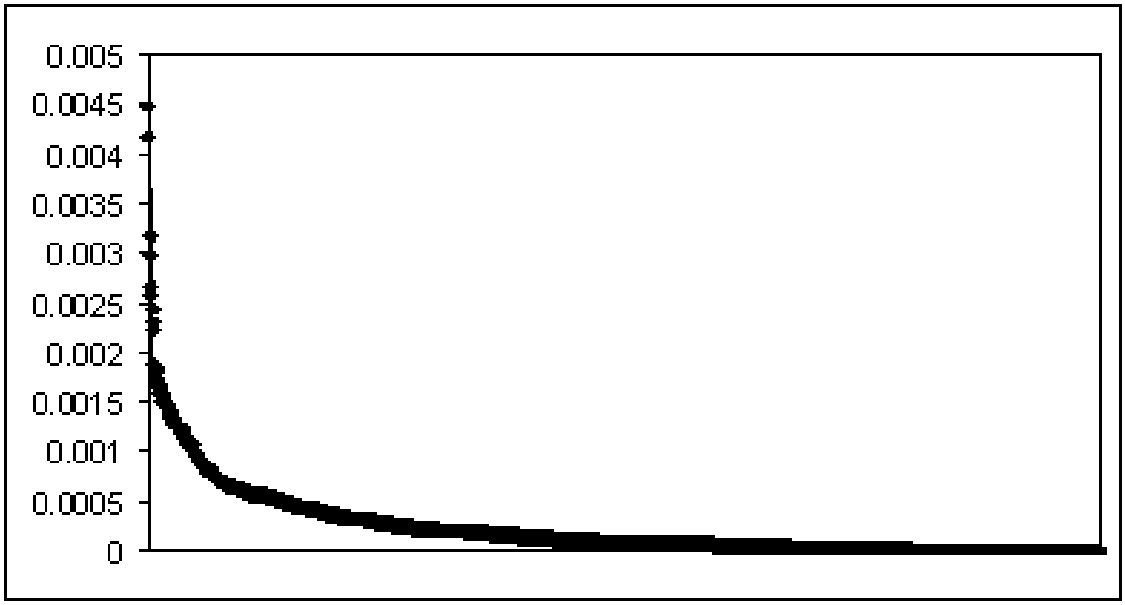


Figure 3.8. Illustrates the unigram model for the "Endangered Species" document set. The unigram model does not include stopwords. In this particular set, "time" is the most frequent non-stopword, followed by "state". The probability distribution is truncated due to space on the page. Many single occurrence terms are not represented in this figure.

when estimating their probability of topicality. We do this for a given phrase p by summing over the individual word's KL contribution in following manner:

$$\text{KL contribution}(p) = \sum_{w \in p} \text{KL contribution}(w).$$

When determining minimum and maximum values for KL contribution, the KL contributions of both single words and phrases are considered. This summation most likely overestimates the probability of topicality for phrases. This is desirable because in general phrases are better topic and subtopic words than single words. We believe phrases are better because they tend to be more specific.

Unlike the estimate of topicality, which is static for the entire construction of the hierarchy, the estimate of a word's predictiveness changes with each level of the hierarchy. Before estimating predictiveness, we construct the bigram model discussed in Section 3.5.

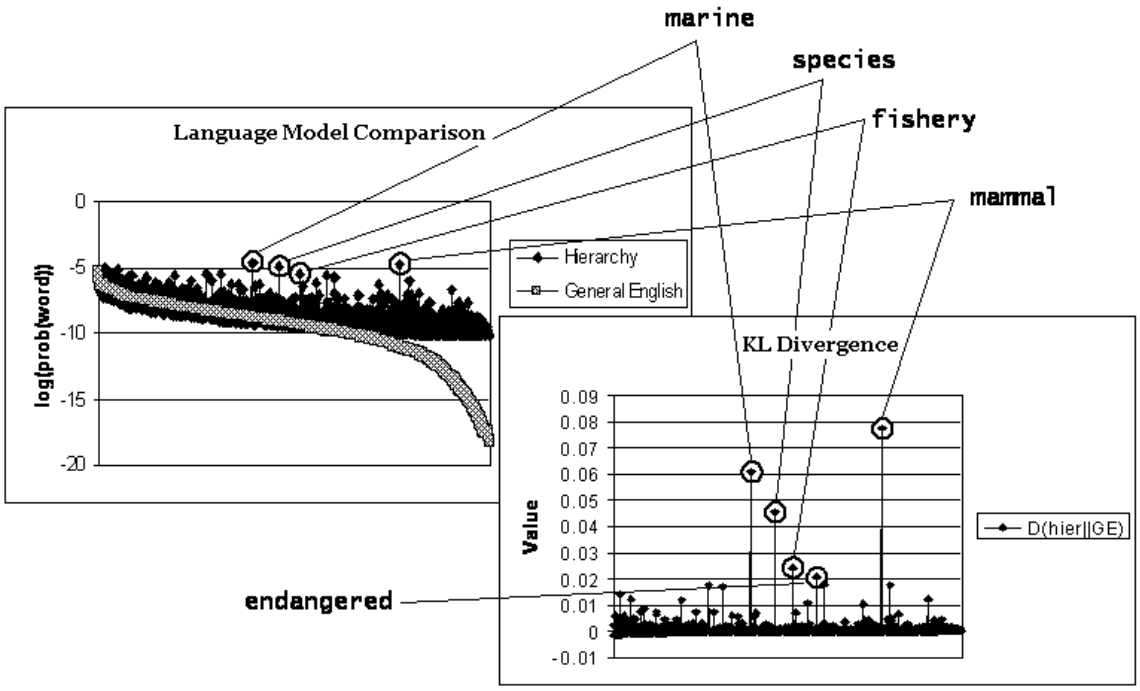


Figure 3.9. The left side of the figure illustrates the comparison of the unigram model for the "Endangered Species" document set and the general English model. The words are ordered by their frequency in general English. The right side of the figure shows the KL divergence for different words. The words that were ranked most highly by KL divergence are "mammal", "marine", "species", and "fishery". The lines connecting the graph show how the frequencies that were most different from general English are the top ranked words in KL divergence. The word "endangered" was the fifth most highly-rated word. Three of the top five words are query words and are thus highly related to the topic of endangered species, and more importantly, the document set that was used to estimate the unigram model.

Figure 3.10 shows a small portion of the bigram model constructed for the top level of the "Endangered Species" document set. These individual probabilities will be used in Equation 3.4 to estimate the probability of predictiveness.

With the language models estimated, the topic words are selected using the Dominating Set Approximation algorithm in Figure 3.6. For this particular hierarchy, we set the maximum number of topics to $k = 10$. The algorithm also requires the bigram language model to be transformed into the graph $G = (V, E)$. The portion of vertices that are associated with the topic aspects of each word is specified in the set CT. The algorithm uses vertex

topics (t) \ subtopics (w)	mammal	marine	species	fishery
mammal		.98	.31	.35
marine	.99		.31	.35
species	.65	.65		.50
fishery	.04	.03	.01	

Figure 3.10. Shows the estimated values of the bigram model for the 4 words that are most likely to be topic words, according to their probability of topicality.

weights to find the best topic candidates. These weights are computed using Equations 3.2 and 3.4. When finding topics for a level, the algorithm sets a threshold that will be used to decide whether the relationship between topic and subtopic vertices is strong enough to consider the topic a dominating vertex for a particular subtopic. In order for the threshold to reflect the fact that the inter-relatedness of different document sets will vary, we choose a document-set-dependent threshold. In this case, we use the mean of all the bigram model probabilities. When finding the top level topics for the "Endangered Species" document set, the threshold was found to be 0.216159. Using this criterion, the five topic words were found to predict all the subtopics:

- marine mammal
- species
- marine
- Marine Mammal Protection Act
- management plan

To find the subtopics of "marine mammal", the bigram language model is recomputed using only the segments of text where "marine mammal" occurs. The new bigram model

is transformed into a graph. The Dominating Set Approximation algorithm calculates the threshold to determine dominating relationships as 0.310097. Notice that topic and subtopic words are more closely related within the context of "marine mammal" than they were for the top level topics. This time ten topics are found, which is the value of k for this hierarchy. The topics are

- Marine Mammal Commission
- marine mammal stocks
- National Marine Fisheries Service
- fishery
- permit
- Merchant Marine
- Endangered Species Act
- fish
- population
- Act

The process of constructing the bigram model and choosing topic words continues until the hierarchy reaches the depth requested by the user or the deepest subtopics have no more than five documents associated with them. A portion of the fully constructed hierarchy appears in Figure 3.11.

3.8 Conclusion

In this chapter we developed a methodology for finding topic words for a hierarchal word-based summary using probabilistic language models as a method for describing the

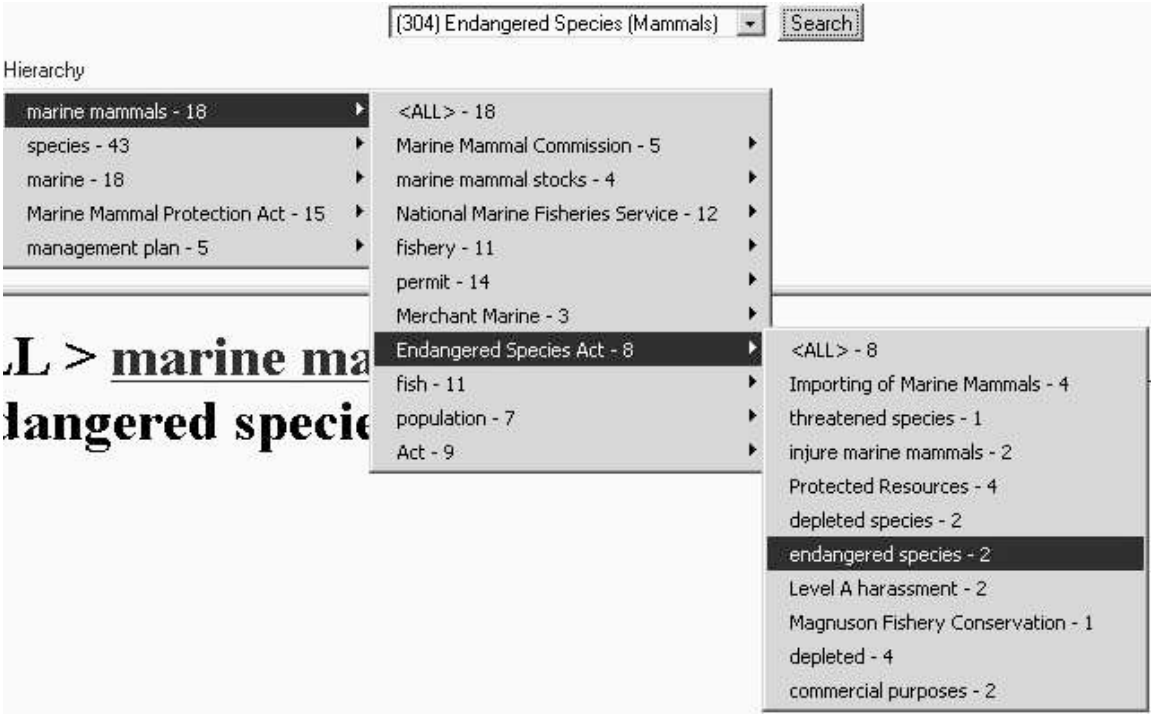


Figure 3.11. Here is a portion of the complete hierarchy constructed from 50 documents retrieved for the query "Endangered Species - Mammals". This hierarchy was constructed with text segments of size 200 for all levels. It did not use a query-biased or sub-collections when calculating the topic model. It did, however, leave out words that contributed negatively to the KL divergence.

text. Although we could not use relative entropy to completely solve our problem, we were able to develop an approximation of relative entropy using a bigram language model. Specifically, we use the Term Selection Formula specified in Equation 3.1 to find the best topic terms. Finally, in Section 3.7 we demonstrated how each of the pieces fit together and can be used to find topic words through the Dominating Set Approximation algorithm which implements the Term Selection Formula.

CHAPTER 4

EXAMPLES

4.1 Collection Statistics

In order to build the hierarchical summaries, we require a set of documents to summarize. We use sets of documents retrieved from queries as the text for the summaries. For the purposes of experimentation, we use 150 queries that have been written by NIST for the Text Retrieval Conference (TREC). The advantage of using these standard queries is that some documents in the associated document sets have been judged based on relevance to the query, and we can use this fact as a means of evaluating the hierarchies.

The particular queries used in our experimentation are TREC topics 201 through 350 inclusive. These queries are divided into sets of 50 and each set of 50 has a different associated collection of documents. The queries 201 to 250 are associated with a collection of text from the Wall Street Journal years 1990 to 1992; the Sun Mercury News of 1991; the Associated Press of 1988 and 1990; the Federal Register of 1989; U.S. Patents from 1983 to 1991; and the Computer Select disks of 1989 to 1992 copyrighted by Ziff-Davis. This collection consists of about 570,000 documents. Queries 251 to 300 are associated with a collection of text from the Wall Street Journal years 1990 to1992; the Associated Press of 1988; the Financial Times Limited from 1991 to 1994; the Congressional Record of the 103rd Congress of 1993; the Federal Register of 1989 and 1994; and the Computer Select disks of 1989 and 1990 copyrighted by Ziff-Davis. This collection consists of about 525,000 documents, some of which are associated with topics 201 to 250. The queries 301 to 350 are associated with a collection of text from the Financial Times Limited from 1991 to 1994; the Foreign Broadcast Information Service of 1996; the Los Angeles Times of

1989 and 1990; the Congressional Record of the 103rd Congress of 1993; and the Federal Register of 1994. This collection consists of about 555,000 documents.

Each of these collections has some text from news and some text from other sources. Because some of the non-news documents are quite long, we decided to create additional collections from the news articles only. This means that for each collection, we created one database with all the documents enumerated in the previous paragraph, and a second database with only the news articles. The news collection that is associated with queries 201 to 250 consists of about 325,000 documents; the news collection associated with queries 251 to 300 contains about 365,000 documents; and the news collection for queries 301 to 350 totals about 475,000 documents.

In order to create the document sets, we retrieved 500 documents for each of the 150 queries. Each query was run on both the complete database and the news database for which it is associated. The actual queries we used consisted of the topic title and description provided by TREC, as well as other words added to the query using Local Context Analysis[72], which expands the query to improve retrieval.

We are also interested in the ability to create hierarchies as a summarization for the results of a web search. Because these hierarchies are created in an interactive environment, the speed with which the hierarchy can be built is paramount. Therefore, rather than downloading each of the retrieved documents and analyzing its full text, we chose to build the hierarchy from summaries of the retrieved webpages, also known as snippets, which are provided by the search engine and can be quickly accessed. For these collections of documents, we retrieved between 500 and 1000 snippets. In most cases we used the topic title as the query to retrieve the document set. However, TREC topics 201-250 were not assigned titles, so we created 1 to 4 word queries from their descriptions. For a few of the other queries, we were unable to retrieve 500 documents with the title provided; in these cases we modified the query in order to achieve the minimum size. The modified topic

titles appear in Appendix C. Table 4.1 contains the average number of documents retrieved for each query set, as well as the largest and smallest group of documents.

Collection	*Mean*	*Minimum*	*Maximum*
201 -250 Web	763.6	563	881
251-300 Web	763.2	540	888
301-350 Web	760.8	576	880

Table 4.1. Information about the number of documents retrieved from the web for the different sets of queries.

The expected length of a document from our three types of collections varies greatly. Table 4.2 provides the average lengths of the nine different groups of documents. As expected, the average length of the documents for each collection of the same type is fairly consistent, but there are large differences in the expected document size among the complete, news, and web data sets. The average length of the documents are illustrated in Figure 4.1.

Collection	*Mean*	*Median*	*Varience*	*Stan. Dev.*
201-250 Complete	1184.13	925.638	599513	774.28
251-300 Complete	3495.42	2905.97	5753831	2399
301-350 Complete	2029.61	1602.13	1930932	1390
201-250 News	368.72	367.96	1998	44.70
251-300 News	405.60	405.64	2838	53.28
301-350 News	760.53	693.54	76778	277.09
201 -250 Web	18.35	18.40	4.66	2.16
251-300 Web	18.39	18.60	5.00	2.24
301-350 Web	18.19	18.27	4.15	2.04

Table 4.2. The average length of a document for each query/database combination. Including non-news documents gives the largest average size, and the snippet documents are the shortest.

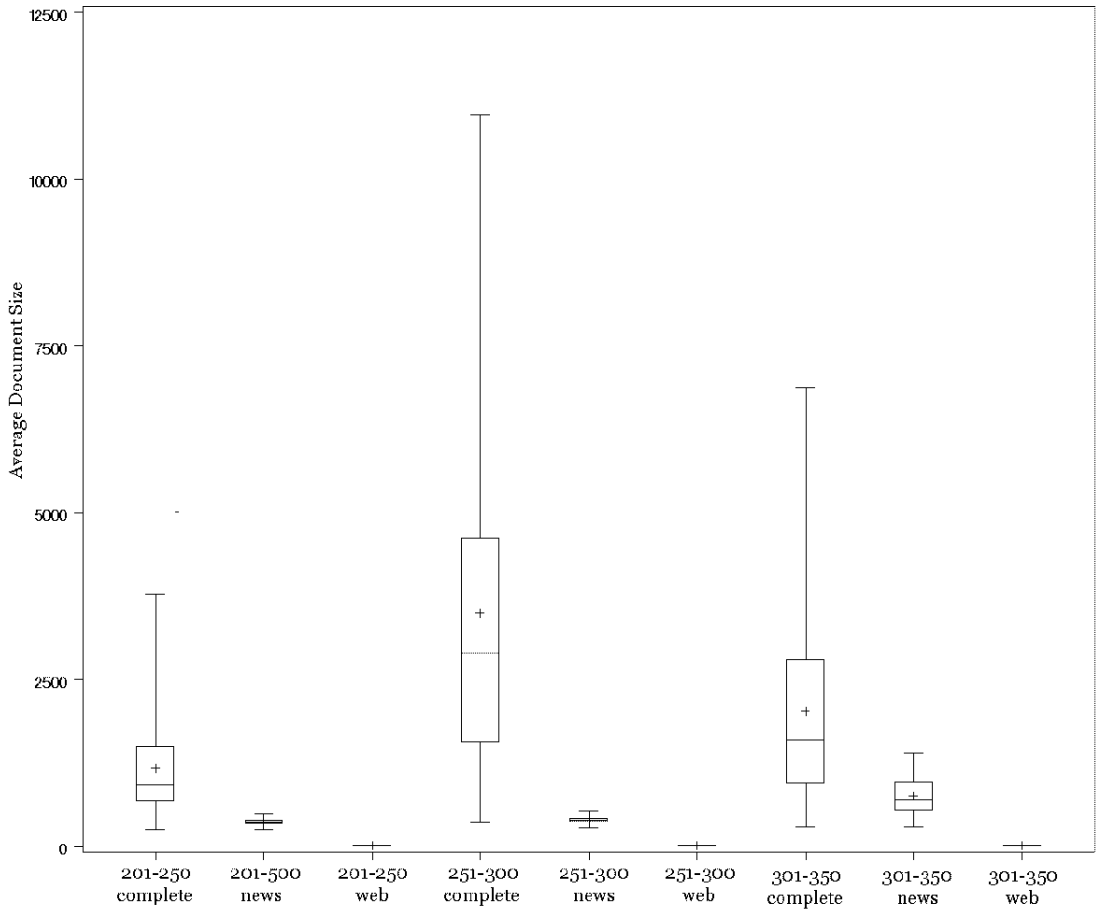


Figure 4.1. Box plot illustrating the average document length for each query in the set. See Appendix B for an explanation of how to interpret this data.

4.2 Hierarchy Characteristics

In this section we explore the differences between hierarchies created from the full text of documents and hierarchies created from snippet text. We also enumerate some common features associated with these two types of hierarchies, such as the variety of phrases found in the hierarchy and the expected number of documents at each node for a given the depth[1] of the hierarchy.

Since the snippets of text are compiled by the search engine based on the words in the query, the snippet hierarchies include words that are much more closely related to the query than the full text hierarchy words. Figures 4.2 and 4.3 illustrate this point. The hierarchies created in both figures summarize documents retrieved for the TREC Topic "Ferry Sinkings". Figure 4.2 shows that the documents about ferries are not necessarily about ferry sinkings, since "sinkings" is not a subtopic of ferry. The topic words "ferry operators", "ferry companies", "ferry terminal", and "ferry crossings" seem to be more generally related to ferries than to ferry sinkings. Some of these documents are in fact about ferry sinkings, however. Half of the documents under the heading "ferry→ferry operators→passenger ferries" and "passenger shipping" were judged relevant. However, the most concentrated group of relevant documents is under the heading "ferry→ferry Estonia", where 14 of the 16 documents were judged relevant. In contrast, half of the subtopics of "ferry" in the snippet hierarchy shown in Figure 4.3 contain the phrase "ferry sinking", and the others are about specific ferries. From a user perspective, the snippet hierarchy may be more satisfying because it leads one to believe there is more relevant information. This is a general phenomenon that one can observe throughout the examples presented in this chapter.

Another difference between the hierarchies is the number of documents attached to nodes at different levels of the hierarchy. In the snippet hierarchy shown in Figure 4.3,

[1] Depth refers to the number of levels from the root of the hierarchy.

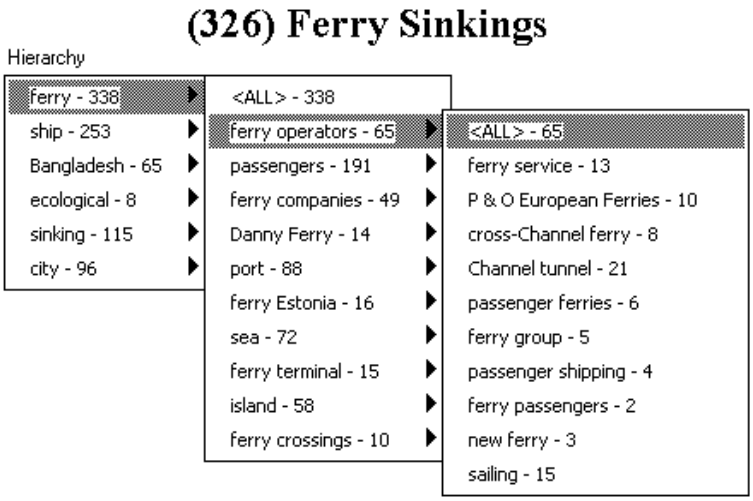


Figure 4.2. The hierarchy for TREC Topic 326: Ferry Sinkings summarizes news documents retrieved for the query.

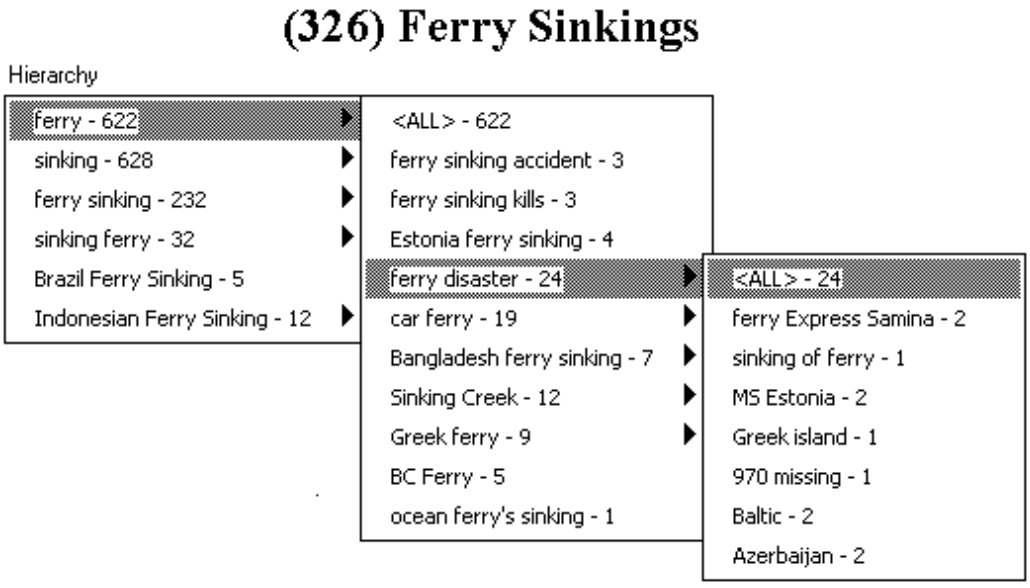


Figure 4.3. The hierarchy for TREC Topic 326: Ferry Sinkings summarizes web snippets retrieved for the query.

there are 622 documents that contain the word "ferry". However, the largest subtopic group ("ferry disaster") contains just 24 documents. This means that the 10 subtopics cannot possibly summarize all of the 622 documents. In the full text hierarchy shown in Figure 4.2, there are 338 documents that contain the word "ferry". The subtopic groups break the documents into sets of 191, 88, 72, and so on down to 10. A much greater percentage of the 338 documents are present among the subtopics. The exclusion of documents at each subsequent level of the snippet hierarchy is another common trend throughout the examples presented in this chapter. Table 4.3 summaries the average number of documents found at a node for a given level of the hierarchy. In light of the example, it is not surprising that the second and third level full text hierarchy nodes are generally much larger than snippet hierarchy nodes at the same levels.

Collection	*Depth 0*	*Depth 1*	*Depth 2*
201-250 Complete	94.9%	94.9%	91.5%
251-300 Complete	91.5%	91.5%	86.1%
301-350 Complete	93.0%	93.0%	88.5%
201-250 News	93.7%	93.7%	90.3%
251-300 News	92.9%	92.9%	92.9%
301-350 News	93.0%	93.0%	88.9%
201 -250 Web	89.0%	88.6%	65.9%
251-300 Web	88.0%	87.5%	65.6%
301-350 Web	89.4%	88.8%	65.4%

Table 4.3. The Average Size of nodes for different depths in the hierarchy

4.3 Examples of Snippet Hierarchies

Search engines have become very good at ranking relevant webpages for certain types of queries. Unlike 5 to 10 years ago, when a search for a company web page would often rank personal home pages above a company's own page, today the company page is likely to be the top ranked page. However, there are other types of queries where web search engines do not perform nearly as well. This occurs frequently when the query is a search

for general information on a topic. Such topical searches are where hierarchies are most useful. The hierarchy provides an alternative to browsing a ranked list, and can be a more effective organization of the documents that are retrieved, thus allowing users to find more relevant information.

Topic hierarchies are very good at assisting a user in several different ways. First, they make it easy to view portions of a ranked list that would not otherwise be seen, and thereby find relevant documents that are not highly ranked. Second, a hierarchy more effectively indicates when the retrieval does not go well. A user may look at several pages of results from a ranked list before finally concluding that there is nothing relevant, while a hierarchy allows the user to determine this almost immediately. Third, the hierarchy can bring alternative uses of a word to the user's attention without requiring her to reformulate her query.

To demonstrate these claims, we have selected a few hierarchies from TREC topics[66] built using our system. These hierarchies were generated by first sending a short query (the TREC "title" field) to the Google search engine[21]. Then we used the titles and snippets that Google returned as the text to generate the hierarchies.

4.3.1 Hubble Telescope Achievements

Figure 4.4 shows the hierarchy generated from snippets retrieved for the query: "Hubble Telescope Achievements". The top part of the figure is a portion of the hierarchy that the user would see when interacting with our system. The list that follows is the portion of the ranked list that would be seen by navigating to "achievements→Hubble→Hubble's achievements" and selecting the topic "Hubble's achievement". The rank numbers are the same as those assigned by the search engine.

The topic "achievements→Hubble→Hubble's achievements" includes eleven documents, although only the first six appear in the figure. Of the eleven documents, three of them are about Hubble telescope achievements, including the ones shown in Figure 4.4

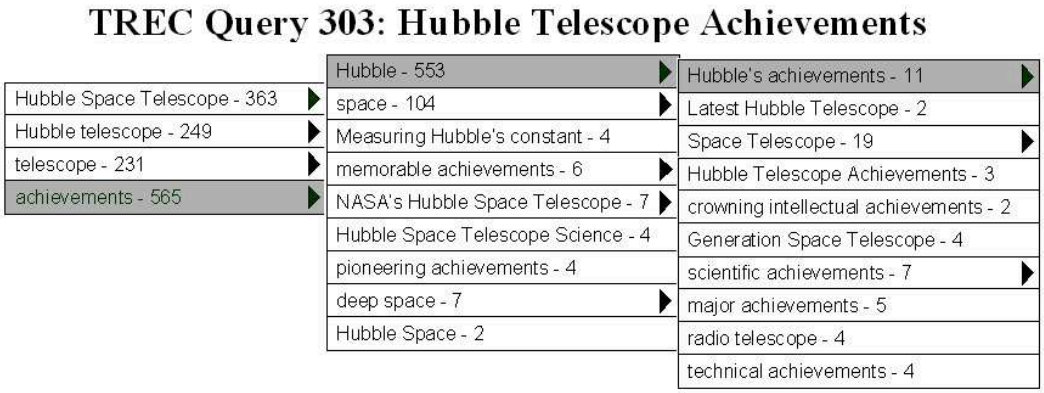
that rank at 100 and 140. In fact, the HubbleSite, which was ranked at 140, is a site devoted to the telescope and includes a fairly in-depth discussion of its achievements. Because of the site's low rank, it is unlikely a user would have found it without using a tool such as the hierarchy.

4.3.2 Abuses of E-mail

The hierarchy created for the query "Abuses of E-mail" appears in Figure 4.5. In this particular case the quality of the retrieval is poor, and this fact is revealed by the hierarchy. Of the six high level topics, only two show any promise as avenues to finding documents about E-mail abuses. The fact that only 274 of the 811 documents retrieved contain the word "E-mail" also supports our claim about the quality of the retrieval. The two promising avenues are expanded in Figure 4.5. The topic "unsolicited commercial e-mail" found under "abuses" summarizes the predominate type of e-mail abuse that websites discuss, also known as spamming. It may have been very difficult for a user to draw such conclusions about retrieval quality by looking at the top of the ranked list.

4.3.3 Airport Security

Figure 4.6 shows the hierarchy created for the query "Airport Security" using snippets. Most of the 853 documents retrieved for this query have to do with the problem of ensuring the security of people while they travel on airplanes, which is the predominant meaning of airport security. However, a user searching for information on Apple's AirPort security using the query "Airport Security" would likely not be persistent enough to find the documents pertaining to this topic using an ordinary ranked list. Because the hierarchy accounts for these minority topics, this small group of documents is easy to locate under the topic "Apple Boost AirPort Security". Although the hierarchy does not contain many documents about Apple's AirPort security, the hierarchy would allow a user to locate them quickly.



achievements→Hubble→Hubble's achievements

Rank 19: *Books with Pictures From Space (Science U)*
... of Our Cosmos, by Simon Goodwin A gallery of the most signifi cant photographs taken by the **Hubble** telescope explains what **Hubble's achievements** can ...

Rank 34: *Amazon. COM: buying info: **Hubble's** Universe: A Portrait of Our ...*
... Ingram A gallery of the most signifi cant photographs of space as taken by the **Hubble** telescope explains what **Hubble's achievements** can tell us about the ...

Rank 63: *ESA Portal - Press Releases - HST's 10th anniversary, ESA and ...*
... A public conference will take place in the afternoon to celebrate **Hubble's achievements** midway through its ... Notes for editors. The **Hubble** Space Telescope ...

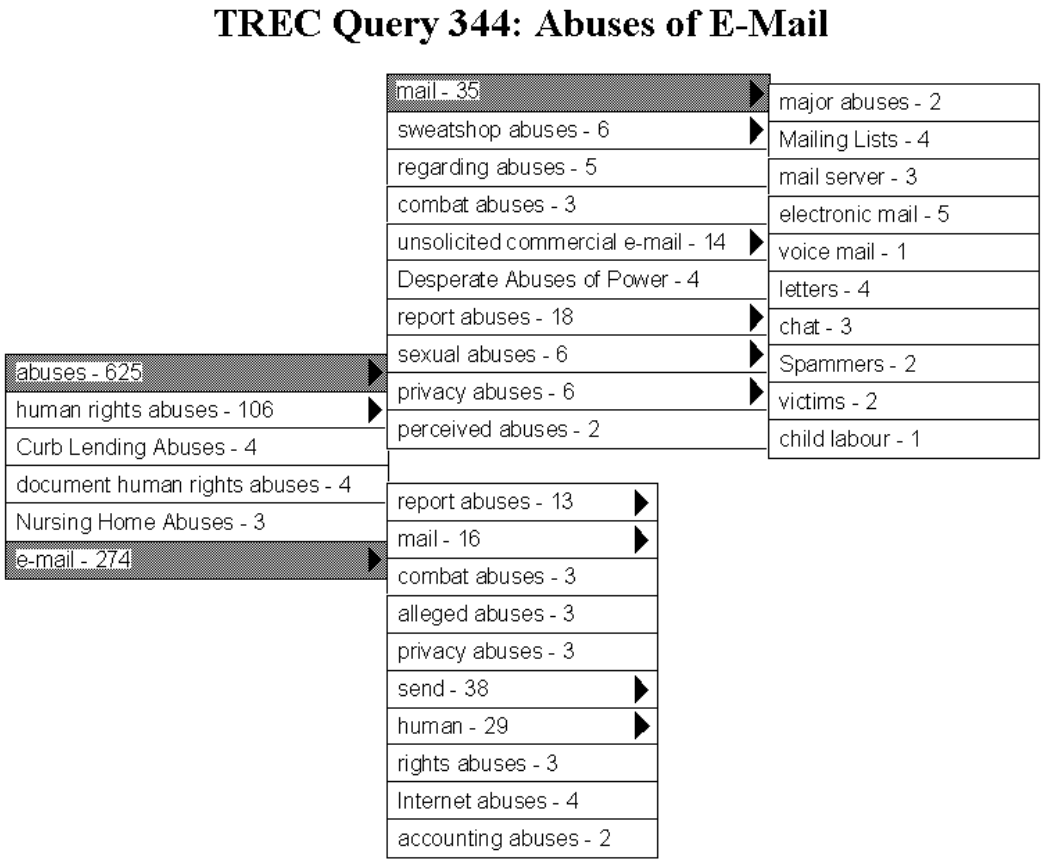
Rank 100: *FirstScience.com - The **Hubble** Decade*
... astronauts' fi rst view of the Earth from the Moon - and the **Hubble** Space Telescope's ... View from the top. On the scientifi c front, **Hubble's achievements** ...

Rank 111: *The Hindu : Discoverer of expanding universe*
... **Hubble's achievements** were recognised during his lifetime by the many honours conferred upon him In 1948 he was elected an ... 'The **Hubble** Space Telescope ...

Rank 140: *HubbleSite — Science*
... farther and sharper than any optical/ultraviolet/infrared telescope ... very specifi c goal (like the Cosmic Background Explorer), **Hubble's achievements** ...

⋮

Figure 4.4. The figure shows a portion of the hierarchy created for the TREC Topic 303: Hubble Telescope Achievements, created from snippets of documents retrieved by Google. The portion of the ranked list displayed corresponds to documents that contain the terms "achievements", "Hubble", and "Hubble's achievements". The snippets indicated that all of the documents contain relevant information about the Hubble Telescope, but one of the best sites for finding out about the Hubble Telescope's achievements is the *HubbleSite — Science* site, ranked 140^{th} by Google.



abuses→mail→electronic mail

Rank 14: *Hearing Witness: Jerry Cerasale: Spamming: TheE-Mail You Want To ...*
... signifi cant developments that are beginning to effectively combat **abuses** of **electronic mail**. These developments include the termination of service by e-mail ...

Rank 85: *RedIRIS - **Abuse** of **electronic mail** services.*
... for both e-mail users and administrators, with the purpose of increasing their awareness of some ways in which these services are being **abused**. These **abuses** ...

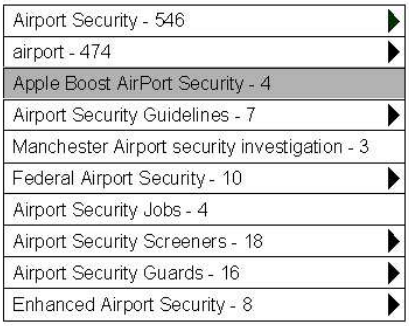
Rank 431: *NTPG : Appendix E : Acceptable Use Policies*
... District will make every effort to protect students and teachers from any misuses or **abuses** as a ... **Electronic mail** (e-mail) is not guaranteed to be private. ...

Rank 529: *http://www.eff.org/CAF/statements/ubc.ca.appendix-g-h-i*
... department name] by phone ([campus local]) or by **electronic mail** ([e-mail ... of these facilities and associated penalties, and the procedures for reporting **abuses** ...

Rank 553: *Thank You!*
... Winner will be sent their prize-winning notifi cation via **electronic mail** (e-mail ... If disqualifi ed for any of the above **abuses**, Sponsor and RealTime Media, Inc. ...

Figure 4.5. Shows a portion of the hierarchy created for TREC Topic 344: Abuses of E-mail. The portion of the ranked list displayed corresponds to documents that contain the terms "abuses", "mail", and "electronic mail". This hierarchy demonstrates how easy it is to tell when retrieval is poor. With topics about human rights, finance, and nursing home abuses, it is not surprising that relevant documents are difficult to find.

TREC Query 341: Airport Security



Apple Boost AirPort Security

Rank 250:***Apple Boost AirPort Security***, *Features*
... ISP November 13, 2001 **Apple Boost AirPort Security**, Features By Jim Wagner Software programmers at Apple (NASDAQ:AAPL) released the latest iteration of its ...

Rank 489:***Apple Boost AirPort Security***, *Features*
... Wireless November 13, 2001 **Apple Boost AirPort Security**, Features By Jim Wagner Software programmers at Apple (NASDAQ:AAPL) released the latest iteration of ...

Rank 575: ***Apple Boost AirPort Security***, *Features*
... **Apple Boost AirPort Security**, Features November 13, 2001 Software programmers at Apple (NASDAQ:AAPL) released the latest iteration of its wireless networking ...

Rank 596:***Apple Boost AirPort Security***, *Features* ... siliconvalley.internet.com November 13, 2001 **Apple Boost AirPort Security**, Features By Jim Wagner Software programmers at Apple (NASDAQ:AAPL) released the ...

Figure 4.6. The figure shows a portion of the hierarchy created for TREC Topic 341: Airport Security. The portion of the ranked list displayed corresponds to documents that contain the term "Apple Boost AirPort Security". This hierarchy demonstrates how a particular aspect of airport security (i.e. the apple networking type) is easy to find using the hierarchy.

4.4 Full Text Hierarchies

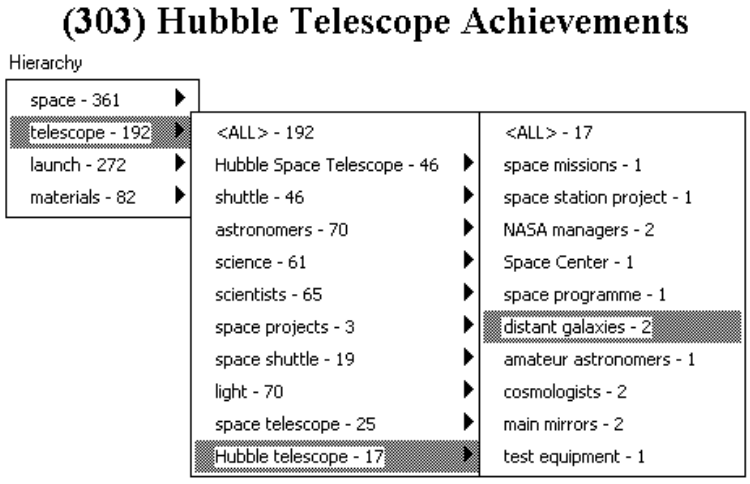
In this section we will examine the differences between examples of full text hierarchies generated from many relevant documents and those generated from few relevant documents. We will also examine the differences in the hierarchy when documents are retrieved from the complete TREC collection versus the news collection.

One of the examples we will use are the full text hierarchies created from TREC Topic 303 "Hubble Telescope Achievements". This topic was also discussed in Section 4.3.1, where we showed how informative the snippet hierarchy was. The full text hierarchies do not reveal very much information about the achievements of the Hubble Telescope. This is because the documents in the complete and news databases are much older than those found on the web. This is significant because the Hubble telescope celebrated its 10th anniversary in 2002, and this celebration is discussed in many of the retrieved web articles, along with a review of the telescope's achievements to date. Yet the other databases lack coverage of this event, since it occurred six years after the latest articles in these databases. Thus, we will see that the snippet hierarchy summarizes many more relevant documents than the full text hierarchies, and the words appearing in the hierarchies reflect that.

A portion of the hierarchy created for the news documents appears in Figure 4.7. There are four high level topics describing the 500 documents in the retrieved set[2]. Of the four, the topic "telescope" is probably the first place a user would look for information about the achievements of the Hubble telescope. In fact, this is the best place to look for relevant documents since the topic describes nine of the ten relevant documents. Examining the subtopics would probably lead a user to explore either the "Hubble Space Telescope" or the "Hubble telescope" subtopic. In order to find all nine relevant documents one would need to explore both. Figure 4.7 reveals the subtopics of "telescope→Hubble telescope", which contains seventeen documents, five of which are relevant. The subtopic "distant

[2]Documents can appear under several headings.

galaxies" seems like a likely candidate for describing the achievements of the Hubble telescope, and both of the documents have been judged relevant. Other relevant documents can be found under the subtopic "amateur astronomers", "cosmologists", and "main mirrors". Unfortunately, these subtopics only describe four of the five relevant documents in "telescope→Hubble telescope"; the fifth relevant document does not fall under any of the subtopics. In general, the subtopics in this hierarchy are not as informative as the web topics in Figure 4.4, since none of the topics include the word "achievements" or a similar word wihch would indicate that the document is about a Hubble Telescope achievement. Although it may be difficult to recognize these subtopics as descriptions of relevant information, the subtopics that describe non-relevant documents are clearly about other aspects of the space program, and can be identified as non-relevant.



telescope→Hubble telescope→distant galaxies

Rank 28: *FT 31 DEC 94 / Crunch for Big Bang: Clive Cookson explains why cosmologists are stunned into silence*

Rank 37: *FT 29 DEC 92 / Mystery of missing mass: This year's models of the universe*

Figure 4.7. The figure shows a portion of the hierarchy created for TREC Topic 303: Hubble Telescope Achievements of the documents retrieved from the news database. Both of the documents in the category "telescope→Hubble telescope→distant galaxies" have been judged relevant to this query.

If we explored the subtopics of "Hubble Space Telescope" instead of "Hubble telescope", we would find that only six of its forty-six documents are relevant, and only one of the subtopics, "Space Administration", contains relevant documents. It is unintuitive that this subtopic would have information pertaining to the achievements of the Hubble telescope. This reveals that because the goal of the hierarchy is to summarize all the documents, it may be difficult to locate relevant information. In this case, we can observe how non-relevant information can overwhelm the relevant information, and so none of the "Hubble Space Telescope" subtopics are particularly informative about the query topic.

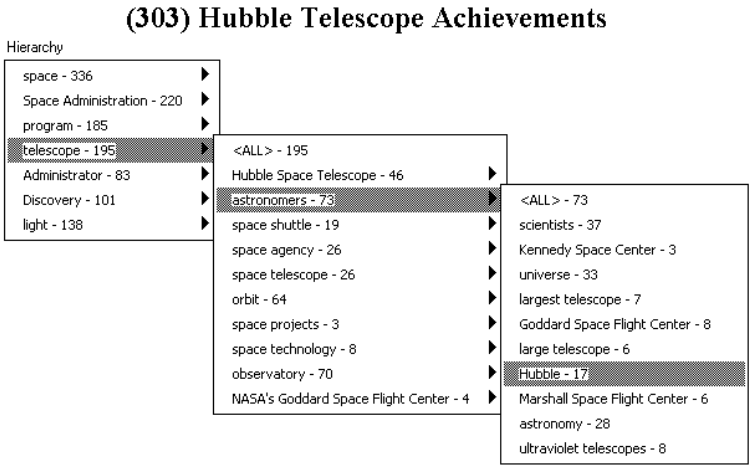
If we consider the amount of relevant information in this hierarchy, only 2 percent of the documents retrieved are relevant. It is not surprising that some of the highest level topics have little to do with the telescope. Of all the high level topics in Figure 4.7, one would likely decide that "materials" is the most vague as a topic word. However, the associated subtopics like "materials→space shuttle" and "materials→research" enable one to refine one's understanding of "materials", as the first subtopic may refer to supplies the space shuttle needs or carries, and the second may refer to supplies used in research. After examining some of the documents listed under these topics, the author believes it is unlikely that a human would have chosen "materials" as the high level topic for these documents. The subtopics, however, are much better summary words for the documents, and so considering the topic and subtopic jointly does give an indication of the kind of information one can expect to find. Of course, it should be noted that none of the documents relevant to the query can be found in the "materials" category.

For comparison, we also consider the hierarchy created for the documents retrieved from the complete database for the same TREC topic, "Hubble Telescope Achievements". This group of documents contains 400 of the same documents that are in the news hierarchy from the previous example, which means 100 of the documents in this hierarchy are different. The relevant documents in this set are the same 10 relevant documents from the previous example. Figure 4.8 shows a portion of the hierarchy for these documents.

One of the most notable differences is that there are twice as many high level topics as there are for the news hierarchy. This is not surprising since the 100 new documents are considerably longer than the news documents, which means this hierarchy summarizes more information than the news hierarchy. The complete hierarchy in Figure 4.8 includes five of the same subtopics of "telescope" as the news hierarchy. Because "Hubble telescope" is not among these subtopics, none of the subtopics groups the relevant information as well as it was grouped in the news hierarchy. In this case, the subtopic "astronomers" is the best one for finding relevant documents. It contains seven relevant documents, while the next best subtopic, "Hubble Space Telescope", contains only six. The subtopics of "astronomers" also describe the relevant documents better than those of "Hubble Space Telescope". The highest percentage of relevant documents is in the category "telescope→astronomers→Hubble", where five of the seventeen documents are relevant.

Overall, it is very difficult to discern any major differences between the two document sets from studying their hierarchies. Although some of the details of the hierarchies differ, there is no word or group of words that are present in one hierarchy that would seem out of place in the other. For instance, "orbit" is a subtopic of "telescope" in the complete hierarchy shown in Figure 4.8 but does not appear in the portion of the news hierarchy revealed in Figure 4.7. However, "orbit" is a subtopic of "materials" and appears as a sub-subtopic in the categories "space" and "launch" in the news hierarchy. The only difference is that "orbit" is not as good as the words that are part of the hierarchy at describing the subtopics of "telescope".

Now we offer a comparison of this example of a hierarchy with few relevant documents to one with many relevant documents. To illustrate a document set with many relevant documents, we chose TREC Topic 319: "New Fuel Sources". In the case of the news document set, 111 of the 500 documents retrieved are relevant. A portion of this hierarchy appears in Figure 4.9. Like the news hierarchy for Topic 303, there are very few high level topics. In this case though, both of the topics present are related to the TREC topic. Of



Rank 1 *FT 25 FEB 92 / Technology: Great eye sets sights sky high – A European project to build the world's most powerful telescope*

Rank 2 *POWERFUL TELESCOPE SOARS INTO SPACE ABOARD SHUTTLE*

Rank 3 *NASA ADVANCES LAUNCH DATE FOR SHUTTLE AND SPACE TELESCOPE*

Rank 5 *POWER UNIT PUT IN SHUTTLE;TELESCOPE BATTERIES CHARGED*

Rank 6 *FT 30 NOV 93 / Toil and trouble: Nasa needs to repair its public image, as well as the Hubble telescope*

Rank 8 *ASTRO'S 4 TELESCOPES MAY FILL IN GAPS ABOUT SPACE*

Rank 9 *NASA SCRUBS LAUNCH OF DISCOVERY; SHUTTLE FLIGHT: PROBLEM IN AUXILLIARY POWER UNIT PUTS HUBBLE TELESCOPE FLIGHT ON HOLD FOR ONE OR TWO WEEKS.*

Rank 11 *OBSERVATORY ATOP MT. WILSON AIMS TO REJOIN SCIENCE*

Rank 23 *TOUCHY TELESCOPE TORMENTS CONTROLLERS; SPACE: SCIENTISTS ENCOUNTER 'MORE PROBLEMS THAN WE ANTICIPATED,' BUT THEY ARE STILL CONFIDENT.*

Rank 24 *FT 31 DEC 94 / Crunch for Big Bang: Clive Cookson explains why cosmologists are stunned into silence*

Rank 25 *FT 14 DEC 93 / 'Flawless' mission over*

Rank 30 *TELESCOPE LAUNCHED; FREED SOLAR PANEL AVERTS SPACE WALK*

Rank 37 *FT 29 DEC 92 / Mystery of missing mass: This year's models of the universe*

Rank 39 *WAR RAGES OVER RED SQUIRREL HOME; ENVIRONMENT: A PLAN TO BUILD TELESCOPES ON THE RODENT'S MOUNTAIN HAS INVOLVED SPECIAL LEGISLATION, PLEAS FROM THE POPE AND ECO-SABOTABE.*

Rank 49 *IN BREIF: SCIENCE/MEDICINE; HUBBLE TO VIEW HUGE SATURN STORM*

Rank 67 *FT 01 Jan 94 / Big science seeks Wimp*

Rank 81 *SOUNDS OF SILENCE; OUTER SPACE: A SMALL GROUP OF SCIENTISTS IS TURNING A TECHNOLOGICAL EAR HEAVEWARD IN A SEARCH FOR OTHER INTELLIGENT LIFE IN THE UNIVERSE.*

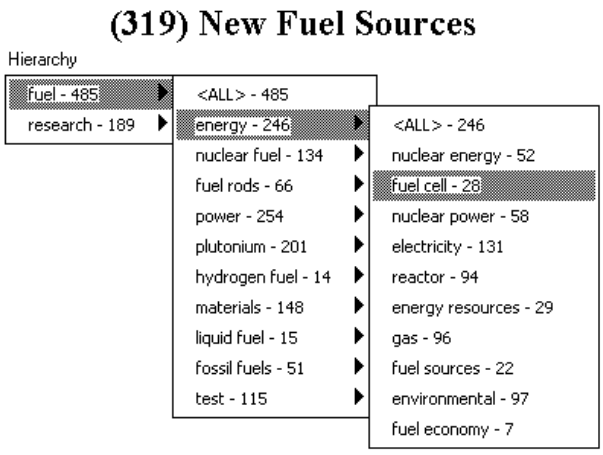
Figure 4.8. The figure shows a portion of the hierarchy created for the TREC Topic 303: Hubble Telescope Achievements for the documents retrieved from the complete database. The documents at rank 1, 24, 37, 39, and 67 in the category "telescope→astronomers→Hubble" have been judged relevant to this query.

the 485 documents under the topic "fuel", 109 of them are relevant, and 65 of the 189 documents under the topic "research" are relevant. Figure 4.9 reveals the subtopics of "fuel". All of the subtopics except "liquid fuel" contain some relevant documents, and approximately one-third of the documents are relevant in most of the subtopics. Figure 4.9 also presents the subtopics of "energy", which could be considered a synonym for fuel. Of the 246 documents about "fuel→energy", 72 of them are relevant. Again, many of the subtopics cover currently available fuel sources, but in fact every sub-subtopic contains at least one relevant document and most contain over 20 relevant documents. The documents related to the topic "fuel→energy→fuel cell" appear in Figure 4.9 with their ranks, of which 18 are relevant.

When judging the hierarchy as an effective tool for presenting useful data, this one contains more information about the TREC topic than the complete hierarchy for the Hubble topic presented in Figure 4.7 does about its topic, and would likely be more useful to a user. However, when considering the hierarchies outside the context of the query, both express an equally accurate description of their respective document sets. The fuel sources hierarchy was generated from a document set that contained more relevant information than did the Hubble telescope's document set, and is therefore more informative with respect to the TREC topic than the Hubble telescope hierarchy.

4.5 Conclusion

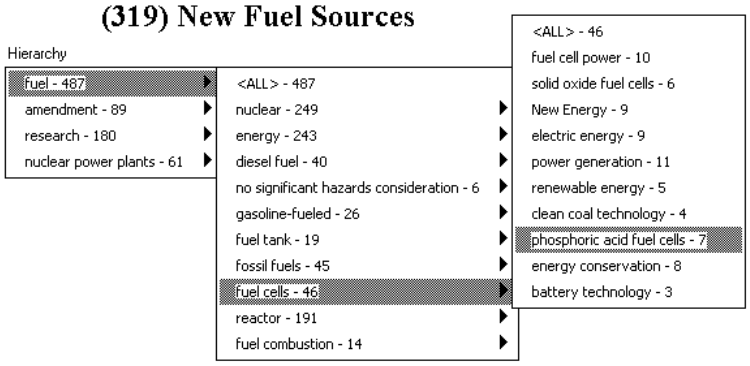
From these examples, it is evident that using snippets of documents focuses the hierarchy on relevant information since the snippets are determined based on the existence of query words in the passage. In a sense, this type of hierarchy can be viewed as a summary of a summary. However, even though the full text hierarchy is not as focused on the query topic, one can still determine how effective the retrieval was based on how closely the topic words are related to the query. A disadvantage present in both snippet and full



fuel→energy→fuel cells

Rank 20: *No Title*
Rank 23: *No Title*
Rank 35: *No Title*
Rank 58: *FT 05 AUG 92 / Business and the Environment: Fuel cells' change to take charge*
Rank 76: *No Title*
Rank 99: *No Title*
Rank 131: *No Title*
Rank 133: *FT 15 APR 94 / Technology: Milestone on the road – Fuel cells are set to become the power source of the future*
Rank 139: *No Title*
Rank 142: *No Title*
Rank 143: *No Title*
Rank 149: *SPEAKING OUT: AQMD RULES TO AFFECT WHAT PROJECTS BUILT; DEVELOPMENT: NEW AIR QUALITY MANAGEMENT POLICY WILL INFLUENCE ALL MAJOR PROJECTS, INCLUDING OFFICE BUILDINGS, SHOPPING CENTERS AND INDUSTRIAL PARKS.*
Rank 155: *No Title*
Rank 181: *FT 17 MAY 94 / Survey of Power Generation Equipment (12): It started in London back in 1839 – The persistent dream of winning cheap electricity from chemical fuel cells*
Rank 188: *No Title*
Rank 195: *FT 06 APR 94 / Private cash targeted in energy plan*
Rank 204: *No Title*
Rank 210: *No Title*
Rank 228: *No Title*
Rank 235: *No Title*
Rank 239: *No Title*
Rank 285: *No Title*
Rank 301: *FT 15 APR 94 / Technology: Getting there by fuel cells*
Rank 322: *No Title*
Rank 339: *No Title*
Rank 351: *No Title*
Rank 436: *No Title*
Rank 437: *FT 01 DEC 92 / Letter: Electric-powered transport potentially cleanest*

Figure 4.9. The figure shows a portion of the hierarchy created for the TREC Topic 319: New Fuel Sources, for the documents retrieved from the news database. Eighteen of the documents shown have been judged relevant, including those at rank 20, 23, 35, 76, 99, 131, 133, 139, 142, 143, 155, 188, 195, 204, 210, 235, 239, and 322.



fuel→fuel cells→phosphoric acid fuel cells
Rank 149: *No Title*
Rank 289: *No Title*
Rank 306: *No Title*
Rank 324: *No Title*
Rank 343: *No Title*
Rank 360: *No Title*
Rank 441: *No Title*

Figure 4.10. The figure shows a portion of the hierarchy created for the TREC Topic 319: New Fuel Sources for the documents retrieved from the complete database. Six of the documents shown have been judged relevant including those at rank 149, 289, 306, 324, 360, and 441.

text hierarchies is that they include uninformative topics. As the hierarchies become further developed, we will look for techniques to identify these uninformative topics.

CHAPTER 5

IMPLEMENTATION AND EFFICIENCY

This chapter discusses the software that we created for the generation and display of hierarchical word-based summaries. In Section 5.1 we discuss the implementation of the hierarchy generation software. This includes a discussion of the types of data structures that were used to increase efficiency. In Section 5.2 we analyze the efficiency of the software and present the actual running time for different test sets. In Section 5.3 we discuss the web-based user interface for interacting with the hierarchies. We also include a discussion of the different parameters users can set when generating the hierarchy. We conclude with a discussion in Section 5.4.

5.1 Hierarchy Software Design

The general implementation for generating hierarchical word-based summaries appears in Figure 5.1. We begin with the full text of the documents. This is used to find all phrases and to compute the first bigram model. Once the phrases are defined, we compute a unigram model for the document set and compare it to a general model of English. We use a large pre-built database to quickly compute a general English model. From these initial stages, we decide on the vocabulary that will be part of the bigram model. While the bigram model is computed, we create a representation of the text in the documents so that future models can be built more efficiently. After producing the topical model, predictive model, and storage of the document set, the initialization stage is complete. We then begin a recursive stage to select the topics that will be part of the summary and to recompute the bigram model for each new level in the hierarchy.

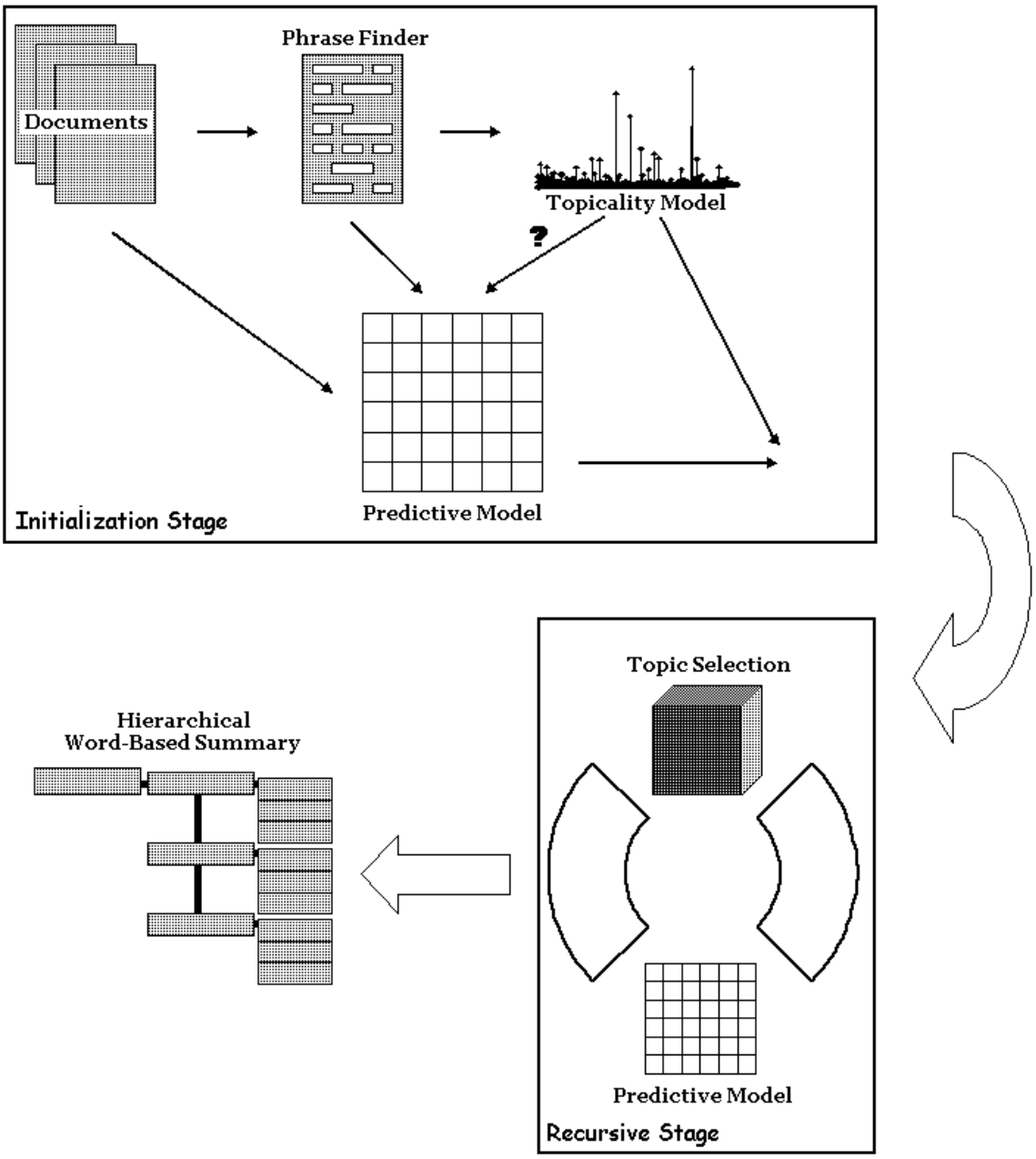


Figure 5.1. Shows the software implementation for creating hierarchical word-based summaries. In the initialization stage, we begin with the document set. We use the text to find phrases, and then compute the topicality model. The text, phrases, and topicality model are used to build the predictive model. The two models are used in the recursive stage to select topic words. For each level in the hierarchy, the predictive model is recomputed and then topic words are selected. After all the words are selected, the full hierarchy is assembled and outputted to a file.

From an efficiency point-of-view, we are mostly concerned with being able to quickly generate the bigram model and choose the topic words, since these two steps are repeated for each individual level of the hierarchy. The algorithm spends on average 74.5% in the recursive stage when it is given the full text of the documents. However, when it is creating the hierarchy from snippets, it spends only about 43.1% of the time in the recursive stage. This is because the co-occurrence statistics are not computed during the recursive stage. Since the algorithm generally spends a smaller portion of its time in the initialization stage when creating hierarchies using the full text of the documents, we decided that currently available solutions would be used in the initialization stage for finding the phrases and computing the KL divergence scores rather than developing our own solutions for these tasks. While we used currently available solutions for the initialization stage, we created our own solution for the recursive stage in order to make it efficient and language independent. Language independence was important so that it would be easier to create hierarchies in different languages.

We now discuss each stage in greater detail. In Section 5.1.1 we describe the tools that we use for individual tasks and the means we use for storing the documents. In Section 5.1.2, we describe how the **DSPapprox** algorithm is used in conjunction with the representation of the models, and how the bigram model is reconstructed.

5.1.1 Initialization Stage

The Initialization Stage consists of inputting the documents, finding the phrases, building the models, and storing the document text so that the predictive models can be efficiently generated during the Recursive Stage.

First we need access to the text of the documents, for which we have two methods. If the documents are stored in an InQuery database, we can access the title and text of each document through the database. If such a database is not available, the program will read directly from ASCII text files in a simple SGML format. We specifically recog-

nize the labels: “<DOC>”, “</DOC>”, “<DOCNO>”, “</DOCNO>”, “<TITLE>”, “</TITLE>”, “<TEXT>”, and “</TEXT>”. The parsing for such files is minimal, and has been designed specifically to read the documents generated from Google snippets for the web hierarchical summaries. If the documents were in a general flat-file format, this portion of the software would need to be improved.

The next step is to find the phrases. During this stage we make initial counts for the occurrence of individual words. To do this, we use a hidden Markov-model approach designed by Feng and Croft[15]. This software locates phrases that are up to six words in length; it has been shown that the quality of the phrases discovered is better than other automatic techniques for finding phrases. However, since the actual method of extracting phrases is unimportant, other techniques could be substituted for the one we are currently using.

After the phrase-extraction stage, we create a list of all single words and phrases present in the document set that meet the specified criteria. These criteria are:

- Occur in at least 1 percent of the documents
- Must be at least 3 characters in length
- Cannot be a stopword
- Cannot be a number

As a first step we examine the frequency of each word in the entire document set. We calculate the number of documents that make up 1 percent of the document set. Any word whose frequency is less than the number of documents is eliminated because we know that these words cannot possibly satisfy the first criterion.

Although numbers and stopwords on their own are not valid topics, these words can be part of a phrase. Examples of such phrases include “Department of Computer Science” and “11 percent”. We believe that when numbers are part of a phrase, they may make sense to a user whereas a number will most likely be meaningless on its own.

With this list of acceptable single words and phrases, we calculate the topicality model. We will refer to this step as **TModel** in later sections. The topicality model is comprised of the KL divergence contributions between general English and the document set. To find the probability of a word occurring in general English, we access the word count in a local implementation of an inverted index. The word counts stored in this database allow the efficient calculation of the query-biased KL divergence. Unfortunately, the local implementation only has statistics for single words. So after calculating the KL divergence contribution of such words, we estimate the KL divergence contribution of phrases, by adding the contributions of the individual words that occur in the phrases. We do this because to do not have data on the occurrence of phrases. This calculation overestimates the probability of phrases, which is acceptable because we believe phrases are better able to inform the user of the topic.

Before the predictive model can be calculated, two things must occur. The list of acceptable words is refined further, and the dictionary of phrases is built.

The acceptable words are those that meet the criteria laid out above, specifically that they are frequent enough, contain at least 3 letters, and are not stopwords or numbers. One of our parameters imposes a requirement that the words contribute positively to KL divergence. This criterion improves the running time of the recursive step by 15-30% on average, because fewer words will be considered topics. We test the impact of this criterion in Section 6.2.1.

Next, we build a phrase dictionary from the acceptable phrases. We store the dictionary in a recursive structure called a trie that looks like Figure 5.2. A single entry in the dictionary is pictured in Figure 5.3. Each head word for a phrase is stored in the top level hash table. The data for the possible completions of the phrase are stored at the identification number provided by the hash table. The "phraseEnd" variable indicates whether or not the traversal up to the particular point is a valid phrase. This dictionary helps us quickly identify all occurrences of phrases when we are parsing the text in order to store

the document. Our phrase software is not used to find all phrases and sub-phrases, because it only identifies the longest possible phrases. This structure allows us index sub-phrases and single words in a phrase when it is desirable.

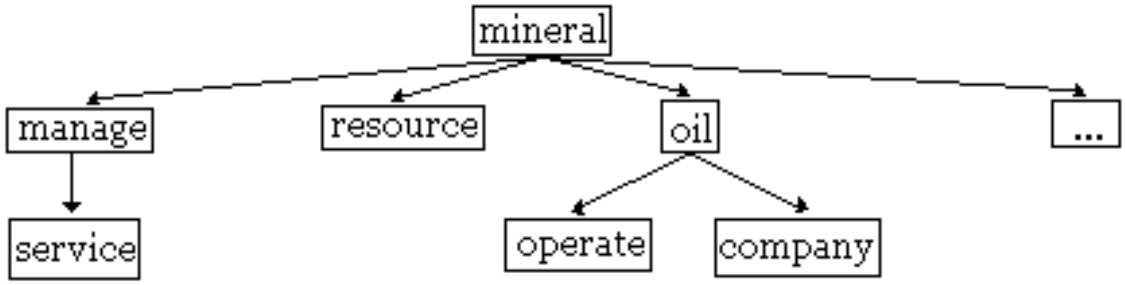


Figure 5.2. Illustrates what the phrase dictionary looks like. The phrases "mineral manage", "mineral manage service", "mineral resource", "mineral oil", "mineral oil operate", and "mineral oil company" are shown.

dictionary	(hashTable pointer)
nextWordList	(phraseDictionary array pointer)
listArraySize	(short)
curNumWordList	(short)
phraseEnd	(boolean)

Figure 5.3. Shows the information stored when creating the phrase dictionary. The hash table, "dictionary", allows the ability to quickly determine if a word is part of a phrase. The word list array, "nextWordList", contains all the words that follow the particular word in a phrase. The end bit, "phraseEnd", stores whether this word is a valid end to the phrase or not. The other two variables are used when creating the dictionary to keep track of sizes.

We have found that some phrases convey information aside from the meaning of their component words. Such phrases are generally referred to as *named entities*. An example is the phrase "United States" which means something different from "united states". The capitalization is significant, since "United States" refers to a specific country, whereas "united states" represents a more general case. Since we decide which words are topics and subtopics based on co-occurrence information, it is important to determine if the sub-parts

of a phrase have meaning within the context of the text. If they do have meaning, such as in "united states" where the text would be talking about "states", we would want the individual words "united" and "states" to be present in the representation of the document. However, just because "United States" appears in the text, does not necessarily mean the document is referring to the topic of "states". If the document is unconcerned with "states", we do not want that word in our representation of the document. This example also reveals the fact that a phrase can sometimes be a named entity but is not always one, which means a simple list of named entities that should be recognized would not provide the correct information. To deal with this, we consider each case individually. If all the first letters in the phrase, except perhaps stopwords, are capitalized, we consider the word a named entity and will not break the phrase up into its parts. If it is not all capitalized, then the individual words and sub-phrases will be indexed and used when calculating the predictive model.

After the list of acceptable words is compiled and the phrase dictionary is built, we construct the predictive model. We will refer to this step as $\mathbf{PModel-I}$. An index of the documents is built simultaneously. The index resembles Figure 5.4. Each node in the index contains the information appearing in Figure 5.5, which includes an identification number for the word, a document identification number, and a position number. Only valid words and phrases are stored in the index.

In order to create the model and index simultaneously, we add words to the permanent storage structure and to an array that is used to calculate co-occurrence information. First, it must be determined if the current word in the text is a valid beginning of a phrase. If it is, we determine if it is head word for a phrase. Any phrases that are found are assigned the location of the first word. We then decide if the phrase is a named entity. If the phrase is not a named entity, the first word is tested to see if it meets the standard criteria for a single word. If it does, it is stored in the index and entered into the array that represents the current text segment. If no phrases are present, the word is also tested to ensure it meets the standard criteria before being entered into the array. If it does not meet the criteria, a

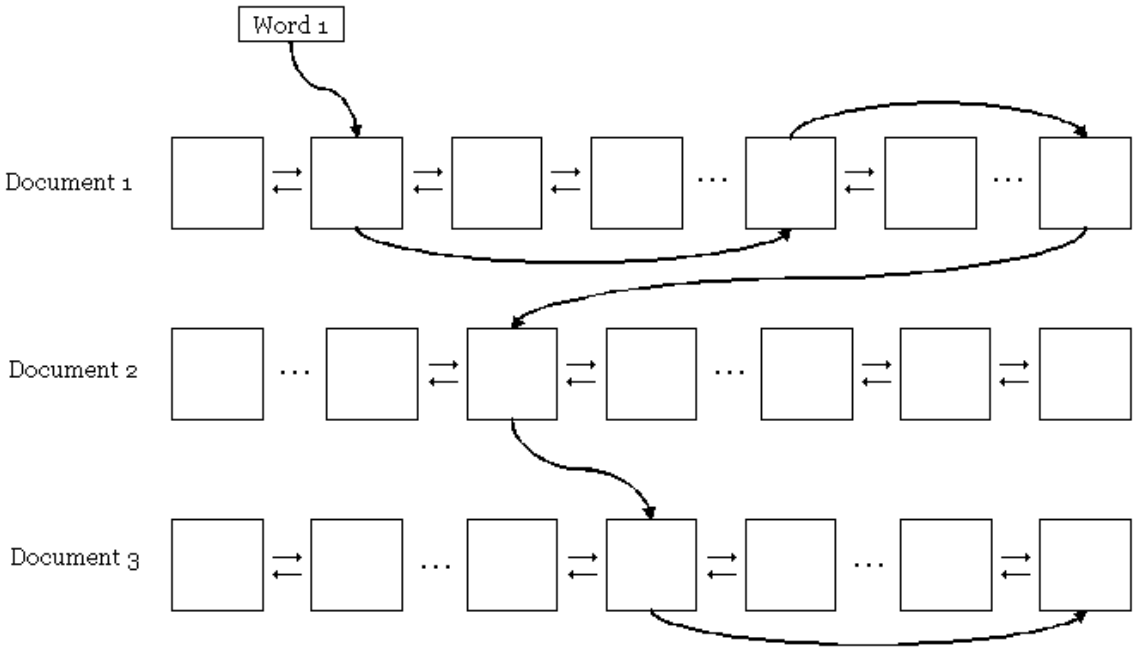


Figure 5.4. Illustrates the index for the document text. This index is a linked structure. Each word is a node in the index. Phrases are stored before the occurrence of the head word. The words are doubly linked to facilitate backward and forward traversals. In addition a linked list of all the occurrences of a specific word is created, which is used when building predictive models in the recursive step.

placeholder is used in the array. When all the words preceding and succeeding a particular word are loaded into the array, the array is used to calculate co-occurrence information for the predictive model.

The array size is determined by a parameter set when generating the hierarchy. There are two ways in which this can be specified. One is by using a fixed length, which we later refer to as a fixed segment size. The other way is to specify the full document.

If the text segment is the full document, the co-occurrence calculation is much simpler. Individual occurrences of a word are no longer important. All that is required is a list of words that occur in the document. Each word co-occurs with all the other words. This information is then stored because the co-occurrence information for a document will not change during the recursive step. Since text segments are fixed and a large number of words

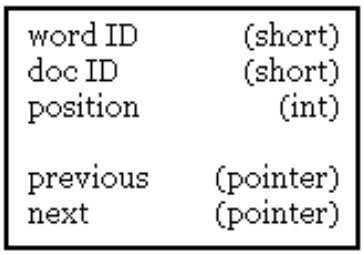


Figure 5.5. Shows the data stored for each word in the text of the documents. A word identification number is assigned to each unique word in the vocabulary, called the word ID. Each document is also assigned an identification number, and the doc ID specifies the document in which the word occurred. The position is a count of the number of words that precedes this particular word in the document. Along with this information are pointers to the previous and next words. Because of the position information, only those words that are part of the predictive model need to be stored.

share the same co-occurrences throughout the construction of the summary, the running time of the software is greatly improved.

5.1.2 Recursive Stage

The recursive stage has two steps. The first step selects the topics for a given level of the hierarchy using the topicality and predictiveness models that were built in the initialization stage. This uses the **DSPapprox** algorithm described in Section 3.6. The second step reconstructs the predictiveness model and is discussed in greater detail in this section.

To build the predictiveness model, we begin with the parent topic word(s) for the hierarchy level under consideration. If there is only one parent topic word, the linked list of topic word's occurrences is used to identify the text segments that will be used to calculate the model. We use these occurrences to find the valid text segments within the documents and calculate the predictive model. If there is more than one parent topic word, the linked lists of occurrences are traversed simultaneously to find segments of text where all parent topic words occur together. These segments will be the ones used to estimate the predictive model. In future discussion we refer to this calculation as the **PModel**.

5.2 Software Efficiency Analysis

In this section we discuss the efficiency of each stage and step in the software. In Section 5.2.1 we analyze the running time algorithmically in terms of big-O notation. After that we present actual data for how long each step takes in Section 5.2.2. For this analysis we break the process up into three parts: locating phrases in Section 5.2.2.1, initializing the models in Section 5.2.2.2, and the recursive stage in Section 5.2.2.3.

5.2.1 Algorithmic Analysis

This section analyzes the running time of the software in terms of the input parameters. These parameters include:

- l, the number of words that comprise the text of the documents
- n, the number of unique words in the text
- q, the number of words in the query
- c, the number of sub-collections from which the documents are retrieved
- d, the number of documents
- w, the size of the text segments used to calculate the predictiveness model
- h, the depth of the hierarchy
- t, the number of candidate topics in the $\mathbf{DSPapprox}$ algorithm
- s, the number of sub-topics in the $\mathbf{DSPapprox}$ algorithm
- o, the number of occurrences of the most frequent candidate word

The parameters are used in the Big-O notation to describe the algorithms.

The initialization stage is subdivided into two parts. The phrase finding step takes $O(l)$ time, since all of the text must be scanned. The second part is building the models and indexing the text which takes $O(\mathbf{TModel} + \mathbf{PModel} - \mathbf{I})$.

There are three variants of $\mathbf{TModel}$, calculating the topicality model. If the topicality model is estimated without a query bias, it takes $O(n)$ time to run. If we calculate the topicality model using sub-collections, the algorithm takes $O(nc)$ time to run. If a query bias is used. The running time is $O(dnq)$.

There are two variations for $\mathbf{PModel - I}$, the predictiveness model and indexing. To build both the predictive model and the index with a constant text segment size takes $O(lw)$. If the full text of each document is used for the text segments, building the model and indexing takes only $O(l)$ time.

The recursive stage is analyzed as a single unit. This stage runs in $O(\mathbf{DSPapprox} + \sum_{i=1}^{h-1} k^i(\mathbf{DSPapprox} + \mathbf{PModel}))$. The running time of $\mathbf{DSPapprox}$ is $O(kts)$. The running time of $\mathbf{PModel}$ is $O(iow)$.

5.2.2 Real Time Analysis

In this section we analyze the running time data collected during the creation of the hierarchies which will be analyzed in Chapter 6. The data presented is drawn from the generation of over 2000 hierarchies for each of nine collections of documents. The collections are described in Section 4.1. The experiments were run on a Sunfire 6800. The machine was consistently balancing over 100 processes at a time, which most likely increased the amount of time spent on some experiments.

This section is divided into three parts. In Section 5.2.2.1 we discuss the running time of the phrase finding part of the algorithm for the collections drawn from the original database and the all news database. In Section 5.2.2.2 we discuss the running time of the initialization of the models. In the case of the Google collection, we include the phrase finding step in our measure of the time, but we do not ro the other two types of collections. In Section 5.2.2.3 we discuss the running time of the recursive stage of the algorithm.

5.2.2.1 Phrase Finding

In this section we examine the time required to gather the phrases for the hierarchy. Table 5.1 contains information about the average time required for each database, and the number of phrases that are likely to be found. Given that relative to the other steps finding the phrases is very efficient, there is not a great necessity to develop a faster algorithm for completing this step. However, only considering the time to find the phrases for hierarchies created from documents in the complete database makes creating such hierarchies in an interactive environment unrealistic. We do propose creating snippet hierarchies in an interactive environment. Since it takes less than a second to find all the phrases in this document, the benefit gained from including phrases in the hierarchy does not come at too dear a price.

Collection	*Avg. Phrase Time*	*Avg.Phrases*
201-250 Complete	23.478	4210.6
251-300 Complete	56.369	11246.1
301-350 Complete	32.855	6557.0
201-250 News	10.057	876.0
251-300 News	10.511	782.9
301-350 News	16.662	1965.04
201 -250 Web	0.912	360.7
251-300 Web	0.925	379.68
301-350 Web	0.949	376.8

Table 5.1. Statistics for the phrase time. Time is reported in seconds.

5.2.2.2 Initialize Models

During the initialization of the models, the full text of the documents is scanned and the structure in Figure 5.4 is created. Also during this stage the predictive and topical models are constructed. Statistics on the running time appear in Table 5.2. One of the most notable features in these statistics is the extent to which the length of the document contributes to the total running time. With each change in the collection, the average document size

changes significantly which greatly effects the running time. The hierarchies created using Web snippets take only a few seconds to find all the phrases and create the models. The hierarchies generated from news text take one-to-two minutes to create the models, and the hierarchies generated from news and other text from the original TREC databases take four to eight minutes each.

	Initialization Time				*Init. Time per Word*	
Collection	*Mean*	*Median*	*Varience*	*Stan. Dev.*	*Mean*	*Avg.Words*
201-250 Complete	216.82	159.64	49039	221.45	0.0300	6758
251-300 Complete	444.99	303.37	204360	452.06	0.0352	11,840
301-350 Complete	317.28	197.10	116413	341.19	0.0329	8789
201-250 News	62.16	54.37	958.84	30.97	0.0199	3156
251-300 News	62.62	53.11	1051	32.42	0.0182	3460
301-350 News	143.70	119.43	9544	97.69	0.0258	5435
201 -250 Web	2.61	2.61	0.35	0.56	0.0021	1243
251-300 Web	2.70	2.72	0.53	0.73	0.0021	1274
301-350 Web	2.73	2.70	0.58	0.76	0.0021	1302

Table 5.2. Statistics for the initialization time. Time is reported in seconds. The time for the web collections includes the phrase-finding stage, although the others do not.

Another attribute to notice is that the mean is always greater than the median when full document hierarchies are generated. This is due to the fact that some individual hierarchies take an excessive amount of time to create. These outliers are illustrated in Figure 5.6, which is a box plot for each of the collections.

Finally, in Table 5.2 the running time is analyzed in terms of the number of unique words considered when generating the hierarchy. This data reveals the difference in calculating the co-occurrence data using a distinct segment size versus using a fixed segment. The time it takes per word to generate the hierarchy with a fixed segment, as is done when creating the snippet hierarchies, is an order of magnitude faster than the distinct segment size, which is used in the original and news hierarchies. The reason that there is a significant difference between the time of the original and news hierarchies is because of the difference in the lengths of the documents

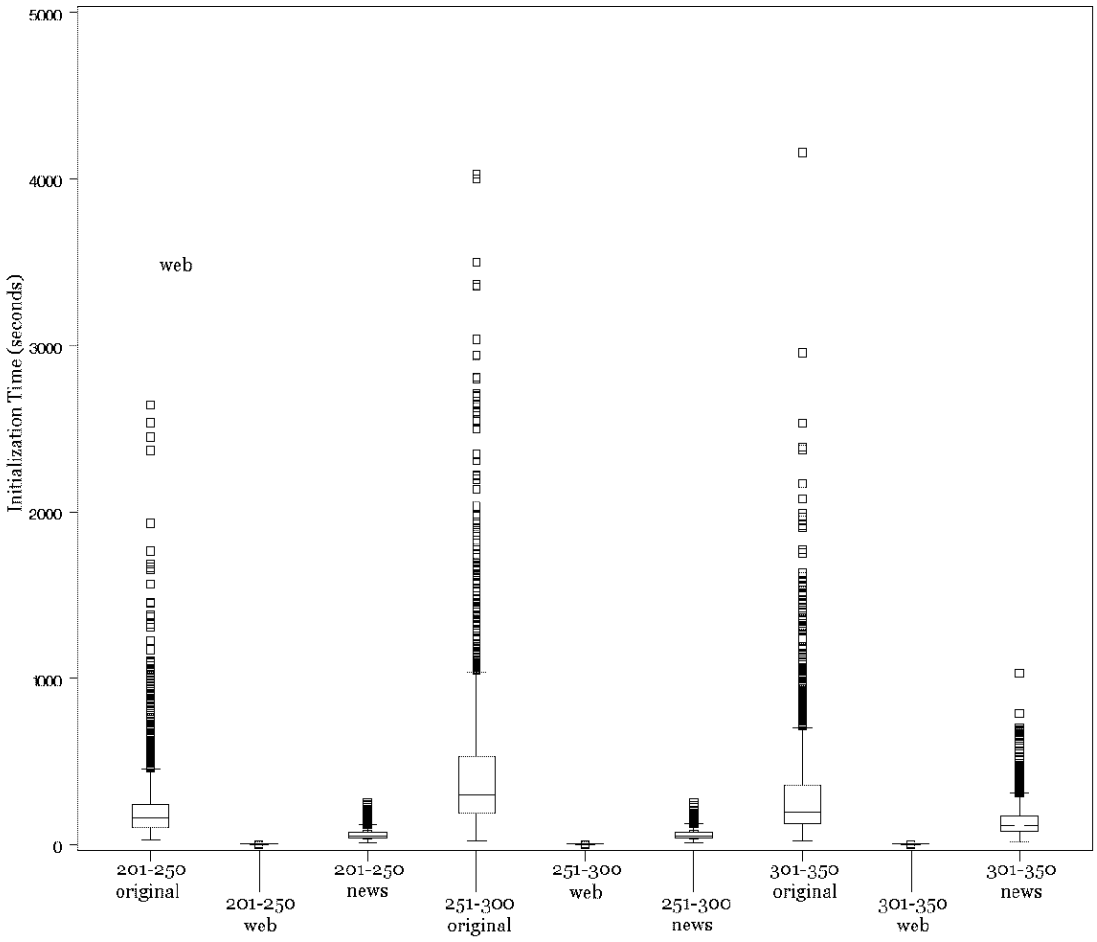


Figure 5.6. This is a box plot of the initialization times for each of the collections. Notice that the original databases had a much greater range of computation time, and that some outliers required more than 4000 seconds for computation. This was never observed in the other datasets. The variability of the web collections is tiny in comparison.

5.2.2.3 Recursive Stage

The recursive stage selects the words for each level of the hierarchy, and in the case of the distinct segment size, the predictive model is recalculated. Table 5.3 lists some statistics for the number of seconds this stage takes for the different collections. Again there is a notable difference in running times for the different types of collections: original, news, and web snippets. On average, hierarchies created from web snippets take about 20 seconds to select the terms. The hierarchies created from news text take about four to ten minutes to select the terms, and the hierarchies created from news and other text take about fifteen to thirty minutes to select the terms.

	Recursive Time				*Recur. Time p. Level*	
Collection	*Mean*	*Median*	*Varience*	*St. Dev.*	*Mean*	*Avg. Menus*
201-250 Complete	902.49	559.86	1308421	1144	16.0300	80
251-300 Complete	2002.52	1112.05	6618510	25023	26.9682	104
301-350 Complete	1535.50	929.97	3409397	1846	22.4426	86
201-250 News	266.56	214.07	47687	218.37	9.1178	50
251-300 News	246.89	190.43	48111	219.34	7.5131	56
301-350 News	683.32	465.82	691398	12826	15.3350	62
201 -250 Web	15.19	1.32	3206	56.63	0.0256	437
251-300 Web	19.62	1.38	9201	95.92	0.0272	468
301-350 Web	20.88	1.28	19599	140.00	0.0278	694

Table 5.3. Statistics for the recursive stage time. Time is reported in seconds.

During this stage, there is also a great discrepancy between the mean and the median for collections. Although the recursive stage for snippet hierarchies usually takes under two seconds, the mean is around twenty seconds. The running times are plotted in Figure 5.7, which illustrates the outliers with boxes. We examine the actual datasets responsible for the poor performance and find that one or two queries required an inordinate amount of time for the snippet hierarchies, while particular parameter settings are responsible for the long running times for the other two types. In the case of the snippet hierarchies, the documents retrieved for these queries produced incredibly large hierarchies. One difference between

the parameters used to create the snippet and full text hierarchies is that an infinite number of levels is used for snippet hierarchies. Rather than stopping after three levels have been generated, the snippet hierarchy continues to add levels until it runs out of words or there are fewer than five documents at a particular node. For queries 201-250, the largest snippet hierarchy includes $27,975$ menus. For queries 251-300, the largest snippet hierarchy includes $44,113$, and the largest snippet hierarchy for queries 301-350 includes $193,844$ menus. The largest possible full text hierarchy is 421.

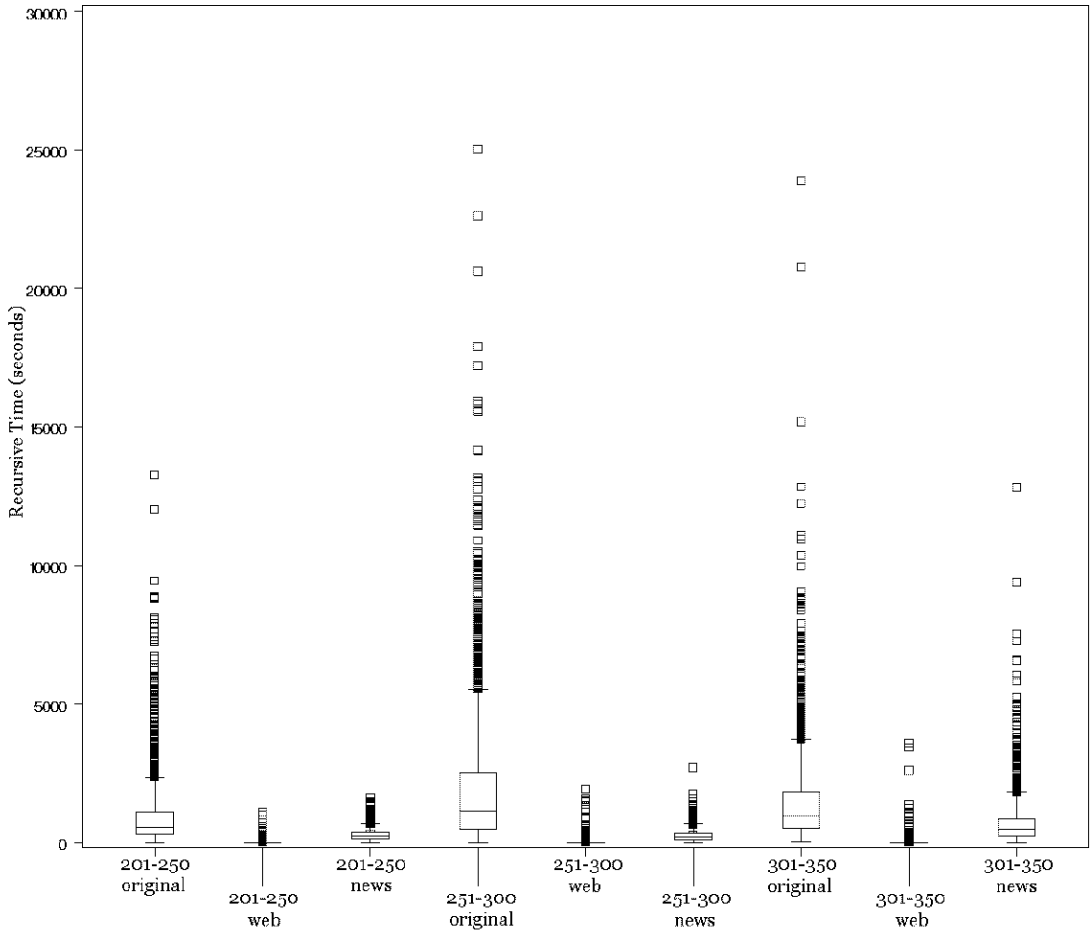


Figure 5.7. A box plot representing the running times for the recursive stage.

Table 5.3 also includes the average running time for creating a single level in the hierarchy, which is equivalent to the number of recursions performed. This data clearly

demonstrates the increased efficiency of using fixed segments rather than distinct segment sizes. Using fixed segments is close to two orders of magnitude faster. This performance increase is due to the fact that the predictive model is not regenerated for each level of the hierarchy.

5.3 Hierarchy Interface Design

The design of the interface for browsing using hierarchical summaries resembles that of a search engine portal as shown in Figure 5.8. The user is expected to enter a query in the text box and then press enter or click the search button. The documents are retrieved from the database specified in the parameter settings. In the figure, the documents are retrieved from Google[21] using APIs provided by the search engine. Once the documents are retrieved, the program that creates the hierarchy is invoked. This program creates a file containing the hierarchy. The hierarchy is displayed using the standard popup menu widgets provided by the JAVA AWT class[63]. The hierarchy can be accessed by pressing on the link labeled "hierarchy".

Besides creating the hierarchy, the interface allows the user to set up preferences. These preferences are divided into two parts, standard and advanced. Each of the preferences is be discussed in the following subsections.

5.3.1 Standard Preferences

The standard preferences consist of six different settings. The interface is pictured in Figure 5.9. These preferences are the depth of the hierarchy, the number of topics at each level, the method of constructing the topical model, the language and database for retrieval, and the number of documents to retrieve. Each of these parameters has several settings, which is enumerated below.

The hierarchy can be set to a number, one through five, or infinite using the pull-down menu labeled "Levels". The infinite setting places no requirement on the depth, but instead

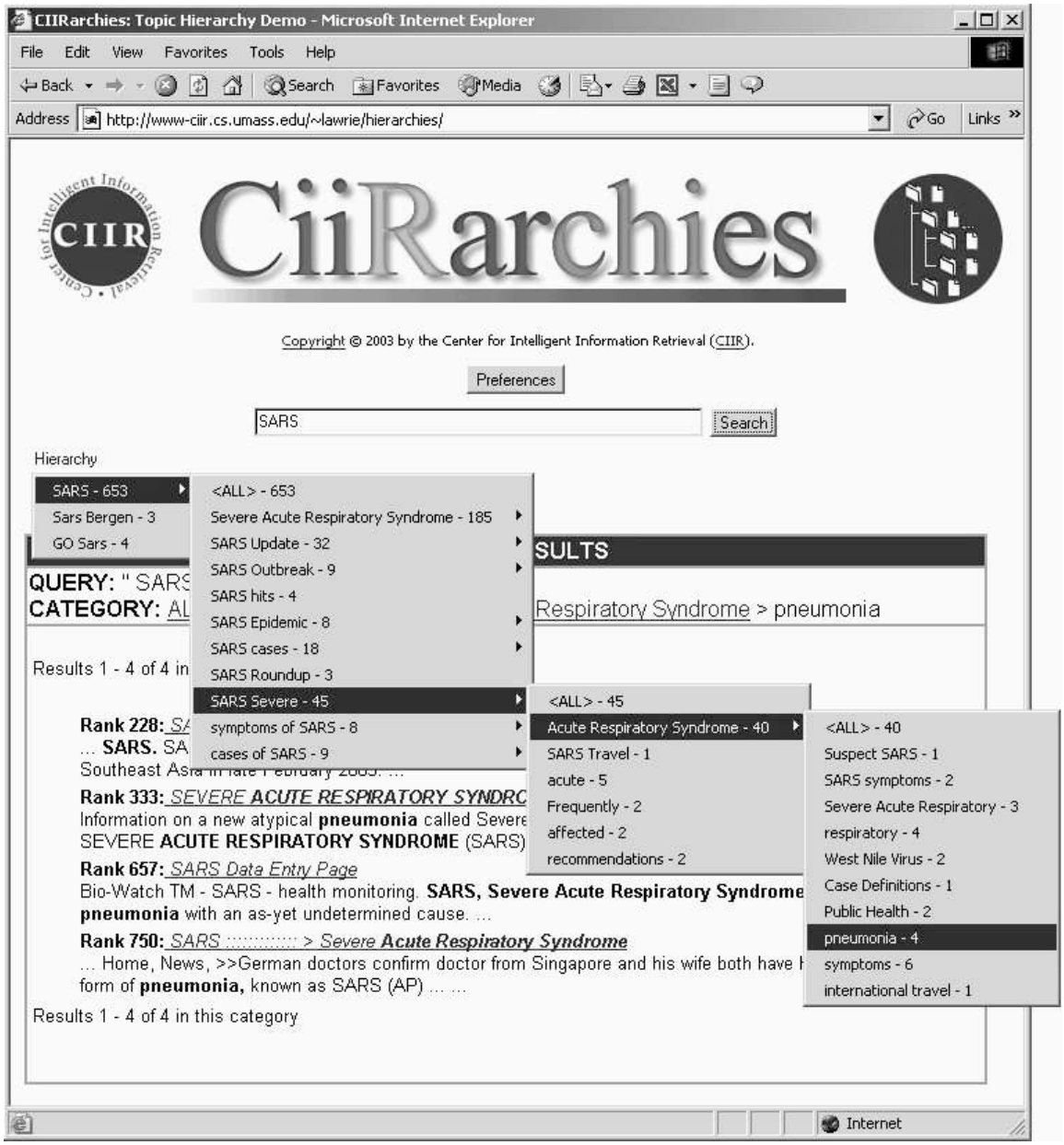


Figure 5.8. Shows the interface to interact with the hierarchical word-based summaries.

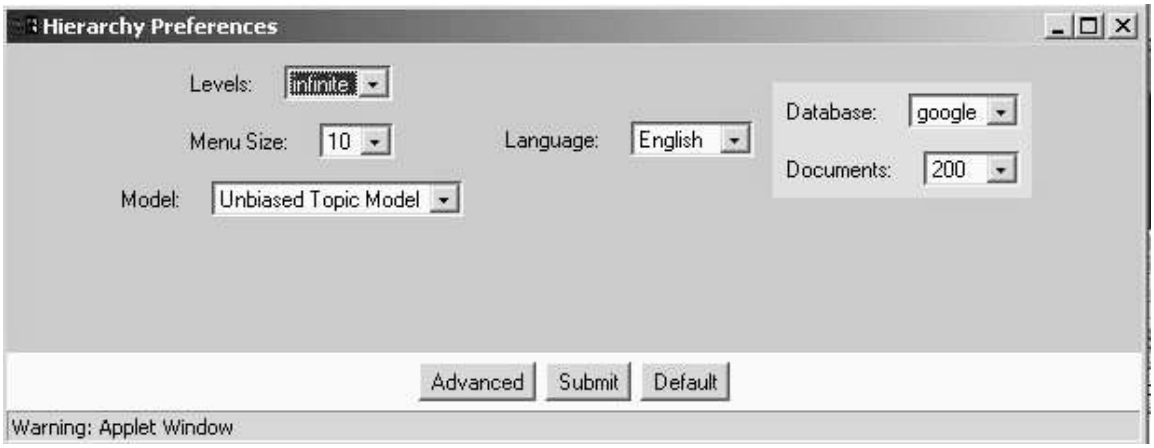


Figure 5.9. The user interface's standard preferences.

will create another level only when there are more than five documents at a node. If a specific number is selected, the software attempts to produce a hierarchy with the specified number of levels, but stops short of the required depth if no new topics can be identified. From an efficiency perspective, the depth plays a role in the time it takes to generate the hierarchy. The shallower the hierarchy, the faster it can be constructed.

The maximum number of topics at a single level of the hierarchy can be set from size five to fifty, at size increments of five, using the pull-down menu labeled "Menu Size". The maximum number of topics also affects the time it takes to generate the hierarchy; fewer topics lead to a faster construction of the hierarchy.

There are three choices for the model parameter accessed using the pull-down menu labeled "Model". The first choice is the "Query Topic Model" which biases the KL divergence contribution to the query when the topicality model is calculate. The second choice is the "Unbiased Topic Model", which calculates the topicality model without any knowledge of the query. The third choice is the "Uniform Model" which considers each word to equally likely as a topic. The differences in the "Query Topic Model" and the "Unbiased Model" are discussed in Section 3.4. The "Uniform Model" assumes that all candidate topic are equally likely to be a topic. Since no calculation is required, this is the

most efficient way to create the hierarchy, but using it negatively effects the quality of the hierarchy.

The language is selected using the pull-down menu labeled "Language". Currently, the only language fully integrated with the user interface is English; however, we are currently working on Chinese hierarchies. Although these hierarchies have been constructed, building and displaying them using this interface has not been fully tested.

Once the language is chosen, the database can be selected[1] using the pull-down menu labeled "Database". For English, there are a total of seven different databases. Six of these are combinations of the TREC collections[66], three of which include government and newspaper documents; the other databases are only the associated news documents. Two database are built from each of the following collections: TREC volumes two and three, TREC volumes two and four, and TREC volumes four and five. The seventh database comprises the documents indexed by the Google search engine.

The number of documents that will be retrieved is specified using the pull-down menu labeled "Documents". The choices are 50, 100, 200, 500, 750, or 1000. The maximum is 1000 because Google will not retrieve more than 1000 documents. The number of documents used to construct the hierarchy has an enormous influence on how fast the hierarchy can be generated. Fewer documents lead to faster construction of the hierarchy.

5.3.2 Advanced Preferences

The advanced preferences appear in Figure 5.10. They consist of eight parameters, two of which are not tested in this thesis. The first parameter alters the "Segment Size" of text used to calculate the predictiveness model. The second parameter, "Threshold", allows the user to set the threshold for the **DSPappox** algorithm. The third parameter, "Word Type", specifies the kinds of words that will be considered as topics. The fourth param-

[1]The language must be selected first because most collections contain documents from a single language. Google is an exception.

eter, “Enforce Domination”, restricts the words that can be chosen as subtopics to those that were considered subtopics when the parent topic was chosen. The fifth parameter, “Exclude Words Chosen at Higher Levels”, disqualifies a high level topic from appearing as a subtopic of another word. The sixth parameter, “Sub-Collections”, specifies whether to use sub-collections when calculating the topicality model. The seventh parameter, “All Terms”, allows all words to be candidates regardless of their KL divergence contribution. Finally, the eighth parameter, “Mixture Model”, indicates that the topicality and predictiveness models should be combined as a mixture of the two models rather than as a joint model.

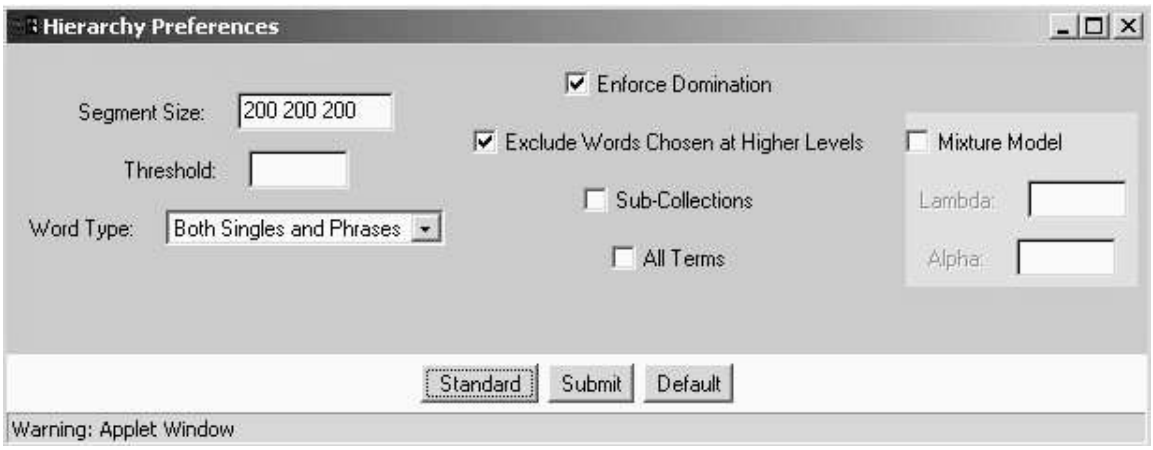


Figure 5.10. The user interface’s advanced preferences.

There are two different ways to specify the segment size parameter. One is to use the keyword “fulldoc” which sets the segment size to the document size, which is the maximum possible segment size. Otherwise numbers separated by spaces are expected for each level of the hierarchy. The top level segment size, specified by the first number, must be greater or equal to the other levels. In turn, the second level segment size mush be greater or equal to all lower levels. If there are three levels in the hierarchy, one valid segment size would be “200 100 50”, and another is just “200”. Specifying “200” is equivalent to “200 200 200”.

The threshold parameter is used to determine which relationships in the predictiveness model will be considered dominating relationships in terms of the **DSPapprox** algorithm. Since these relationships are all expressed as numbers between 0 and 1, the threshold must also be a number between 0 and 1. For example, if one wanted the hierarchical summaries to be more like subsumption hierarchies[59], then one could set the threshold to 0.8, which is the threshold used in that work to identify subsuming relationships. If the threshold is not set, the mean value of all the relationships is used, and will vary based on the document set and the level being constructed. In this thesis, we did not experiment with setting the threshold because we were uncertain that a universal threshold would adequately describe the relationships in which we are interested. The mean is used in all of our experiments.

The word type parameter specifies the conditions that will be imposed on candidate topics and subtopics. The choices are "Both Singles and Phrases", "Singles Only", and "Phrases Only". In order to create a hierarchy faster, the vocabulary can be restricted to only single words or only phrases. Phrases can be more informative than single words; for instance, "Hubble Space Telescope" is much more informative than "telescope" because it is a more specific term. Creating hierarchies containing all phrases is a very good way to limit the vocabulary, but invariably topics that do not occur in many phrases are missed.

The option to enforce domination restricts lower level terms to those that were dominated by the topic word and requires that subtopics be less frequent than the topics in the overall document set. By default this option is turned on because by definition, a hierarchy should have general terms at higher levels and become more specific at lower levels. This restriction is not implicitly enforced by combining the qualities of topicality and predictiveness. Setting this parameter also mandates that subtopics occur less frequently than their parent topics. The frequency of a term is a good indication of generality or specificity[59]. This means that lower level terms should be less frequent than their parents. Aside from enforcing a quality that should be present in the hierarchy, this restriction also limits the

number of terms that will be considered at a given level of the hierarchy, improving the running time of the algorithm.

Setting the option "Exclude Words Chosen at Higher Levels" attempts to eliminate repetitiveness of words in the hierarchy, which is one of the observed problems. Consider Figure 5.11 about "Abuses of E-mail", where the hierarchy is created without controlling repetitiveness. The topic "send" occurs in three of the four levels displayed in the figure. This type of repetitiveness can be eliminated by excluding occurrences of "send" at levels below the first occurrence of the topic. In effect, the subtopic is promoted to a higher level. In Figure 5.11 the subtopics of "send" can be explored when "send" is a subtopic of "abuses". The subtopics of "send" do not need to be re-explored when "send" is a subtopic of "e-mail". This does not eliminate all repetitiveness, however, since in Figure 4.5 the topics "combat abuses", "privacy abuses", and "report abuses" appear as subtopics of "e-mail" and "abuses". Since these are two different contexts where the subtopics occur, this may illuminate different aspects of the subtopics and thus is not undesirable repetition.

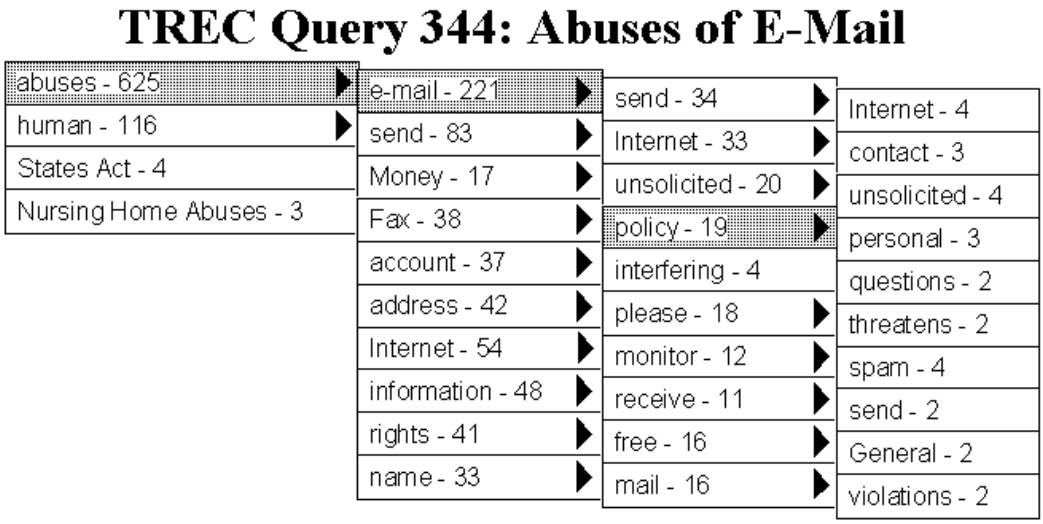


Figure 5.11. The figure shows a portion of the hierarchy created for the TREC Topic 344: Abuses of E-mail, using a uniform topic model. This hierarchy includes a number of non-topic words including "name", "address", and "please". It also illustrates the problem with repetition when words at higher levels are not excluded, as in the case of "send".

Making use of sub-collections was one way we found to improve the topicality model without biasing the model using the query. If the "Sub-Collections" parameter is activated, sub-collections will be used to calculate the topicality model. This parameter has no effect when creating hierarchies from documents retrieved from Google because that collection is not divided into Sub-Collections. However, the TREC collections are divided into sub-collections based on the source of the document, for example, the LA Times or the Federal Registry.

The "All Words" parameter provides another method of limiting the vocabulary. If this parameter is unselected, words must have a positive KL divergence contribution to be included in the vocabulary. A negative KL divergence contribution means that a word is less likely to appear in the document set than in general English. We eliminate these words because the likelihood that they are actually topic words is very small because of the way we estimate topicality. The increase in efficiency is usually worth any possible errors caused by eliminating words that would be selected as topics, and in fact, usually helps improve the hierarchy which is demonstrated in Section 6.2.1.

By setting the "Mixture Model" parameter, one changes the Term Selection Formula of Expression 3.1 to the following:

$$T* = \arg\max_{T \in W} \sum_{i=1}^{|T|} (1-\lambda)\mathcal{P}(\mathit{Topical}(w_i)|w1...w_{i-1}) + (\lambda)\mathit{Predictive}(w_i)|w1...w_{i-1}) \quad (5.1)$$

For each level, λ can be decreased by α. The values of λ and α are set by the user by filling in the fields "Lambda" and "Alpha" in the user interface. These two parameters default to 0.6 and 0.1 respectively. These particular values mean that as the hierarchy becomes deeper the words chosen favor the topicality model more. From a theoretical perspective, we have developed the Term Selection Algorithm as a joint probability rather than a mixture. Experimenting with the effects of a mixture model is left for future work.

5.4 Conclusion

This chapter explains implementation details and efficiency of the software created to automatically generate hierarchies. From the real time data, it is evident that we cannot generate the hierarchy fast enough in an interactive setting when using the full text of the document. However, if hierarchies are generated from summaries of documents, the procedure is fast enough in an online setting. Most of the time, the hierarchy can be constructed in less than 10 seconds. When using the web-based demonstration, it usually takes about a minute to retrieve the documents and create a hierarchy. This is because we are using the JAVA APIs to access the Google Search Engine, and such requests are given low priority by Google. Therefore, the requests take longer to service than the usual interactive behavior one expects from the search engine. If this software were fully integrated with a search engine, the running time would be fast enough to meet the demands of users.

The chapter also includes a discussion of the user interface and specifically describes each of the parameters that can be set when generating the summary. These parameters also play a significant role in the time it takes to create the hierarchy, although it is not as significant as the lengths of the documents.

CHAPTER 6

OFFLINE EVALUATION METHODS

The task of determining the quality of a hierarchy is particularly difficult because standard methods of evaluation cannot be applied. Information retrieval generally analyzes precision and recall; however, because the hierarchy allows documents to be found in several places, it is difficult to compute precision and recall. Another technique used in evaluation is to compare the results to a "gold standard". In the case of the hierarchies we are developing, there is no "gold standard". People have not built hierarchies for these documents, and even if they had, it is unclear whether we would want to mimic their results. Each individual who is asked to create a hierarchical organization of any set of documents is likely to create a different hierarchy based on personal biases. An alternative is to evaluate the hierarchy with respect to a particular task, which we do in Chapter 7. In this chapter, we focus on evaluations that measure particular attributes of the hierarchy in a non user-based environment. We refer to these as offline evaluations.

Our goal from the outset has been to develop a methodology for constructing topic-oriented hierarchies which summarize a group of documents and provide a user with a mechanism for accessing the documents. We have developed a number of offline evaluations that can be used to help us with the problem of ascertaining the quality of particular hierarchies. Such offline evaluations are critical to the development of the hierarchies because of the large number of parameters that must be set in order to construct a hierarchy. These parameters generate so many different versions of the hierarchy that it would be impossible to test each with a user study, especially because of the difficulty in designing a user study that will show significant differences. These evaluations also provide a mech-

anism for comparing the probabilistic hierarchies developed in this dissertation with the subsumption and lexical hierarchies introduced in Chapter 2.

This chapter presents the three different evaluations we use to measure the quality of the hierarchies. The evaluations are explained in Section 6.1. We use the evaluations to verify our hypotheses about the best parameter settings in Section 6.2, to compare probabilistic hierarchies to other methodologies for constructing topic hierarchies in Section 6.3, and to compare the hierarchy to other types of summaries and organizations in Section 6.4. These sections only present a discussion of the data collected. In Appendix D we present a sample of the actual results obtained from the statistical analysis. Finally, in Section 6.5 we conclude with a discussion.

6.1 Description of Evaluations

The evaluations presented in this section examine three important attributes of a hierarchy. It should be noted that this suite of evaluations does not cover all attributes one may wish to consider when determining the quality of a hierarchy, but it does provide an initial assessment of quality. In Section 6.1.1 we describe an evaluation which measures the quality of the hierarchy as a summary by computing the Expected Mutual Information Measure (EMIM) between the hierarchy and the original set of documents that the hierarchy summarizes. In Section 6.1.2 we describe an evaluation which measures the time required to read all relevant documents. This evaluation determines the maximal efficiency that a user could achieve when using the hierarchy. The third evaluation, discussed in Section 6.1.3, evaluates the access to the documents in the hierarchy.

6.1.1 Summary Evaluation

First, we introduce an evaluation that measures how well the terms chosen to be part of the hierarchy summarize the documents by calculating how well the chosen terms predict

the general vocabulary. Since the hierarchy is intended to be viewed as a summary, it is important to determine how well it summarizes the text in the documents.

This can be done using automated techniques because we view the hierarchy as a predictive summary; this means the words that occur in the hierarchies should predict or describe the other words in the text that are not in the hierarchy. If one were to remove the structure of the hierarchy, a bag-of-words is all that would remain. By treating the documents as a bag-of-words, we can compare the distribution of words found in the hierarchy to the distribution of all words. To do this, we calculate the Expected Mutual Information Measure (EMIM)[65] between the set of topic words and set of all non-stopwords occurring at least twice in the document set. We use this definition of the vocabulary because it is the same restriction that we use when selecting words for the hierarchy. This may introduce a slight bias; however, all the hierarchies incorporate these restrictions into the vocabulary, even those other methods that we use as a comparison. EMIM measures the extent to which the distributions of the topic words and the vocabulary words deviate from stochastic independence. The formulation of EMIM follows:

$$I(T,V) = \sum_{t \in T, v \in V} P(t,v) \log \frac{P(t,v)}{P(t)P(v)},$$

where T is the set of topic terms and V is the set of non-stopwords that occur at least twice. In order to calculate the joint probability, we use the formulation in Lavrenko and Croft[35]:

$$P(t,v) = \sum_{d \in D} P(d)P(t|d)P(v|d),$$

where D is the set of documents and $P(d)$ is a uniform distribution. The greater the dependence between the two distributions, the better the hierarchy is a summary of the text.

Once EMIM has been calculated for each set of topic terms, we use ANOVA analysis to compare the different types of hierarchies, and Tukey's Honest Significant Difference at p=0.05 to find where there are significant performance differences in the hierarchies. For

these experiments we looked at the effect that different numbers of topics at each level of the hierarchy have on the results.

6.1.2 Accessing Relevant Documents

One way to use the hierarchy would be to attempt to read all relevant documents while reading as few non-relevant documents as possible. This evaluation determines the upper bound to the efficiency that can be realized by using a particular hierarchy. Although we need users in order to evaluate which groups of documents in the hierarchy are likely to be examined, without a user we can determine the optimum method for reading all relevant documents. Unlike a ranked list, which provides a single ordering of the documents, there may be several different configurations of a hierarchy that would enable a user to read all the relevant documents. In this evaluation, we only calculate an upper bound on performance. This evaluation is similar to one purposed by Leuski[37], which evaluated clustering is a spring embedded visualization.

The evaluation uses the location of all relevant documents in the hierarchy to find the most efficient way that a user could read all relevant documents. Several constraints are built into the evaluation. First, only leaf-node groupings in the hierarchy are considered when looking for relevant documents. This constraint is included because our user study results indicate that users prefer to examine the most specific groups of documents in the hierarchy. Users in our study chose leaf nodes more than half the time (62%), thereby exhibiting a moderate preference for such nodes. A second constraint is that once a group of documents is selected, all documents in that group are considered to be read by a user. This is not an entirely accurate model of a user's behavior; however, it is very difficult to determine which documents a user would read, so we have accepted this limitation. Since some of the documents would be non-relevant and we cannot foresee which ones a user would choose to read, we decided to include all documents, both relevant and non-relevant, in the score of the hierarchy. Because the best case occurs when very few non-relevant

```
Score(RelDocs)
docsViewed ← ∅
foreach d ∈ RelDocs
        if (d ∈ docsViewed)
          d.pathlength ← 0
        else
          leaves ← getLeaves(d)
          if (|leaves| > 0)
            l ← leafWithMostRelDocs(leaves)
            d.pathlength ← l.newMenus + l.totalNewDocs
            docsViewed ← l.docs ∪ docsViewed
          else
            d.pathlength ← −1 #no path
```

Figure 6.1. Algorithm assigns a score to the hierarchy based on the path length to each relevant document.

documents are grouped with relevant ones, this evaluation favors a hierarchy that has small clusters with a very high concentration of relevant documents, so that only a few non-relevant documents need to be read.

Such an evaluation is important because the hierarchy should allow efficient access to the documents for which a user is looking. The evaluation estimates the time it takes to find all relevant documents by calculating the total number of hierarchy menus that must be traversed and the number of documents that must be read. The algorithm aims to find an optimum path through the hierarchy by traveling to nodes that hold the greatest concentration of relevant documents. Since we begin with the knowledge of where the documents are located, and which ones are relevant from the TREC judgments of the documents, our algorithm iterates through all relevant documents and assigns a path length to each. Any relevant documents not found in the hierarchy (which is possible) are assigned a path length of negative one as an error flag. The total path length for a hierarchy is the summation of all positve document path lengths.

Figure 6.1 shows the path length algorithm, which assigns a numeric score to the hierarchy. Given that documents often belong to more than one leaf in the hierarchy, it is necessary to choose which instance of each document will be used to calculate the path length. From among the leaf menus, we favor the topics with large numbers of relevant documents and small numbers of non-relevant documents. The path to a relevant document is composed of previously unexplored topic menus, and the unread documents associated with the leaf. It is assumed that all new documents in the group are read by the user in the process of reading the relevant ones.

Although this algorithm leads to a succinct analysis of the hierarchy, it is worth noting that it contains certain simplifying assumptions. First, all documents are regarded as equal size despite the variability in document length. Similarly, all menus are treated equally despite the variability in their length. Finally, when computing the path length, documents and menus are treated the same; i.e., the time and effort required to read a document is regarded as being the same as that to read a menu. This most likely overestimates the effort required to read a menu; however, it is unclear exactly how this should be modeled. In most instances, the number of menus examined across different types of hierarchies is very similar. Differences are observed in the number of documents a user must read, so this assumption has a very minor effect of the results.

Hierarchies are assigned a score based on the average path length to a relevant document in the hierarchy, which is computed by finding the sum of all path lengths and dividing by the number of relevant documents found in the hierarchy. A lower score denotes a superior hierarchy. Once all the hierarchies are scored, they are compared on the basis of this average score calculated by the algorithm. This is used instead of doing a straight comparison of total path lengths because it is possible that some relevant documents are unreachable, as previously mentioned. If we did not use the average, the total path length for a particular hierarchy could end up being shorter simply because it excluded relevant

Hierarchy	*% no path*
Lexical	3.04%
Subsumption	5.96%
Probablisitc	6.28%

Table 6.1. The percentage of relevant documents which have no path in the hierarchy with regards to the total number of relevant documents retrieved for a query.

documents. By using the average path length, we neither reward nor penalize a hierarchy for excluding relevant documents.

The average number of relevant documents that are excluded from the hierarchy can be found in Table 6.1. These averages are based on the number of relevant documents excluded compared to the total number of relevant documents.

When comparing different hierarchies, we use ANOVA and determine if there are significant differences in the scores with Tukey's Honest Significant Difference at $p = 0.05$.

6.1.3 Reachability

Another important attribute of the hierarchy is the ability to find documents within it without regard to being relevant, which we refer to as the reachability of the hierarchy. Because of the statistical nature of the probabilistic hierarchies, there is no guarantee that there will exist a path to all documents used to create the hierarchy. Additional words will not be used once all the vocabulary is "predicted" by the words already chosen. It is possible that this "stopping criterion" based on prediction may be reached despite the fact that no topic words are present in a particular document. A document may also be excluded by the evaluation if it does not occur in any small groups. This behavior occurs because of the way we formulate the evaluation. During the user study, we found that ninety percent of the time users chose topics that contained fewer than 30 documents. We concluded that it is unlikely that a user will explore a group that is too large. This evaluation imposes different cut-offs as the maximum size group a user would explore, and then calculates the percentage of documents that are reachable.

We use the evaluation to compare different methods of creating hierarchies. We assert that the more documents to which a user has access, the better the hierarchy. In order to find significant differences among the hierarchies, we use ANOVA and determine if there are significant differences in the scores with Tukey's Honest Significant Difference at $p = 0.05$.

6.2 Examination of Different Parameter Settings

In the following sections we discuss seven different parameters, and discuss the effects of the possible values individually. We then look at the cumulative effect of five parameter settings[1]. In this analysis we divided the different parameter settings into a number of groups, since differences among the hierarchies become less clear as more parameter settings are included in a single comparison. Before performing the evaluations, we examined examples of each parameter setting to determine which setting seemed to be the best based on the kinds of words included in the hierarchy. In the following sections, we use to evaluations to determine if the hierarchies that appear to be the best from a user perspective perform well in the evaluations.

For these evaluations we examine three different groups of query sets comprising 50 queries each. The queries were used to retrieve documents in the three different databases described in Chapter 4. We also look at four different versions of the hierarchy, each with a different number of maximum topic words. The values we used for the maximum topic words parameter are 5, 10, 15 and 20. We do not test larger hierarchies because we believe users would be overwhelmed by more topics. In our user study, we limited the hierarchies to 10 topics per level, and one participant commented that they felt there were too many topics in the hierarchy.

[1]The other two parameters are not used because it was unclear how they should be set in the comparison.

We present results from both full text hierarchies and snippet hierarchies. Recall that the full text hierarchies are very different from snippet hierarchies because the snippet hierarchies are composed of documents with very few words. It is, therefore, not surprising that our results will differ depending on the type of hierarchy. A sample of the data used to produce the analysis appears in Appendix D. All tables are not included because of their size.

6.2.1 Using All Words

In Section 5.3.2 we described how we can limit the number of words eligible to be part of the hierarchy by requiring that a word contribute positively to the KL-divergence score. We call this parameter "All Terms". When it is not active, the vocabulary will be limited by KL-divergence. One of the main reasons for limiting the vocabulary is that the hierarchy can be generated faster if fewer words are possible topic words.

Evaluating the hierarchy using the EMIM evaluation, we find that all hierarchies which use the KL-divergence as a cut-off and are constructed from the full text of documents are significantly better than hierarchies that do not use the cut-off. The full text documents are those retrieved from the complete and news databases. We found exactly the opposite for hierarchies created from web snippets. For all query sets, the hierarchies created with all the words were significantly better than those created when not using all the words. This is likely due to the fact that there are only a few words that both describe the topic and appear in many documents. Once these words have been used, the hierarchy will either include general English words that also occur in many documents, or words that occur in only a few documents but are better at describing the topic. The former occurs when all words are used and because these words appear in more documents, hierarchies using them perform better in the EMIM evaluation.

We found significant differences between some sizes of hierarchies created from query set 251-300 when using the Access to Relevant Documents evaluation and the complete

database for retrieval. For size 10 and 20, we found that it is significantly better to use all words, but for 15 topics it is significantly better to use the KL-divergence cut-off. For all other comparisons, there is no significant difference. It should be noted that this evaluation was used only on the complete and news databases where NIST provides relevance judgments. We do not have relevance judgments for the documents in the snippet hierarchies.

Evaluating the hierarchy using the Reachability evaluation, we found significant differences between full text hierarchies for some query sets. Of the twenty-four different groups, twelve reported significant differences between hierarchies that were created using the KL-divergence cut off and using all words. Of the twelve, seven agreed with the results of the EMIM evaluation. We found that in following instances using the KL-divergence cut-off significantly improves the hierarchy:

- Query set: 201-250; Database: complete; Size: 10
- Query set: 201-250; Database: news; Size: 5
- Query set: 251-300; Database: complete; Size: 20
- Query set: 301-350; Database: complete; Sizes: all.

For five other instances we found that using all terms significantly improved the hierarchy. This is contradictory to the results of the EMIM evaluation. The instances are as follows:

- Query set: 201-250; Database: complete and news (both); Size: 15
- Query set: 251-300: Database: complete; Size: 15
- Query set: 251-300; Database: news; Size: 10
- Query set 301-305; Database: news; Sizes: 15, 20.

All of the snippet hierarchies reported that using all words is significantly better according to the reachability evaluation.

From these tests, we conclude that using the cut-off is preferable when hierarchies are built from the full text of the documents. There are only six tests that disagree with this assessment; four come from the Reachability evaluation and two from Accessing Relevant Documents. However, in the case of the web hierarchies, it is clear that using all words creates significantly better hierarchies according to both the summarization evaluation and the reachability evaluation.

6.2.2 Excluding Higher Level Words

In Section 5.3.2 we discussed the possibility of excluding words that appeared at higher levels of the hierarchy. We refer to this as a promotion of the topic. We believe this improves the hierarchy because it eliminates redundant subtopics.

Using the EMIM evaluation, we found that excluding higher level words significantly improved the hierarchy in 8 out of 36 instances. All of these instances occurred when the hierarchy was created from the full text of the documents. In the other 16 instances, there was no significant difference between excluding higher level words and not excluding them. When hierarchies were created from document snippets, however, not excluding higher level words significantly improved the hierarchy in all instances. This result is most likely due to the fact that vocabulary is so limited in snippet hierarchies that the repetition enables them to address a topic multiple times and include different aspects.

In the evaluation Access to Relevant Documents, there was no significant difference between excluding higher level words or not for all types of hierarchies.

The Reachability evaluation showed that the topic promotion parameter yielded significant differences between the types of hierarchies for all instances of the hierarchies created from full documents. In 23 of 24 instances, excluding high level words produced significantly better hierarchies. However, for query set 251-300, where twenty topics were chosen and retrieval was performed on the complete database, not excluding higher level terms produced a better hierarchy. This may be because the documents retrieved for these

queries were much longer than the documents retrieved for any other set of queries or from any other databases. In the case of the snippet hierarchies, all 12 instances yielded significantly better hierarchies when higher level words were not excluded.

From these tests, it can be concluded that excluding higher level words will not harm the effectiveness of the hierarchy in most cases when the hierarchy is created from full text documents. However, according to these evaluations, when creating snippet hierarchies, it is better if higher level words are not excluded.

6.2.3 Enforcing Domination

In Section 5.3.2 we discussed the option of restricting words chosen as subtopics to those that are less frequent than and dominated by the main topic. To be dominated by the main topic, the subtopic must co-occur with the main topic more frequently than the average co-occurrence rate. We believe that requiring subtopics to be dominated improves the characteristics of the hierarchy by making it more likely that subtopics are highly related to and more specific than the main topic.

The EMIM evaluation detected significant differences in the results from changing this parameter in all instances of the hierarchies. When hierarchies were produced from full text documents, enforcing domination produced significantly better hierarchies. In seven of the eight instances involving hierarchies produced from snippets, not enforcing domination produced significantly better hierarchies. However, in the case of query set 251-300, with a hierarchy of at most 5 topics per level where web snippets were retrieved, enforcing domination produced a better hierarchy. The snippets generally benefit from not enforcing domination because when domination is enforced, the hierarchies are actually smaller.

The evaluation of the Access to Relevant Documents found two instances of significant differences. For the query set 251-300 with a hierarchy of size 15 retrieved from the complete database, enforcing domination produced a better hierarchy. However, for query

set 251-300 with a hierarchy of size 5 retrieved from the complete database, not enforcing domination produced a better hierarchy.

The Reachability evaluation generally agreed with the EMIM evaluation. For full text hierarchies, 23 of 24 instances revealed significant results. Most of them found that enforcing domination produced a significantly better hierarchy. In four instances, the opposite was found:

- Query set: 251-300; Database: complete; Sizes: 15, 20
- Query set: 301-350; Database: complete; Sizes: 15, 20.

For query set 301-350 with documents retrieved from the complete database and 10 topics per level, there was no significant difference between the hierarchies. For the snippet hierarchies, all instances revealed that not enforcing domination produced significantly better hierarchies.

From these evaluations, it appears that enforcing domination generally improves full text hierarchies but does not improve snippet hierarchies. This is not surprising given the differences in the text used to produce the hierarchies.

6.2.4 Comparing Topic Models

In Chapter 3 we discussed different methods for calculating the topic model which was used to estimate the probability that a word is a topic word. In this section we compare a query-biased topic model, an unbiased topic model, and a uniform topic model. The query-biased topic model is presented in Section 3.4, and uses the query to improve the estimates of words closely related to the query. The unbiased topic model is also presented in Section 3.4 and estimates the topic model without considering the topic. The uniform topic model estimates that every word is equally likely to be a topic word.

The EMIM evaluation could not detect significant differences between the query-biased model and the unbiased model in any instance. However, there was significant difference

between the uniform model and the other two models in every instance. The full text hierarchies found that hierarchies created using the query-biased or unbiased model were significantly better, while the snippet hierarchies created from the uniform model were significantly better. This seems to reveal that topicality does not help choose better summarization words in the case of the snippet hierarchy. The problem is that it is difficult to interpret some of these words as topic words, which means a user might have a more difficult time using the hierarchy. Examples of such words include "first", "recognized", "major" and "number", all of which came from the snippet hierarchy for TREC Topic 303 created using a uniform topic model. In fact, even using a model other than the uniform model does not eliminate all undesirable words because a non-topic term like "Welcome" may occur in a disproportionate number of snippets for some queries. This is actually a more common problem of the unbiased topic model than the query-biased topic model.

In the evaluation of the Access to Relevant Documents, most of the instances revealed no significant differences among the different models. For the query set 301-350, six of the eight instances found than the query-biased model produced significantly better hierarchies that the uniform topic model. There were no significant differences for hierarchies with maximum number of topics of 10 and 15 where the documents were retrieved from the news database. For the query set 251-300, the only significant difference was found in the hierarchies with a maximum number of topics of 5 where the documents were retrieved from the complete database. Here it was found that the query model produced significantly better hierarchies than the unbiased model.

The Reachability evaluation showed that the query-biased and unbiased models produce significantly better hierarchies than the uniform model if the hierarchies are produced from full-text documents. In 18 of 24 instances the unbiased model produced a better hierarchy than the query-biased model. No significant difference between the models was found for the following:

- Query set: 201-250; Database: both; Size: 5

- Query set: 251-300; Database: complete; Sizes: 5, 15, 20
- Query set: 301-350; Database: news; Size: 5.

When snippet hierarchies were produced, all of them revealed a significant difference between all three models. They found that the uniform model produced the best hierarchies, followed by the unbiased model.

From these evaluations, we conclude that the topic model improves the full document hierarchies, and that we can produce good summaries of the documents even if a query is unavailable. This means that we should be able to produce good summaries of document sets with many disparate topics. When snippets are used as the text to produce the hierarchy, the uniform model produces better hierarchies in terms of the two measures. Because these hierarchies do not seem as useful upon inspection, it may be that a better test would be to evaluate how well the hierarchy summarizes the full text of the web pages rather than the snippets. Changing the nature of the EMIM evaluation will have no effect on the Reachability evaluation. If it turns out the other models do perform better in the alternative EMIM evaluation, the uniform model will still create a hierarchy with the best access to documents.

6.2.5 Use of Sub-collections

In Section 3.4 we discussed using sub-collections as a way to make sure that frequent words in one particular sub-collection do not score highly with respect to KL-divergence and thus appear to be better topic words than they actually are. This phenomenon is observed in the complete database, but not in the news database where the language that occurs is more standardized. The snippet hierarchies are not part of this evaluation because we do not have sub-collections for the web data.

In the EMIM evaluation, 4 of the 24 instances revealed significant differences when sub-collections were used. All four showed that using sub-collections significantly improved the hierarchy, and they all occurred when the documents were retrieved from the complete

database. The significant differences occurred for query set 201-250 where hierarchies had a maximum number of topics of 5 and 15, and for query set 251-300 where hierarchies had a maximum number of topics of 5 and 20. In all other instances, there were no significant differences between using sub-collections and not using sub-collections.

When evaluating the Access to Relevant Documents, 5 of the 24 instances revealed significant differences. All five of them showed that making use of sub-collections significantly improves the hierarchy. Again the significant differences were observed when retrieval was performed on the complete database. These significant differences occurred for query set 201-250 when the maximum number of topics was 20 and for all sizes in query set 301-350.

The Reachability evaluation found significant differences in 16 of the 24 instances. For this evaluation, all instances where retrieval was performed on the complete database found that using sub-collections significantly improved the hierarchy. The evaluation also found that for query set 301-350, using sub-collections significantly improved the hierarchy when retrieval was performed on the news database.

Since none of the evaluations found that using sub-collections was detrimental to the performance of the hierarchy and many confirmed that using such information improves the hierarchy, we conclude that such information should be used when it is available.

6.2.6 Combined Effect of Parameters

In this section we look at the combined effect of setting the parameters we have discussed in the previous 5 subsections versus not setting these parameters. This means that, in the case of the full text hierarchy, we compare the combined effect of using the KL-divergence cut-off, excluding higher level words, enforcing domination, using the unbiased topic model, and using sub-collections to not doing any of these things and using the uniform topic model. In the case of the snippet hierarchy, we compare using the KL-divergence cut-off, excluding higher level words, enforcing domination, and using the un-

biased topic model to using the uniform topic model without setting any other parameters. The evaluations revealed that the collective effect of these parameter settings is very similar to comparing them each individually.

The EMIM evaluation found significant differences in every single instance. All the full document hierarchies with the attributes turned on were significantly better than the hierarchies without the attributes. The snippet hierarchies without the attributes were significantly better than those with the attributes.

When comparing hierarchies with the Access to Relevant Documents evaluation, 7 of the 24 instances found significant differences. They revealed that turning on the attributes improved the hierarchy for the following instances:

- Query set: 251-300; Database: news; Size: 20
- Query set: 301-350; Database: complete; Sizes: 5, 10, 20
- Query set: 301-350; Database: news; Sizes: 10, 15, 20.

The Reachability evaluation found that turning on the attributes always significantly improved the full text hierarchies, and turning off the attributes significantly improved the snippet hierarchies for all instances.

From these evaluations, we conclude that turning on all the parameters improves the full text hierarchies, but does not improve the snippet hierarchies.

6.2.7 Phrases verses Single words

In this section we examine the effect of creating hierarchies with only single words, only phrases, or combining the two. An examination of the hierarchies reveals that because phrases are more specific, we believe it is easier for a user to interpret them as a topic. However, there may be some topics that cannot be expressed by a phrase, and thus it seems prudent to allow both to be topic words.

The EMIM evaluation revealed that all instances are significantly different. For the hierarchies created from full text, using only phrases produced a significantly better hierarchy. Also, in most cases using both single words and phrases was significantly better than using single words alone. There was no significant difference between both single words and phrases and only single words in the following instances:

- Query set: 201-250; Database: complete; Sizes: 5, 10
- Query set: 251-300; Database: complete; Sizes: 5, 10
- Query set: 251-300; Database: news; Size: 5
- Query set: 301-350; Databases: both; Size: 5.

This means that when fewer words are chosen as topics it is less clear that using single words is detrimental; however, it is clear that using phrases alone creates a better hierarchy.

For the case of snippet hierarchies, the EMIM evaluation found that using single words was always significantly better than the other two groups. In all but two instances, using both single words and phrases produced a significantly better hierarchy than using phrases alone. The cases where there was no significant difference between using both and using only phrases occurred for query set 251-300 with hierarchies having a maximum number of topics of 15 and 20.

When examining the results of the evaluation of Accessing Relevant Documents, significant differences among the three sets of words were harder to find. For query set 201-250, using phrases only was significantly better than one of the other sets of words in five instances. In four instances, using phrases was significantly better than using only single words. These included when the hierarchies had a maximum number of topics of 5 and retrieval was performed on the complete database, and when retrieval was performed on the news database for all four maximum numbers of topics used. For query set 251-300, using only phrases produced a significantly better hierarchy that the other two sets when

there were a maximum of 5 topics and retrieval was performed on the news database. For this query set, single words only produced a significantly better hierarchy than both single words and phrases in the case where the maximum number of topics was 10 and retrieval was performed on the complete database; and single words alone produced a significantly better hierarchy than phrases only in the case where the maximum number of topics was 15 and retrieval was performed on the complete database. For query set 301-350, both single words and phrases was better than only phrases in the case where the maximum number of topics was 20 and retrieval was performed on the complete database. In light of these results, it is difficult to draw any conclusions about which set of words is likely to produce the best hierarchy in terms of accessing relevant documents.

For the Reachability evaluation, using both single words and phrases was significantly better than the other two sets for all instances of full text hierarchies. For query set 201-250, phrases only was significantly better than single words only for all instances of full text hierarchies. However, for query set 301-350, single words were significantly better than phrases in 7 of 8 instances. The only instance where phrases were better than single words was in the case where the maximum number of topics was 5 and retrieval was performed on the news database. For the query set 251-300, single words were significantly better than phrases in 6 of 8 instances. There was no significant difference between the two in the cases where the maximum number of topics was 5.

The results of the snippet hierarchies were almost the complete opposite. In all cases, single words alone produced the significantly better hierarchies than the other two sets of words. When comparing the other two sets, single words and phrases performed significantly better than only phrases.

The results of these evaluations are ambiguous for full text hierarchies. In terms of summarization, it is better to use phrases, while in terms of reachability it is better to use both single words and phrases. However, for snippet hierarchies, the evaluations clearly favor hierarchies created from single words. This could be because the documents are so

short and that using single words only is more inclusive, even though it may be harder for a user to interpret.

6.2.8 Segment Sizes

In Section 3.5 we discussed the fact that the segment size should represent the portions of the document that are related to each other in order to best calculate the predictiveness quality of a word. In this evaluation we tested segment sizes of 200, 400, 600, 800, and 1000 words. This evaluation was conducted only on the news and complete databases; since the snippet documents are generally less than 100 words, there would be no distinction between the different segment sizes.

The EMIM evaluation found that a segment size of 200 was significantly better than all the other segment sizes in 23 of 24 instances. For the query set 301-350, there is no significant difference between hierarchies with a segment size of 200 and a segment size of 400 when there is a maximum of 5 topics retrieved from the complete database. The second-best segment size is 400 in all cases, although in about half of the instances there is no significant difference between it and the next segment size.

The evaluation of the Access to Relevant Documents found only six significant differences. For the query set 201-250, the segment size of 200 was significantly better than the segment size of 1000 for hierarchies with documents retrieved from the complete database and a maximum of 20 topics; and the segment size of 800 was significantly better than the segment size of 200 for hierarchies with documents retrieved from the complete database and a maximum of 5 topics. For the query set 251-300, the segment size of 400 was significantly better than the segment size of 200 for hierarchies with documents retrieved from the news database and a maximum of 20 topics; and the segment size of 600 was significantly better than the segment size of 200 for hierarchies with documents retrieved from the news database and a maximum of 15 topics. For the query set 301-350, the segment size of

200 was significantly better than the segment size of 1000 for hierarchies with documents retrieved from both the complete and news database and a maximum of 20 topics.

The Reachability evaluation found that the optimal segment sizes for every instance were ordered from 200 to 1000, where 200 always produced the best hierarchies. There were not always significant differences between the segment sizes, however. The segment size of 200 produced significantly better hierarchies than all other segment sizes in all but one instance. The exception was for the hierarchies created for the query set 251-300 with documents retrieved from the complete database and a maximum of 5 topics. In this case, there was no significant difference between segment sizes of 200 and 400. In all cases, the segment size of 400 was significantly better than the next segment size.

The EMIM and Reachability evaluations agree that the 200 segment size is the best of the ones tested. Since smaller segment sizes were not tested, we cannot conclude whether this segment size yields the absolute best results.

6.3 Comparing Different Types of Hierarchies

In this section we compare the probabilistic hierarchy to the lexical hierarchy and the subsumption hierarchy, both of which were introduced in Chapter 2. We feel that this comparison is important because the probabilistic hierarchy was developed from the ideas of these hierarchies and thus they represent a kind of baseline. The evaluations will only compare hierarchies created from full text documents.

In a personal communication with Mark Sanderson [58], he stated that creating web hierarchies with subsumption did not work as well as when hierarchies were created from documents in the TREC collections. From this it is unclear how well subsumption would perform when creating hierarchies from snippets. A full evaluation should be conducted to determine if the performance suffers when snippets are used as input.

When comparing hierarchies created from the full text documents using the EMIM evaluation, the subsumption hierarchy is always significantly better than the lexical hierarchy.

The probabilistic hierarchy generally fell in between the two. In nine of the twenty-four instances there was no significant difference between the subsumption and probabilistic hierarchy. For the query set 201-250, there was only one instance where subsumption performed significantly better than the probabilistic hierarchy, which was for hierarchies with a maximum of 20 topics and documents retrieved from the complete database. For the query set 251-300, there was only one instance where there was no significant difference between the subsumption and probabilistic hierarchy, which was for hierarchies with a maximum of 5 topics and documents retrieved from the news database. This same instance was also the only one for the query set 301-350 where there was no significant difference between subsumption and probabilistic hierarchies. The probabilistic hierarchy performed significantly better than the lexical hierarchy in 20 of 24 instances. The instances where there was no significant difference occurred in query set 251-300 for all hierarchies created from documents retrieved from the complete database. From this evaluation, it seems that probabilistic hierarchies perform best when trying to find a very small number of topics, and when the documents are shorter as in the case of the documents from the news database.

The evaluation of the ability to access relevant documents found significant differences in only eight instances. These occurred for query set 251-300 where subsumption was found to be significantly better than the lexical hierarchy when hierarchies were created from documents of the news database with maximums of 5, 10, and 15 topics, and from the complete database with a maximum of 10 topics. In the case of the hierarchy created from the news database with a maximum of 10 topics, the subsumption hierarchy was also significantly better than the probabilistic hierarchy. For the query set 301-350, the probabilistic hierarchy was significantly better than the lexical hierarchy for hierarchies created from documents retrieved from the complete database with a maximum of 10, 15, and 20 topics, and for hierarchies created from documents retrieved from the news database with a maximum of 10 topics. In addition, this last instance found that the subsumption hierarchies were significantly better than the lexical hierarchies.

The Reachability evaluation found that the probabilistic hierarchy performed significantly better than the subsumption hierarchy in every instance. The lexical hierarchy also performed significantly better than subsumption in 23 of 24 instances. The only instance where there was no significant difference between lexical and subsumption hierarchies occurred for query set 251-300 where the hierarchies were created with a maximum of 5 topics and the documents were retrieved from the complete database. In 11 of 24 instances, the probabilistic hierarchy performed significantly better than the lexical hierarchy. Ten of these instances occurred when documents were retrieved from the complete database. In 8 of the 24 instances, the lexical hierarchy performed significantly better than the probabilistic hierarchy, all of which occurred when the documents were retrieved from the news database. In 5 instances there was no significant difference between the two.

From the evaluations, we conclude the probabilistic hierarchy is slightly better at providing access to the documents in general, although not necessarily the relevant documents. When examining the summarization results, it is important to keep in mind that the subsumption hierarchies were specifically designed to heuristically choose informative words. It is therefore unremarkable that it performed best in the summarization evaluation. The fact that it did not perform well in the Reachability evaluation reveals that the words it chooses are not as inclusive as the words chosen by the probabilistic hierarchy.

6.4 TF.IDF and the Ranked List

In this section we compare the hierarchy to other techniques used in information retrieval. In Section 6.4.1 we discuss comparing the hierarchy to the top TF.IDF words to find out which set of words is the better summary. In Section 6.4.2 we compare the hierarchy to the ranked list to ascertain which organization provides better access to documents based on the Reachability evaluation.

6.4.1 Hierarchy versus TF.IDF

One of the popular techniques for choosing a set of words to describe a document set is to calculate the TF.IDF weight of the words and select the top n as a description. This was used in Scatter/Gather research [14], among others. For this reason, we test the words chosen to be part of the hierarchy against an equal number of the top TF.IDF words and use the EMIM evaluation to determine which set of words is a better summary of the document set.

Our results from the test were conclusive. For all query sets, databases, and hierarchy types, the probabilistic hierarchy's words form a significantly better summary than the top TF.IDF words. This evaluation reveals that this method of choosing summarization words is better than the current state of the art.

6.4.2 Hierarchy versus the Ranked List

In this section we examine the access to documents provided by the hierarchy and the ranked list. This test assumes that a user would have a fixed policy when presented with either a ranked list or a hierarchy. In the case a ranked list, a policy could be that she would examine the top 100 documents. In the case of the hierarchy, a policy could be that she examines all documents that could be found in groups with 10 or fewer documents. In either case, the policy will prevent the user from accessing a portion of the document set. These policies allow us to equate the access provided by the ranked list to the access provided by the hierarchy.

The results reveal that for hierarchies generated from the complete database with 10 topics per level, a policy that looks only at documents in groups of 30 or fewer provides better access than looking at the top 250 documents in a ranked list, but inferior access than examining 400 documents in a ranked list. A policy that looks at documents in groups of 5 or fewer for the same size hierarchy falls between examining 200 and 100 documents. We believe the average user will look at fewer than 100 of the retrieved documents. Even

when a user follows the most stringent policy, the hierarchy provides better access to the documents than the ranked list.

6.5 Conclusion

When evaluating the parameters, the full text hierarchies performed as expected, revealing that techniques which appear, from visual inspection, to improve the hierarchies also produce better hierarchies according to our measures. The same is not true for the snippet hierarchies. A qualitative analysis of the snippet hierarchies led us to conclude that the same parameter settings used for the full-text hierarchies produce the best snippet hierarchies. The evaluations performed in this chapter did not confirm this analysis. Because of this, we performed a user study using the snippet hierarchies, which is discussed in the next chapter. This study looks at how easily a user can find information when using the hierarchy versus using a ranked list.

The probabilistic hierarchy performs well when compared to the lexical and subsumption hierarchy. Although it does not always perform the best, it rarely performs the worst. When comparing the hierarchy to non-hierarchical tools, the hierarchy selects better words in terms of summarization than TF.IDF weighting does, and the hierarchy provides better access to documents than a ranked list does.

CHAPTER 7

USER STUDY

In this chapter we discuss the user study that was performed to compare the use of a hierarchy to the use of a ranked list. In Section 7.1 we discuss the experimental design. In Section 7.2 we present an example of how the hierarchy is used in the study. In Section 7.3 we examine the statistical results. In Section 7.4 we discuss the user comments, which is followed by a general discussion in Section 7.5. In Section 7.6 we conclude.

7.1 Experimental Design

This experiment was designed following the model developed by NIST for the interactive track in TREC 6, 7, and 8[51, 52, 26]. In these experiments users are asked to perform several information retrieval tasks, some on a control system and others on an experimental system. For each task, the user is presented with a topic and asked to identify as many different *aspects* of the topic as she can in a specified amount of time. An aspect can be thought of as an answer to the topic. For example, if one were looking for countries that have monarchs, an aspect would be a country whose government type is a monarchy. Aspect retrieval is somewhat different than standard document retrieval as studied in information retrieval. Rather than measuring the user's ability to find relevant documents, this study measures the ability of the user to find relevant aspects. This means that a user is not given credit for finding an aspect that has already been discussed in a previously saved document.

We made some alterations to the design. One of the main differences between our study and the user studies performed in the interactive tracks is that we issued a query for

the user. This prevented the user from reformulating the query as they learned more about the topic, which forced them to use a static organization of the documents. In the control system, we present the user with the ranked list obtained when the query was issued. For the experimental system, a single hierarchy was generated and the users were able to use both the hierarchy and the ranked list to find relevant aspects of the topic. The parameter settings we used to create the hierarchy are as follows:

- *All Words* – false
- *Excluding Higher Level Words* – true
- *Enforcing Domination* – true
- *Topic Model* – Unbiased Topic Model
- *Word Type* – Single Words and Phrases

This is not the combination of parameters that performed best according to our measures in Chapter 6; however, it is the setting that seemed to be the easiest for human interpretation when the hierarchies were qualitatively analyzed. By eliminating the retrieval from the search and providing a single type of hierarchy, we felt that there would be less variance in the results, since each user would have access to the same configuration of the documents.

Another difference between our study and the TREC study was in the types of documents the users were asked to judge. In the TREC studies, the retrieval was performed on a collection of news documents; we used web documents instead. We wanted to test the snippet hierarchies in a user study because our intuition on the best parameter settings was not confirmed by the offline evaluation, whereas our intuition was confirmed in the case of the full text hierarchies. To make the task of judging relevant instances easier, we asked users not to follow any links from the pages and only to include a document as relevant if a relevant instance was mentioned on that specific page. Had the users been allowed to follow links, our evaluators would have also had to access the pages linked to the retrieved

page in order to find relevant instances. Because a page might be retrieved because of a link of the page, this frustrated some of the users.

We were also unable to utilize any of the TREC Topics used in the TREC Interactive Track because there are discussions of these topics in papers published on the web by computer scientists, which skews the results of the query. When we submitted these topics to the Google search engine, a majority of the top ten documents retrieved discussed performing studies using the query. Instead we created ten queries, which asked similar questions to the queries used in the TREC studies. These queries are discussed in Section 7.1.1 along with some statistical information about the queries.

The study was organized in the following way:

1. First the user was asked to sign a form that briefly described the study and informed her that she would not benefit from the study in any way apart from the small monetary sum that she would receive.

2. The user was then asked to fill out a questionnaire to assess her background.

3. She then read a description of the type of tasks she would be asked to perform.

4. That was followed by a worksheet which asked how she would go about completing an information retrieval task and how she would determine that she was finished.

5. After this the experimenter guided the user through a tutorial to introduce the user interface. The interfaces were exactly the same for the control and experimental systems, except that the user was given access to a popup menu that contained the hierarchy in the experimental system.

6. After completing the tutorial, the user worked on 5 information tasks. For each task, she was given 15 minutes to find relevant instances, and after each task the user was asked to fill out a short survey about the topic.

7. Then the experimenter and user went through the tutorial for the second system, and the user again completed 5 information tasks, each taking 15 minutes.

8. At the end of the session, the user completed an Exit survey and informally discussed the experience with the experimenter.

Following the discussion of the queries, we discuss the users who participated in the study in Section 7.1.2.

7.1.1 Topics in the Study

The following is the list of topics used in the study and how they were described to the users. For each topic, the users were given a title, a description which asks a question that the users tried to answer, a narrative which gives more details about the topic, and a definition of valid aspects to be found for that particular topic. In the discussion, we also include the query that was issued to the Google search engine to gather the documents. The topics were designed so that multiple documents would be required to locate all aspects of the topic.

Topic 1 Accounting Fraud

Query Accounting Fraud

Description Which corporations have been accused or found guilty of accounting fraud?

Narrative A relevant document is any report that identifies a corporation who is accused or found guilty of accounting fraud. A specific corporation must be identified for a document to be relevant; generalities are not relevant.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT corporation of the sort described above. If one document discusses several such corporations, then you need not save other documents that repeat those aspects, since your goal is to identify different corporations of the sort described above.

Topic 2 Grape Varietals

Query grape varietals produce wine

Description Which grape varietals are used to produce wine?

Narrative To be relevant, a document must identify one or more grape varietals that are used to produce wine. A reference to a grape variety without any information about whether the variety is used to produce wine would not be considered relevant.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT grape variety of the sort described above. If one document discusses several such grape varietals, then you need not save other documents that repeat those aspects, since your goal is to identify different grape varietals of the sort described above.

Topic 3 Parkinson's Drug Treatment

Query Parkinson Drug Treatment

Description What drugs are being used in the treatment of Parkinson's Disease and how successful are they?

Narrative A relevant document should name a drug used in the treatment of Parkinson's Disease and also its manufacturer, and should give some indication of the drug's success or failure.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT drug of the sort described above. If one document discusses several such drugs, then you need not save other documents that repeat those aspects, since your goal is to identify different drugs of the sort described above.

Topic 4 Uranium Producers

Query country produce uranium

Description What countries produce uranium?

Narrative A relevant item will specify the country, perhaps how much uranium it produces in a year. A document is not relevant if it discusses a country with uranium but makes no mention of where it came from.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT country of the sort described above. If one document discusses several such countries, then you need not save other documents that repeat those aspects, since your goal is to identify different countries of the sort described above.

Topic 5 Extinct Animals

Query extinct animal

Description Identify animals that have become extinct in the last 500 years

Narrative A relevant document will specify the name of an animal that became extinct in the last 500 years and an estimate of when the animal became extinct. If you are unable to determine when the animal became extinct, the document is not relevant.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT animal of the sort described above. If one document discusses several such animals, then you need not save other documents that repeat those aspects, since your goal is to identify different animals of the sort described above.

Topic 6 Active Volcanoes

Query Active Volcanoes

Description Where are volcanoes located that have erupted in the past five years or that scientists are watching because they believe an eruption is imminent (geologically speaking), and what is the volcano's name?

Narrative A relevant document will identify a volcano that has erupted in the past five years, or a volcano that is likely to erupt soon, and specify where it is located.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT volcano of the sort described above. If one document discusses several such volcanoes, then you need not save other documents that repeat those aspects, since your goal is to identify different volcanoes of the sort described above.

Topic 7 Cuba, cigars, imports

Query Cuba cigars imports

Description What countries import Cuban cigars?

Narrative A relevant document will identify a country that imports Cuban cigars.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT country of the sort described above. If one document discusses several such countries, then you need not save other documents that repeat those aspects, since your goal is to identify different countries of the sort described above.

Topic 8 Countries with Monarchies

Query country monarchy

Description Which countries have a monarch?

Narrative A relevant document will identify a country who has a king or queen. This person does not have to have any political power, but must have some function in government. The document should identify the monarch by name.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT country of the sort described above. If one document discusses several such

countries, then you need not save other documents that repeat those aspects, since your goal is to identify different countries of the sort described above.

Topic 9 Tourism, decrease

Query Tourism decrease

Description What countries have experienced a decrease in tourism in the last 2 years?

Narrative A relevant document will identify a country that has experienced a decrease in tourism in the last 2 years.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT country of the sort described above. If one document discusses several such countries, then you need not save other documents that repeat those aspects, since your goal is to identify different countries of the sort described above.

Topic 10 International Space Station

Query International Space Station

Description What countries have contributed to the international space station and what have they contributed?

Narrative A relevant document must contain both the name a country that has participated in the construction of the International Space Station as well as what their contribution is.

Aspects Please save at least one RELEVANT document that identifies EACH DIFFERENT contribution and country of the sort described above. If one document discusses several such contributions, then you need not save other documents that repeat those aspects, since your goal is to identify different contributions of the sort described above.

Before the study, each query was used to retrieve documents from the Google search engine and all the retrieved pages were downloaded. Some of the documents were removed because relevant information was no longer located on the page, or the page was no longer available. All pages were judged with respect to the query in order to achieve accurate aspectual recall data. Nine judges were used for the ten topics[1]. Table 7.1 summarizes information about the size of the retrieved set and relevant data.

Topic	Docs. Retrieved	Relevant Docs.	Relevant Instances
1	579	231	86
2	780	692	440
3	749	372	116
4	711	238	67
5	779	25	45
6	738	184	164
7	602	129	39
8	787	484	50
9	819	64	37
10	758	196	17

Table 7.1. Table contains retrieval statistics about each topic: the number of documents retrieved for the query, the number of relevant documents, and the number of relevant aspects.

7.1.2 Participants in the Study

For this study, we had 12 participants. Relative to the TREC studies, this is equal to the average number of people to perform a user study, and we asked the users to perform more tasks than were conducted in these other studies, which included between 6 and 8 topics. We solicited users through posters hung in the University of Massachusetts' library, English department, and Business School. Four of the participants had obtained their bachelors. The other eight were in their sophomore, junior, or senior years at the University of Massachusetts, majoring in a variety of fields in the humanities, social sciences, and

[1]The same person judged topic 6 and topic 8.

sciences. Four of the participants were female and six were of Asian decent. The participants said they had been performing searches for themselves or others for an average of 6.3 years. The least experienced searcher had been searching for the past year, while the most experienced had been searching for the past ten years. None of the users had participated in a user study before.

Table 7.2 contains the users' average familiarity with each topic and the average satisfaction with the amount of relevant information discovered during the session. From this data, we can discern that users generally did not feel very knowledgeable about the topics. By comparing the satisfaction with the amount of relevant information available, there seems to be little correlation between how much was available and how the users felt about the results of the task.

Topic	Avg. Familiarity	Avg. Satisfaction
1	2.58	3.83
2	1.83	3.75
3	1.67	3.33
4	2.08	3.33
5	2.33	2.33
6	2.42	3.33
7	1.83	2.33
8	2.25	3.33
9	2	3.33
10	1.92	3.33

Table 7.2. Users' perspective on the topics: how familiar they were with the topic and how satisfied they were with the amount of relevant information they found. They were asked to circle a number between 1 and 5, 1 indicating "Not at All" and 5 indicating "Extremely". The number 3 indicates "Somewhat".

7.2 Example Task

In this section we present an example illustrating how users interacted with the hierarchy and ranked list. For this example we use the second topic "Grape Varietals". Figure 7.1 shows the first page of the interface. This page was shown to all users regardless of which

system they were using. Before pressing the "Begin" button, users were expected to familiarize themselves with the topic. Figure 7.2 shows the page that appears after the "Begin" button is pressed. If the user was using the experimental system, as is shown in Figure 7.2, the hierarchy would immediately appear in the top part of the window and the ranked list would appear in the bottom part of the window. The query information is still available in a separate window. At this time, a clock begins to count down from 15 minutes, so that users are aware of how much time they have left to find relevant information.

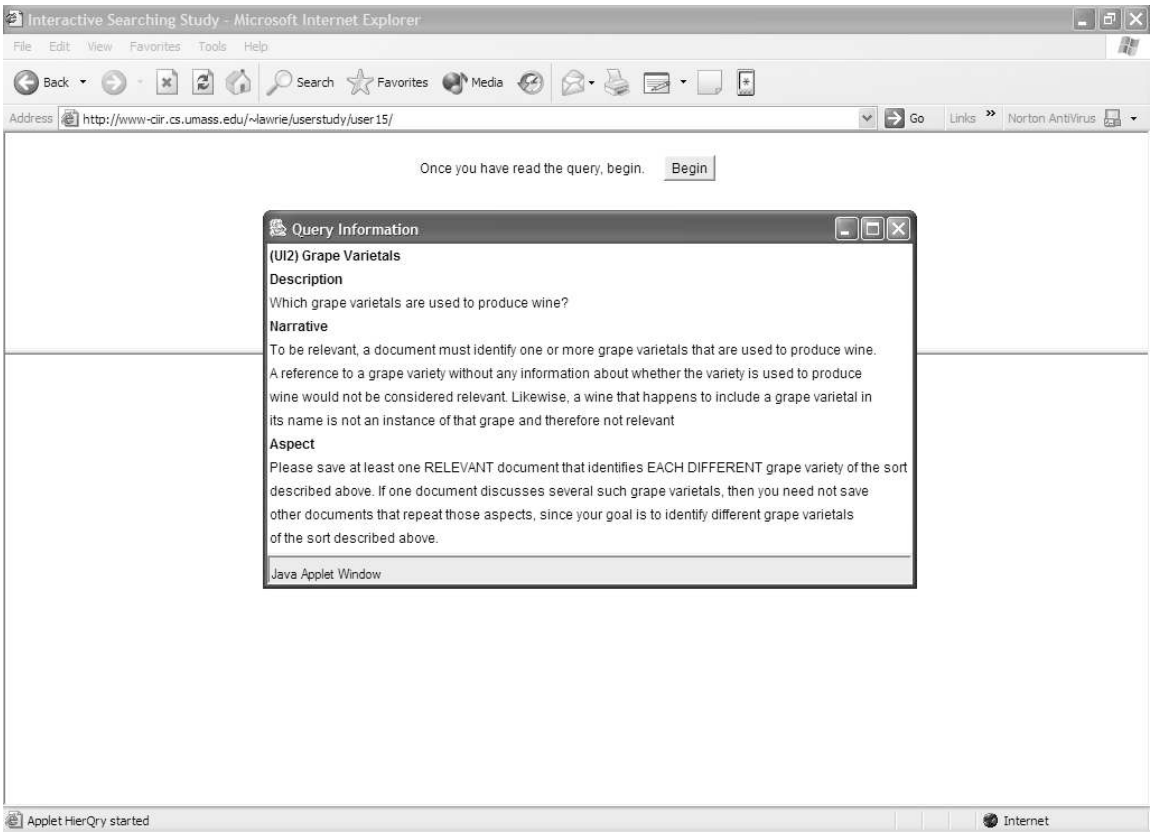


Figure 7.1. Displays the beginning page for each task and the "Query Information" window which describes the topic used as the example task.

For this example, we will use the interaction recorded for one of the users. After twenty seconds of studying the hierarchy, the user selected the topic "varietals→produce wines→premium varietals" shown in Figure 7.3. This topic has 2 related documents. The

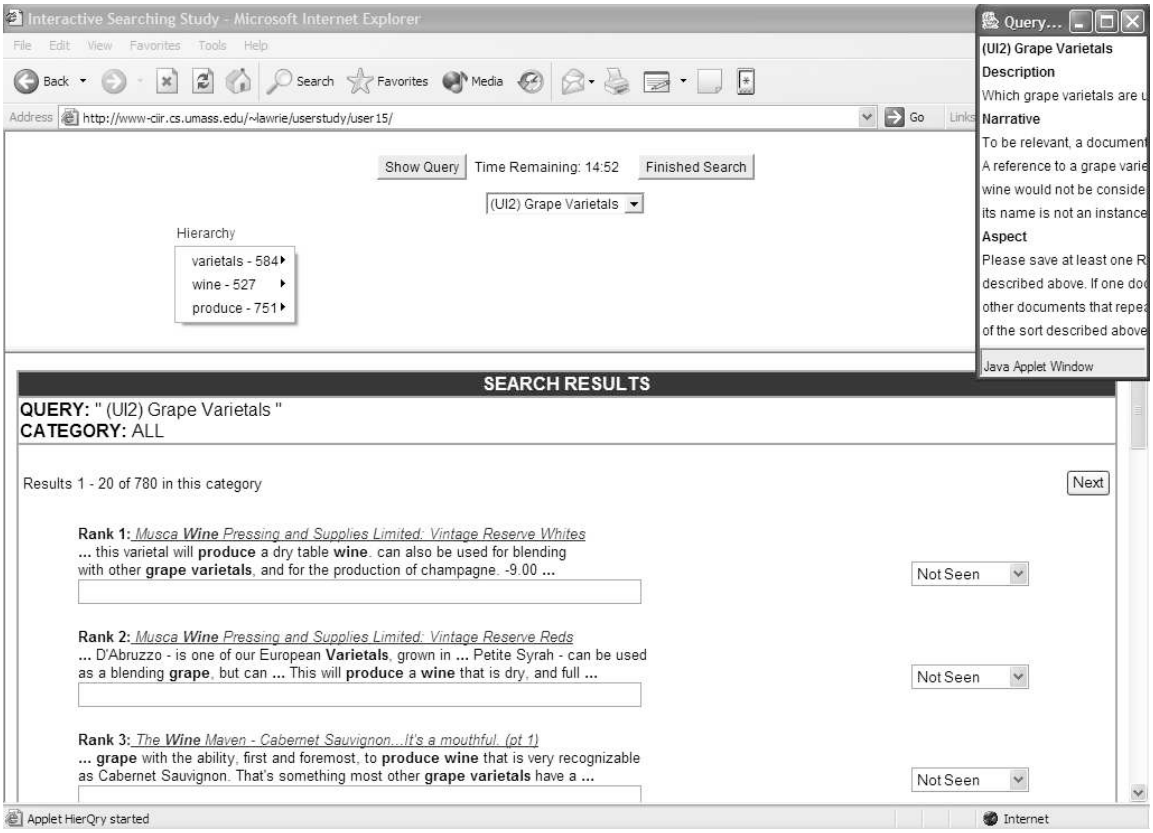


Figure 7.2. Displays the screen as it looks once all the information has been loaded. In the upper part of the window, the top level topics of the hierarchy appear. The title of the topic is also present, along with a clock that displays the amount of time remaining to work on this particular topic. In the bottom part of the window, the ranked list is displayed. The "Query Information" window is also visible. The box under each document is provided to the user for annotation purposes. The pull down menu to the right of the document description allows the user to judge the document with respect to the topic.

ranked list is also shown in Figure 7.3. The user looks at the first document for two minutes and then judges the document as "not relevant" as shown in Figure 7.4. Then he looks at the second document for slightly over one minute and judges this document to be relevant. Before going to the next topic, the user presses the "Save" button which turns the two judged documents gray, also shown in Figure 7.5.

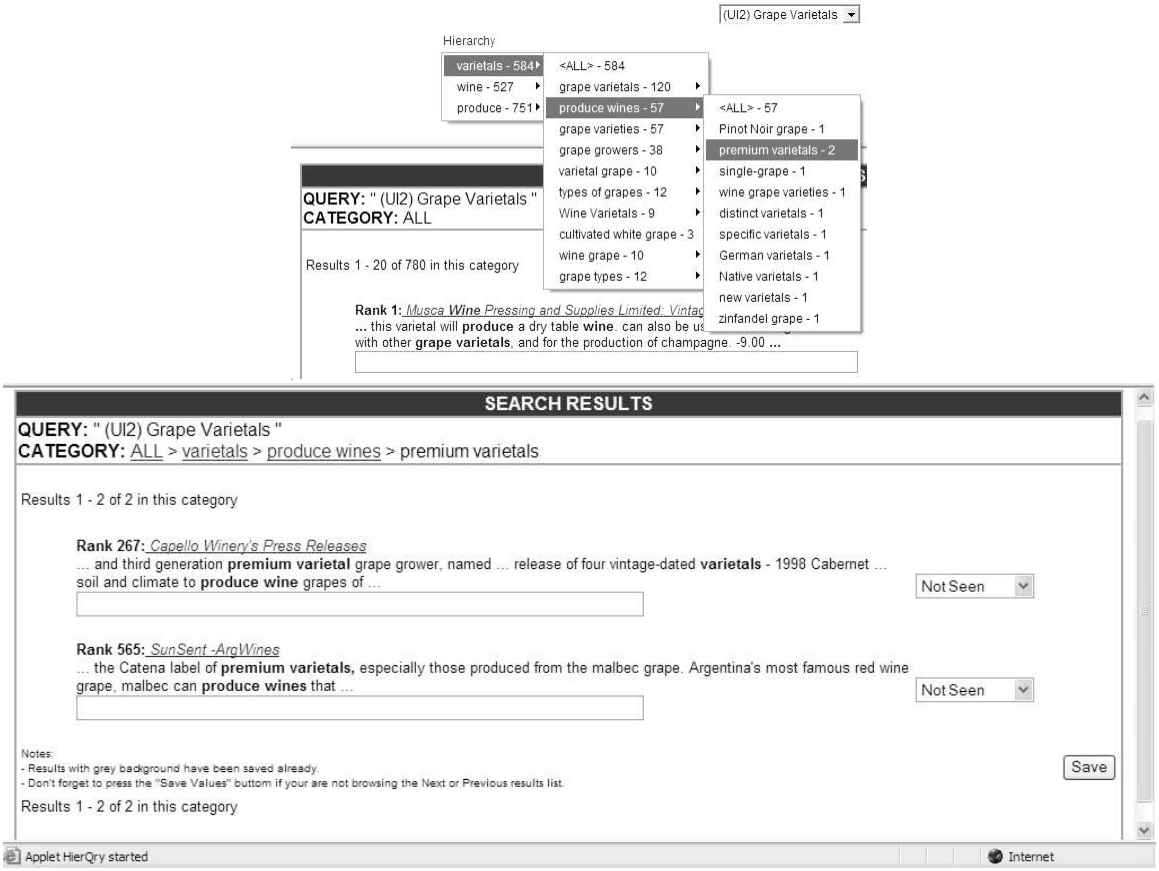


Figure 7.3. The upper picture shows the hierarchy with the first user's selection highlighted. The lower picture shows the ranked list of documents for this topic.

Now the user reexamines the hierarchy and selects the topic "varietals→types of grapes" shown in Figure 7.6. This topic includes 12 documents. The user spends four and a half minutes examining 6 of these documents. If any judgments were made, the results are lost because the user forgets to press the "save" button. Several other topics are examined: "varietals→grape types", "produce→white grape varieties", "varietals→wine

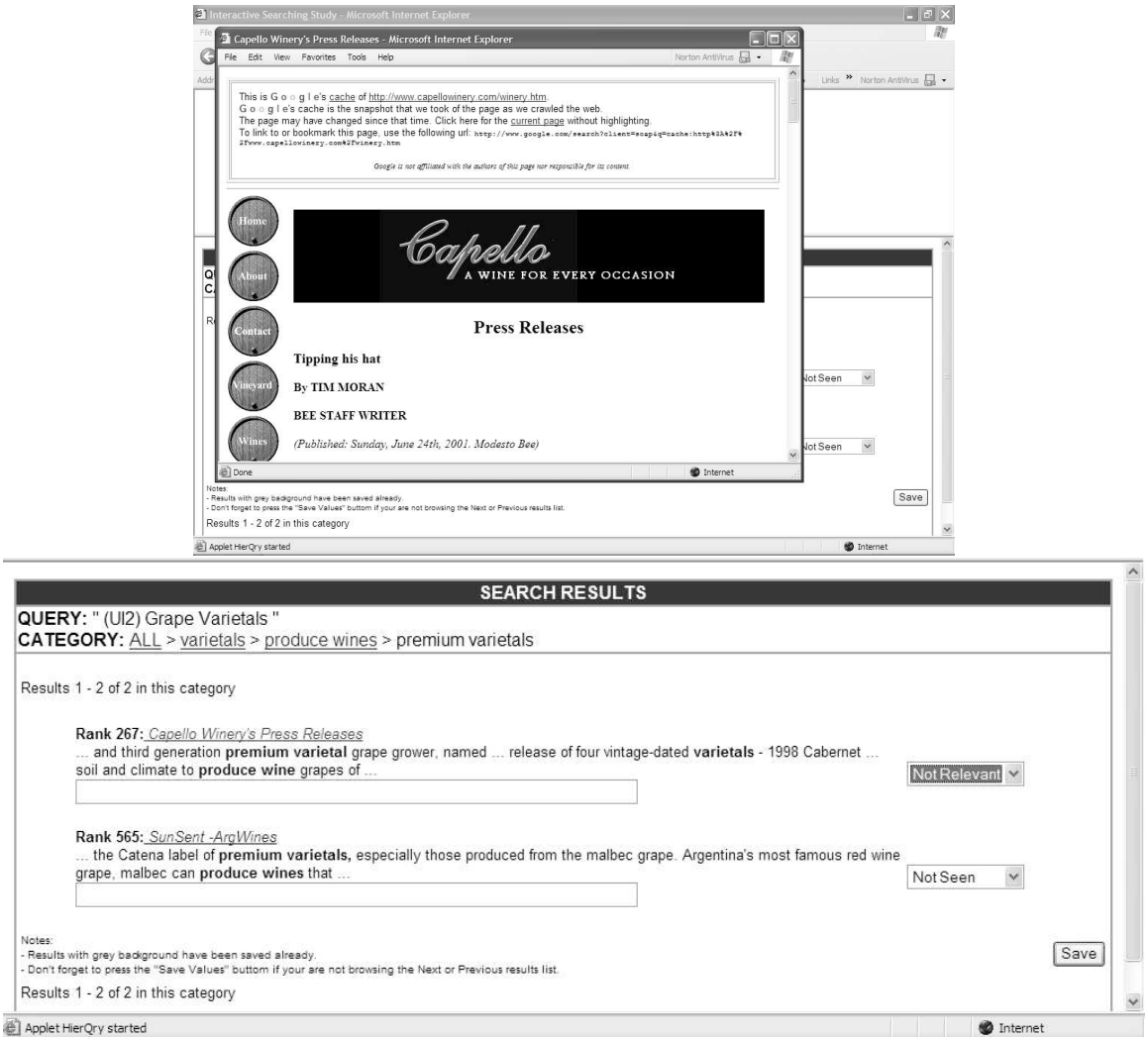


Figure 7.4. The upper picture shows the document "Capello's Winery Press Release" in a separate popup window on top of the hierarchy window. The lower picture shows that the user judges this document as not relevant, which means it does not contain any grape varietals used to produce wine.

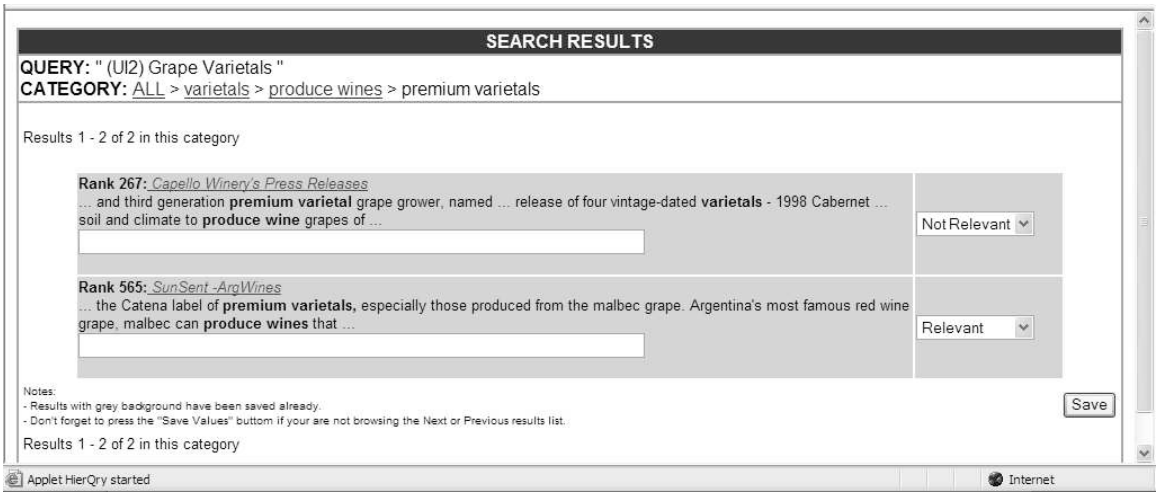


Figure 7.5. Illustrates that the documents have been judged and saved because they are outlined in gray.

grape→noble varietals", and "varietals→Wine Varietals". These portions of the hierarchy appear in Figure 7.7. For all topics except "produce→white grape varieties", the first-ranked document under the topic is examined. Only the final document under "varietals→ Wine Varietals" is judged relevant. It is likely that the user also forgot to save the judgments for the other two documents. Then the user ends work on the task by pressing the "Finished Search" button. A new page appears as shown in Figure 7.8 reminding the user to save the last page he is working on and fill out the task questionnaire. The user's judgments of this last page are saved, likely because of the final reminder. After filling in the questionnaire, the user would continue onto the next task by pressing the "Continue Now" button. If for some reason the user had to take an exceptionally long break, during which the study could not be kept running, the user would be able to press the "Continue Later" button. When the user reloaded the application, he would be able to continue with the next task.

In total the user spends 13 minutes 41 seconds searching for different grape varietals. In the two documents that were judged relevant, he finds 26 relevant instances. In the eleven documents that the user viewed, he finds 35 relevant instances. Ten of these documents

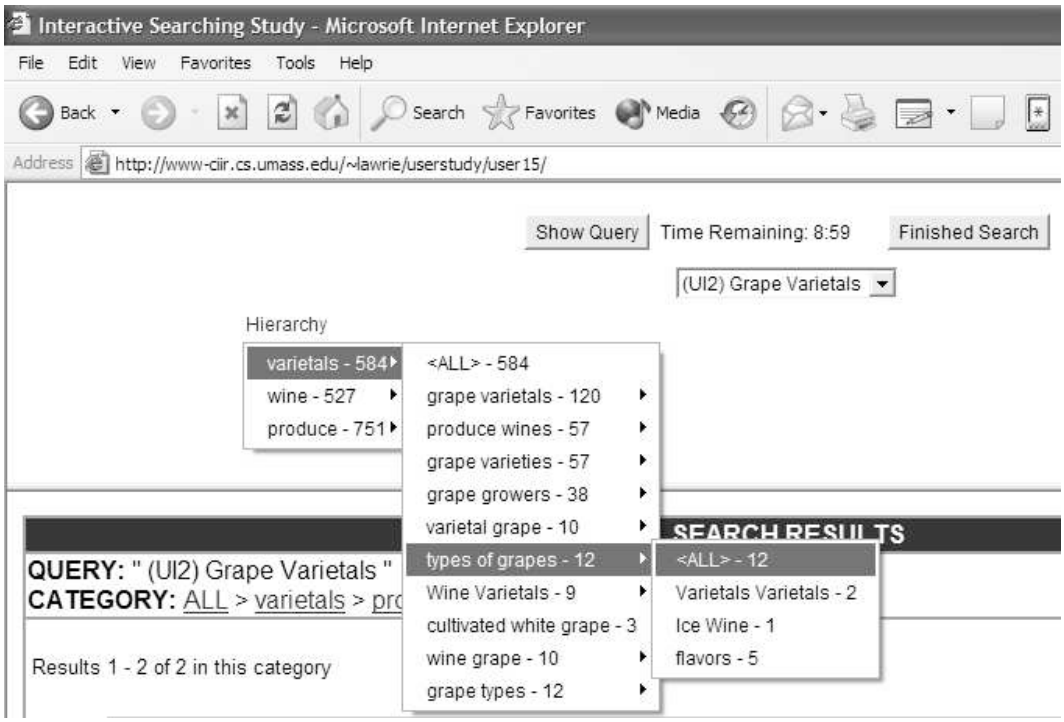


Figure 7.6. Shows the next topic explored by the user.

were judged relevant by the assessor for the topic, including the one document the user judged not relevant.

7.3 Statistical Analysis

From the study, we gathered aspectual recall and precision data. Aspectual recall is the fraction of aspects found in documents discovered by the user out of the total unique aspects found in all documents. Precision is the fraction of documents which contain at least one relevant aspect, or in other words that fraction of the documents which are relevant. We calculated two recall values and two precision values. One is the recall and precision of the documents that a user viewed. The other is the recall and precision of the documents that the user judged to be relevant. We examine the set of viewed documents to see if users are able to use the hierarchy to eliminate some the non-relevant data available, and because

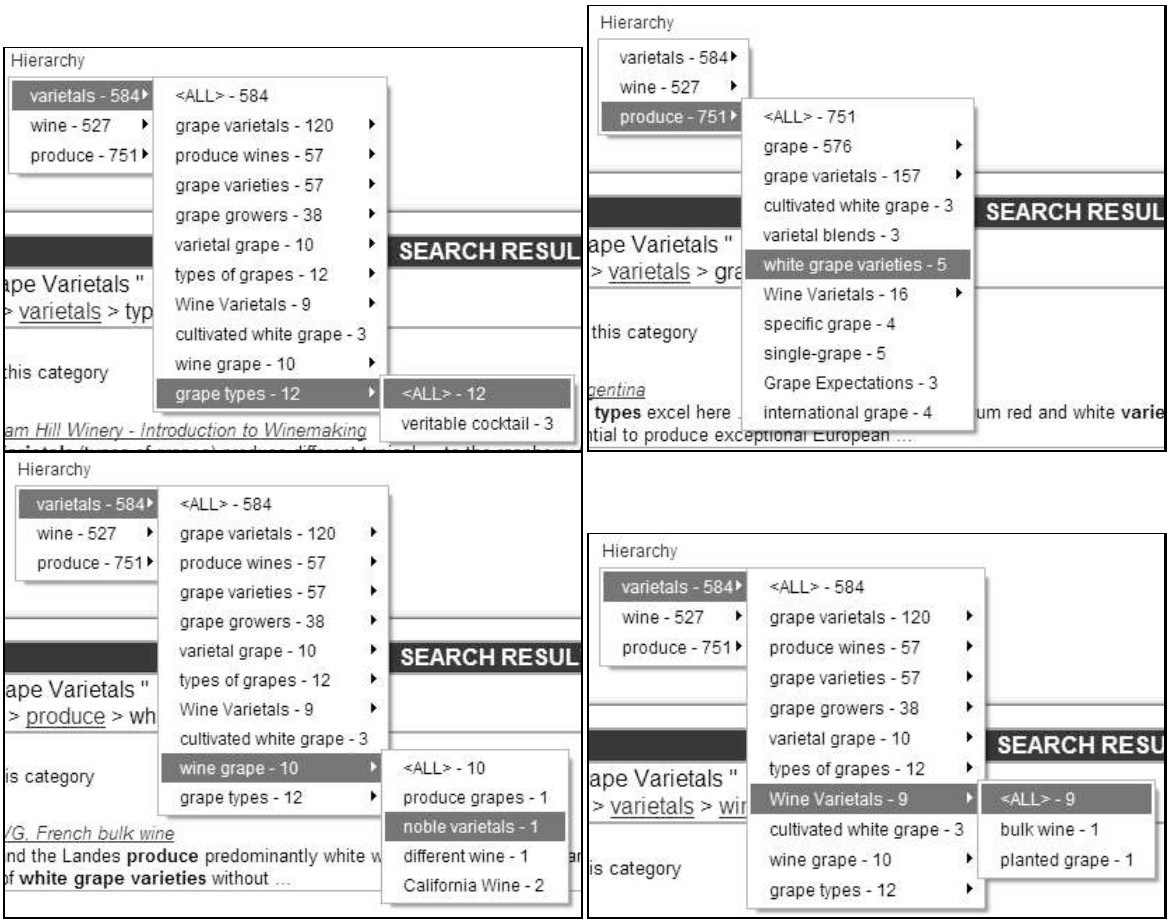


Figure 7.7. The upper left side shows the third topic selected. The upper right shows the fourth topic selected. The lower left shows the fifth topic selected, and the lower right shows the sixth and final topic selected during this task.

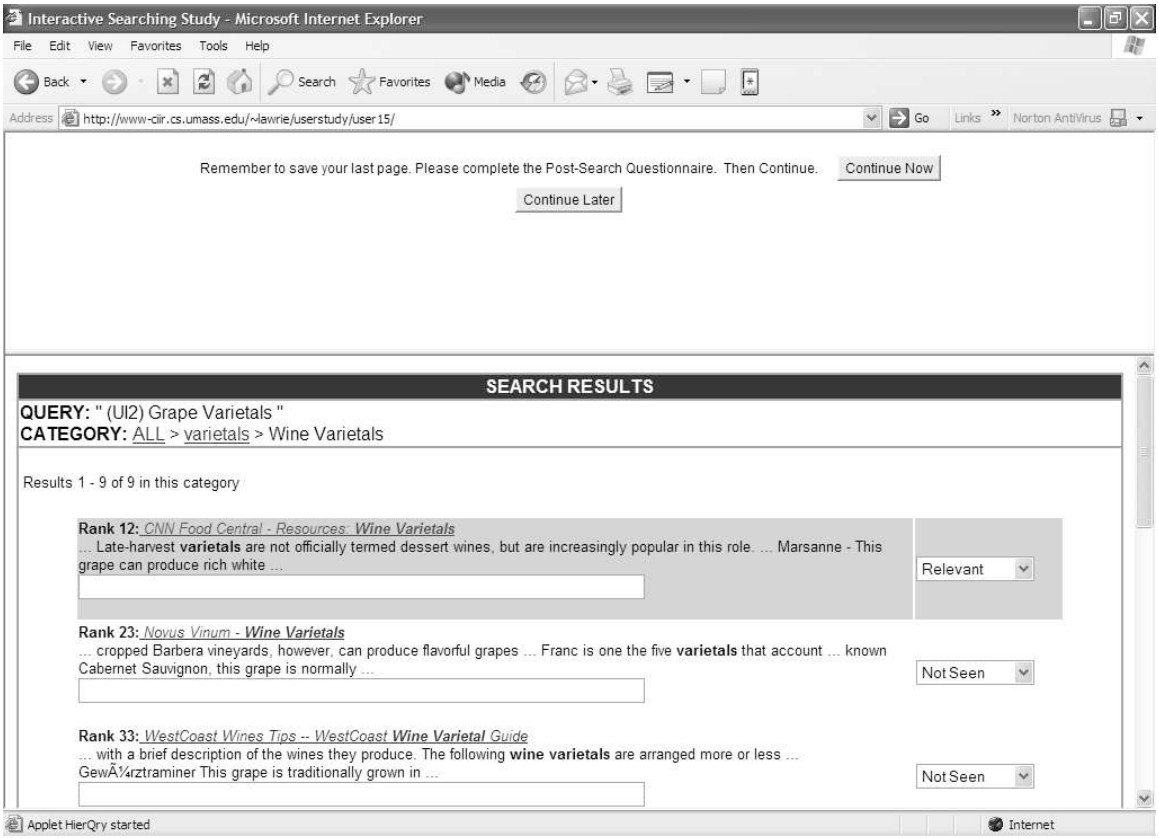


Figure 7.8. The last page shown for a particular task before the user continues with the next task.

we suspect that some users forgot to save the judgments of some of the documents they viewed.

To analyze the data, we compare the mean of precision or recall on the hierarchy to the mean performance on the ranked list, and then use a paired t-test to ascertain if the results are significant. We refer to the hierarchy as the experimental system and the ranked list as the control system.

Table 7.3 shows the results of the test. The first conclusion that can be drawn is that the users' recall was significantly higher when they used the control system when all topics are included in the mean. This may have occurred for several reasons. The first is that all the users were familiar with ranked lists and experienced with using them, so there was no learning curve for using the control system as there was when using the hierarchy. All of the users said that it was easier to learn to use the ranked list in their exit questionnaire. We tested this hypothesis by eliminating the first two topics on each system from the analysis. The results of this test appear in Table 7.4. For this test, the paired t-test shows no significant difference between the control and experimental systems. This change indicates that our hypothesis about the learning curve is most likely correct. Our tutorial did not allow the users to become sufficiently familiar with the hierarchy, and thus they were unable to use it effectively, especially in the beginning of the study.

There are other factors that may have played a role in the fact that the users performed significantly better on the control system when all topics are considered in the analysis. One possible reason is that some users did not try to understand the hierarchy. Some users, when provided with a hierarchy, never selected any of the topics. In Figure 7.9 it is obvious that a few users (C, W, and Z) were never able to use the hierarchy successfully, since these users generally discovered no relevant documents when using the hierarchy, but did find relevant documents when using the ranked list. Besides lack of effort on the part of the user, this preference for the ranked list may be attributed to the fact that the design of the user interface for the hierarchy was not as polished as the design of the ranked list, which has

had several years to evolve into the current format found as the output of search engines. There may also have been a problem with the design of the study. A hierarchy is most helpful in a situation where the user knows something about the topic and is interested in investigating particular details. From the data in Table 7.2 it is evident that the participants in the study were not very familiar with the topics. It should also be noted that no user had a higher average recall on the experimental system than on the control system when all topics were included in the mean value.

Type	Lower CI Mean	Mean (E-C)	Upper CI Mean	Lower CI Std. Dev.	Std. Dev (E-C)	Upper CI Std Dev.
Precision – VD	-0.13	0.0377	0.2058	0.1874	0.2645	0.449
Recall – VD	-0.097	-0.075	-0.052	0.0252	0.0355	0.0603
Precision – JD	-0.118	0.0392	0.1968	0.1757	0.2481	0.4212
Recall – JD	-0.117	-0.091	-0.065	0.0291	0.0411	0.0698

Type	t Value	Pr >|t|
Precision – VD	0.49	0.6310
Recall – VD	-7.29	< 0.0001
Precision – JD	0.55	0.5953
Recall – JD	-7.69	< 0.0001

Table 7.3. Paired T-Test of the Experimental System (E) against the Control System (C). The Mean (E-C) compares the mean performance value on the experimental system to the mean performance value on the control system. In cases where the mean is negative the control system performed better. In cases where the mean is positive the experimental system performed better. The "Pr $> |t|$" value indicates the probability that the results occurred randomly. Values less than 0.05 are considered statistically significant. Types followed by "VD" are calculated using the documents that the user viewed during the session. Types followed by "JD" are calculated using the documents the user judged as relevant. Some users judged documents from reading the summaries alone, without ever viewing the document. Some documents were viewed without ever being judged.

The precision tests in Table 7.3 show that users did achieve better precision scores when using the hierarchy, both when selecting documents to view and when judging the documents. Unfortunately these values are not statistically significant, so this study does not provide any conclusive results with respect to precision. Achieving higher precision indicates that the users were able to use the hierarchy to focus on the relevant portions of

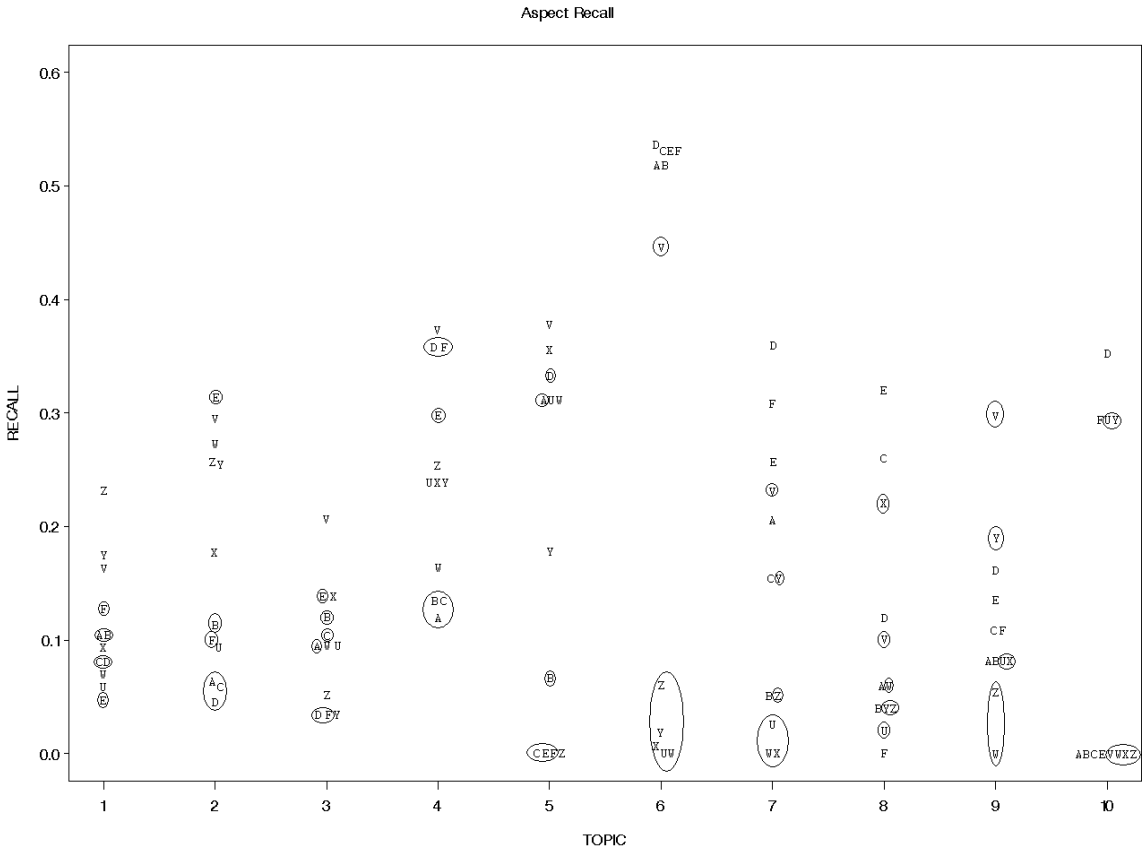


Figure 7.9. Illustrates the recall values users achieved. Each letter represents a user. Users A-F used the experimental system for topics 1 to 5 and the control system for topics 6 to 10. Users U-Z used the control system for topics 1 to 5 and the experimental system for topics 6 to 10. Users using the experimental system are circled.

Type	Lower CI Mean	Mean (E-C)	Upper CI Mean	Lower CI Std. Dev.	Std. Dev (E-C)	Upper CI Std Dev.
Precision – VD	-0.141	0.0434	0.2273	0.205	0.2894	0.4914
Recall – VD	-0.115	-0.056	0.0028	0.0658	0.0929	0.1578
Precision – JD	-0.205	0.0158	0.2361	0.2456	0.3467	0.5887
Recall – JD	-0.13	-0.039	0.0513	0.101	0.1426	0.242

Type	t Value	Pr $>\|t\|$
Precision – VD	0.52	0.6141
Recall – VD	-2.10	0.0599
Precision – JD	0.16	0.8776
Recall – JD	-0.95	0.3609

Table 7.4. Paired T-Test of the Experimental System (E) against the Control System (C) using the last three topics in the study for each system. This test tries to eliminate differences that might have been caused by users learning the system.

the ranked list, which means they were able to exclude non-relevant information. The fact that precision is somewhat better when using the hierarchy may also indicate that the users spent less time looking at the documents than they did when only presented with the ranked list, due to time spent looking at the hierarchy. Figure 7.10 shows that users were using the hierarchy in most cases where they achieved 100% precision. This data also revealed that some users have very poor precision when using the hierarchy, which indicates that the ability to use the hierarchy as a tool to improve precision varied greatly among the participants, which is why the results of precision are not significant.

7.4 User Comments

The user's general impression of the hierarchy was positive. One of the most interesting outcomes of the exit interview was that 11 of the 12 participants stated that they liked using the hierarchy better than the ranked list alone. The preference was fairly strong (an average of 3.64 on a point scale of 5); however, many users did not feel that they were able to use the hierarchy successfully. One said that he liked the hierarchies but he did not think they helped him accomplish his task, while a few others said that they did feel the hierarchy improved their search. Another participant stated that he felt that he was more successful

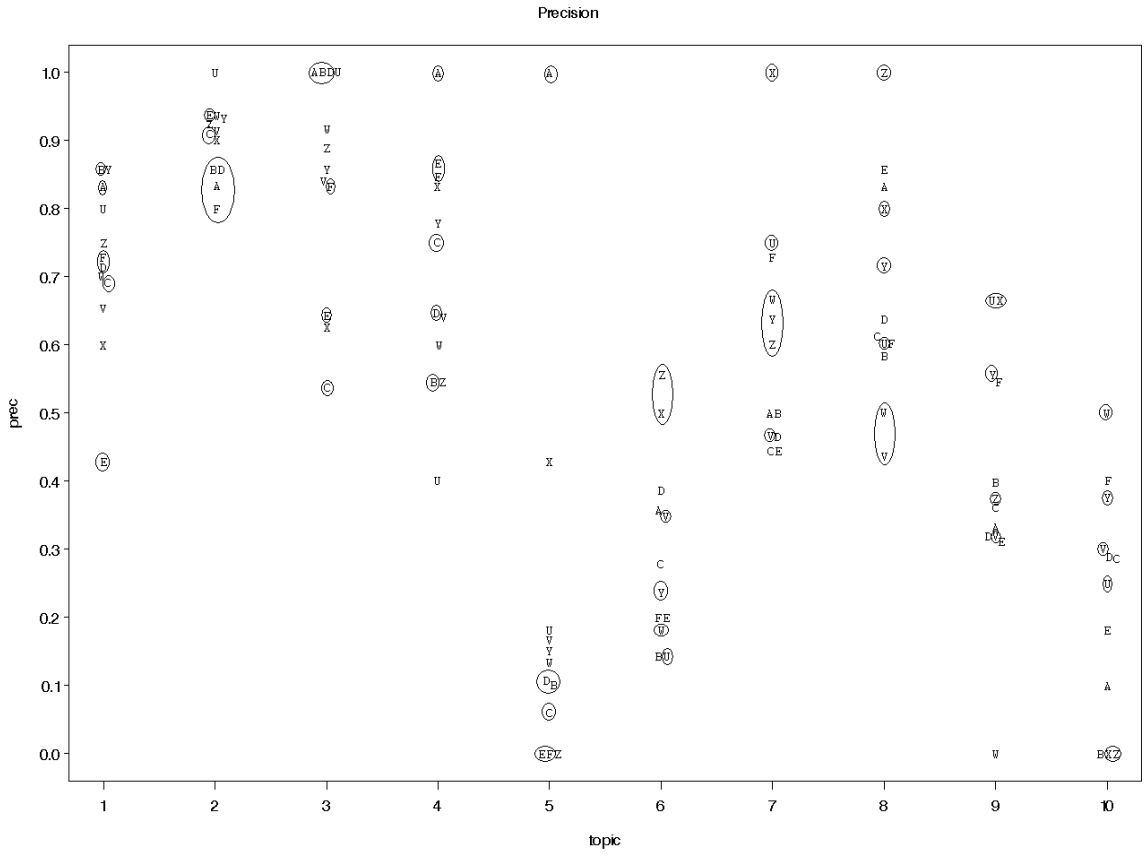


Figure 7.10. Illustrates the precision scores users achieved. Each letter represents a user. Users A-F used the experimental system for topics 1 to 5 and the control system for topics 6 to 10. Users U-Z used the control system for topics 1 to 5 and the experimental system for topics 6 to 10. Users using the experimental system are circled.

using the hierarchies than without. The one person who preferred the ranked list said that there was too much information in the hierarchy for a general search. She was just as happy to look at the top ranked documents. However, had she been more knowledgeable about the topic and wanted specific information, she thought the hierarchy would be useful. In fact, she was very excited about the prospect of using the hierarchy to hone in on particular aspects of subjects about which she had a lot of knowledge. Another user wrote, "The hierarchy was helpful but was vague and sometimes misleading." This person said that he did prefer the hierarchy, but his preference was the smallest of all users, assigning it a 2, which is between "Not at all" and "Somewhat". He also chose not to use the hierarchy on the ninth topic, although he had access to it.

The results of the study, which found that users had better recall using the ranked list when analyzing all topics and statistically insignificant success in terms of precision, may have been due to the fact that users had insufficient time to familiarize themselves with the new tool. One participant wrote, "The hierarchy system was a little bit more difficult to jump into, but was much faster once I got going. The ranks within the hierarchies helped narrow down potential unuseful sites. Hierarchies made it much easier to narrow down the question." From this we can hypothesize that a user's first few searches would not have gone as well as subsequent searches, when the user was more familiar with the tool. Table 7.3 supports this and we can observe in Figure 7.9 that after the third query the top four performances of that query include multiple users who were using the hierarchy.

As can be expected, some users liked the hierarchy while others did not, but one important outcome of the study is that some of the users have begun using the hierarchy for some of their own searching tasks. This indicates that these users feel the hierarchy does reveal interesting information about the documents that cannot be accessed from the standard ranked list. Of course, this is merely anecdotal evidence that the hierarchy provides information in which a user is interested.

7.5 Discussion

When the user study was designed, it was believed that the hierarchy would be most useful in situations where the user was interested in many different aspects of a particular topic. The type of study conducted seemed well suited to evaluate the hierarchy because it asks users to find all aspects of each topic. The hierarchies provide a high-level representation of the document set, and the users expressed their preference for using this global summarization of the information. However, the users were not able to effectively interpret the hierarchy to locate relevant information. We believe this was because the users were unable to understand topic descriptions in the hierarchies, since they lacked detailed knowledge of the topics. In the end it turned out that this was not a particularly good test of the hierarchies because of this lack of knowledge.

The type of situation where the hierarchy is most useful occurs when a user is interested in specific details about a topic which are difficult to express in a few words. These are topics of which a user already has some general knowledge, and is interested in learning about particular aspects of that topic. In this case, the user will be able to understand what the words and phrases appearing in the hierarchy mean and thus be able to navigate to the specific documents that discuss these details.

One problem with analyzing the hierarchy is that we expect different people will use it in different ways, and that it will not be suitable for all of the searches that a person performs. This means that one single study will not adequately measure the usefulness of the hierarchy. Other studies could be designed to test different uses of the hierarchy. One modification to the present task could be to ask the user to find unusual aspects relevant to a specific topic rather than all aspects relevant to a specific topic. Unusual aspects might be defined as those aspects not found within the top 20 or 40 documents. Another task could look at using the hierarchy for an exploratory task. Such a task might require users to gather information from several different sources to complete the task, and perhaps would measure time spent to achieve a successful completion. Alternatively, one could try to

measure the ability of the hierarchy to summarize a particular document set. Such a study might give a user either a ranked list or a hierarchy. After allowing the user to examine the document set, she would be required to answer questions to ascertain how much the user was able to learn.

7.6 Conclusion

The user study reveals that users prefer using the hierarchy to a ranked list. This preference indicates that users like being able to choose which parts of the retrieved set that they will look at. The fact that precision is sometimes higher when using the hierarchy indicates that at least some of the users were able to use the hierarchy to focus on information that is considered relevant.

CHAPTER 8

CONCLUSION

In the current information age, people have access to a vast amount of information. The state of the art technology provides users with the ability to issue a query to enormous databases and receive in return a list of documents from the search engine ranked with respect to the user's query. This fixed form of results allows limited access to the documents because only one ordering of the documents is available.

The work presented in this paper addresses this problem by automatically generating a hierarchical, word-based summary of the documents. This allows users to navigate to the most interesting portions of the ranked list, regardless of the rankings assigned by the search engine. In order to create the hierarchical summary, we expressed the information contained in the documents using language models, and used statistical techniques to choose topic words to form the summary and constructed the hierarchical structure. We then qualitatively evaluated the hierarchy in terms of how understandable the hierarchy is and how well it allows a user to access the documents. We also included a detailed description of the software used to generate hierarchical summaries, and analyzed the software with respect to efficiency. Finally, we evaluated the hierarchies using non user-based evaluations in addition to a user study. From these evaluations, we concluded that the probabilistic hierarchy performs as well as and sometimes better than previous heuristic techniques used to create topic-oriented hierarchies. We also showed that the words chosen to be part of the summary are better summarization words than the top TF.IDF weighted words. Finally, we find that users generally like having access to the hierarchy, and that it may improve their ability to locate relevant documents.

In the following sections, we review the research contributions of this work and present ideas for future research.

8.1 Research Contributions

One of the challenging problems faced in research that attempts to automatically summarize documents is that people often disagree about what information should be part of a summary. Because of this disagreement, it is difficult to determine if an automatically-generated summary adequately describes the information. Hierarchical summarization is faced with a larger problem because of the human resources required to construct such summaries for large document sets. This means that the human-built hierarchies available for comparison are all general purpose hierarchies which try to inform the user only about the high level concepts.

Given that the detailed type of hierarchies we would like to generate are currently not being created by humans, we have developed statistical techniques to create detailed hierarchies and have focused on methods for evaluating the hierarchies to determine their usefulness. This work makes the following three general contributions:

- We provide a formal framework for creating hierarchical summaries.
- We develop a methodology for using language models as an abstraction of documents in order to create a hierarchical, word-based summary that can be used to navigate a collection of documents.
- We develop non user-based evaluation measures for hierarchical summarization using large document collections. In order to develop human-usable hierarchical summaries, the current methods of evaluation are expanded to encompass a broader range of attributes. These measures allow us to verify the best parameter settings for the hierarchy without involving a user study.

We discuss each of these points in more detail in the following subsections.

8.1.1 Formal Framework

The formal framework developed in this dissertation relies on two attributes to generate a hierarchical summary: topicality and predictiveness. By combining these attributes, we can select words from the vocabulary used in the documents which describe aspects of the documents. These attributes also provide a mechanism for organizing the words into a hierarchical structure.

8.1.2 Application of Language Modeling

The approach to hierarchical summarization developed in this dissertation utilizes language modeling as an abstraction of the documents. Using this description of the document set enables us to develop efficient algorithms for estimating topicality and predictiveness. These models also provide a principled means of modifying the estimates.

8.1.3 Non User-Based Evaluations

Developing mechanisms for evaluating the hierarchical summaries is one of the most challenging aspects of this research. As our user study illustrates, it is challenging to develop a study that accurately models user behavior. For this reason, methods of evaluating hierarchies without users and in an offline setting are extremely important. We have developed three such evaluations which measure how well the words in the hierarchy summarize the document set, how well a user would be able to access relevant documents, and how a user can access all the documents that are found in the hierarchy. These evaluations allowed use to discover the differences between full text and snippet hierarchies, and that we made need to develop different estimations in order to improve the snippet hierarchy.

8.2 Future Work

8.2.1 Improving the Hierarchy

In this dissertation we presented a probabilistic framework for finding topic words by combining the probabilities of topicality and predictiveness. These values are estimated, and there may be several ways to improve the estimates.

One such improvement could be to incorporate a confidence factor devoting the likelihood that these estimated values are accurate. This could be especially helpful in the estimate of predictiveness when used to determine which words should be considered as candidate subtopics. This confidence factor could be based on how much data was used to determine the estimated value. This factor could be used as an additional threshold to ensure that all subtopics included in the hierarchy surpass a minimum confidence level. The predictiveness estimate may also be improved by smoothing the bigram language model.

The topicality estimate may be improved through the use of topic models. When calculating topicality, we use a word's contribution to relative entropy. Rather than looking at the frequency of individual words in the document set versus general English, one could compute the topic model and compare this with general English using relative entropy. The concept of a topic model was discussed in Section 3.4, but not implemented because of the difficulties described in Section 3.4.3. Yet these problems do not arise if segments of the document set are not removed in iterative steps. As long as the topic model is never computed with less that the full text of the documents used to create the hierarchy, there should not be a problem with sparseness of data. Since the topicality estimate remains constant throughout the entire process, the full text will always be used. This type of calculation would incorporate the presence of words co-occurring with the topic word rather than the probability of the topic word itself. Combining both of the methods may lead to the best way of estimating topicality.

Further exploration of the segment sizes used to generate the bigram language model for predictiveness should be studied. In the examination of this parameter in Section 6.2, it

was concluded that a size of 100 words before and after the topic word is optimal. However, in this study no smaller sizes were examined. It would be interesting to determine if smaller size segments are more effective, especially in terms of summarization, where the evaluation showed that the 100 word segment size was clearly the best. It would also be interesting to determine if natural language boundaries such as sentences or paragraphs are more effective than fixed size segments.

Given the differences observed between the full text and snippet hierarchy, further analysis of these differences should be pursued. Specifically, developing another way of estimating topicality is important. The short comings of the current method may stem from the fact that we assume a normal word distribution that is present in the full text of documents. Our estimation technique may fail to take into account the fact that there are more content-bearing words in the fragments of sentences that appear in the snippets than in full text.

Another possibility is to explore the user interface design of the hierarchies. More information could be provided to the user than merely allowing them to navigate a popup menu. For example, if a user found a particular document very useful, she may also be interested in other portions of the hierarchy where that document occurs. The interface could be changed to point out all the hierarchy branches where these interesting documents are found. Another possibility would be to implement a history of visited branches, so that the user would know which branches she had already explored. Also, a more intuitive design may help to improve the results of future user studies by overcoming the learning curve inherent in unfamiliar interfaces.

Finally, it would be interesting to compare the probabilistic hierarchies to subsumption and lexical hierarchies using the snippet data. To do this, the software to produce the other types of hierarchies has to be modified to accept the web input. Then these evaluations could be conducted.

8.2.2 Other applications

One direction for further research is to develop hierarchies from less homogenous document sets. Since probabilistic hierarchies are designed to handle more heterogeneous material, but they have not been tested on such sets, it would be appropriate to construct such hierarchies. For example, a document set composed of personal collections of documents or emails could be used to generate a topic hierarchy. This type of hierarchy could be used as an effective method of organizing personal collections. Such groups of documents would likely be more heterogeneous than the sets of retrieved documents that are experimented with in this paper.

Developing hierarchies based on document summaries with more information than the snippet hierarchies would be interesting. By including more natural language in these summaries, the same parameter settings used to create the best full text hierarchies may be used. This intermediate type of summary could be composed of the most relevant passage(s) or sentence(s). Such a hierarchy could be produced nearly as rapidly as we are able to produce snippet hierarchies, but may not suffer as much from the same performance problems evident in of our offline evaluations.

Another interesting avenue of pursuit would be to further develop hierarchies in a cross-lingual environment. One of the first steps in such work should be to identify how the hierarchy can be created so that it provides a bridge between documents in a user's native language and those that are in foreign languages. Using excerpts from the documents, rather than the entire document, to create an excerpt hierarchy may allow the hierarchies generated for documents in different languages to be more similar, and thus more useful in a multilingual environment.

APPENDIX A

DOMINATING SET PROBLEM

In this appendix we establish the NP-hardness of the DOMINATING SET problem (DSP). In Section A.1 there is a proof that the decision problem: "Is there a dominating set of size $\leq K$?" is NP-Complete. In Section A.2 we discuss the optimization problem of finding the smallest dominating set, and we describe how hard approximations to the optimization problem are.

A.1 NP-Completeness of the Decision Version of DSP

PROBLEM: DOMINATING SET (DSP)

INSTANCE: Graph $G = (V, E)$, the number of vertices sought in the dominating set $K \leq |V|$.

QUESTION: Is there a dominating set of size K or less for G, i.e., a subset $V' \subseteq V$ such that $|V'| \leq K$ and such that every $v \in V - V'$ is adjacent to some $u \in V'$?

DSP is easily shown to be in NP since a nondeterministic algorithm could guess at the set V' and then check, in polynomial time, whether an edge exists between each $v_i \in V - V'$ and some $v_j \in V'$, thereby determining if V' is a dominating set.

We prove that DSP is NP-Hard via a reduction from the SATISFIABILITY problem.

PROBLEM: SATISFIABILITY (SAT)

INSTANCE: Set U of variables, collection C of clauses over U.

QUESTION: Is there a truth assignment for C at least one variable in each clause is true?

Let $U = \{u_1, u_2, ..., u_n\}$ be a set of variables and $C = \{c_1, c_2, ..., c_m\}$ a set of clauses making an arbitrary instance of SAT. From the variables and clauses, a graph is constructed

and a positive integer $K \leq |V|$ is chosen such that G has a dominating set of size K or less if and only if C is satisfiable. For each variable $u_i \in U$, graph G has a *truth setting component* $T_i = (V_i, E_i)$, with $V_i = \{u_i, \overline{u}_i\}$ and $E_i = \{\{u_i, \overline{u}_i\}\}$; T_i introduces two vertices joined by a single edge to the graph G. This enables a truth setting to be dominated by its opposite setting.

For each clause $c_j \in C$, graph G has a *satisfaction testing component* $S_j = \{V'_j\}$, consisting of l_j vertices where l_j is the number of variables in clause c_j, and no edges:

$$V'_j = \{a_1[j], a_2[j], ..., a_{l_j}[j]\}$$

The only part of the construction that depends on which literals occur in which clauses is the collection of *communication edges*. These are best viewed from the vantage point of the satisfaction testing components. For each clause $c_j \in C$, let the literals in c_j be denoted by $x_{j1}, x_{j2}, ..., x_{jl}$. Then the communication edges emanating from S_j are given by:

$$E''_j = \left\{ \begin{array}{l} \{a_1[j], x_{j1}\}, \{a_1[j], x_{j2}\}, ..., \{a_1[j], x_{jl}\}, \\ \{a_2[j], x_{j1}\}, \{a_2[j], x_{j2}\}, ..., \{a_2[j], x_{jl}\}, \\ \vdots \\ \{a_l[j], x_{j1}\}, \{a_l[j], x_{j2}\}, ..., \{a_l[j], x_{jl}\} \end{array} \right\}$$

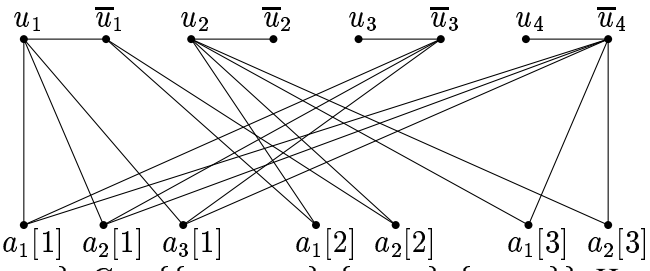
The construction of our instance of DSP is completed by setting $K = n$, the number of variables, and $G = (V, E)$ where

$$V = (\bigcup_{i=1}^{n} V_i) \cup (\bigcup_{j=1}^{m} V'_j)$$

and

$$E = (\bigcup_{i=1}^{n} E_i) \cup (\bigcup_{j=1}^{m} E''_j)$$

Figure A.1 shows an example of the graph obtained when $U = \{u_1, u_2, u_3, u_4\}$ and $C = \{\{u_1, \overline{u}_3, \overline{u}_4\}, \{\overline{u}_1, u_2\}, \{u_2, \overline{u}_4\}\}$.



$U = \{u_1, u_2, u_3, u_4\}$, $C = \{\{u_1, \overline{u}_3, \overline{u}_4\}, \{\overline{u}_1, u_2\}, \{u_2, \overline{u}_4\}\}$. Here $K = n = 4$.

Figure A.1. DSP instance resulting from SAT instance.

The construction takes time $O(n+ml^2)$ which is polynomial in the size of the instance. All that remains to be shown is that C is satisfiable if and only if G has a dominating set of size K or less.

First, suppose that $V' \subseteq V$ is a dominating set for G with $|V'| \leq K$. V' must contain at least one vertex from each T_i. Since this gives a total of at least $n = K$ vertices, V' must in fact contain $exactly$ one vertex from each T_i. Thus we can use the way in which V' intersects each truth-setting component to obtain a truth assignment: $t : U \rightarrow \{T, F\}$. We merely set $t(u_i) = T$ if $u_i \in V'$ and $t(u_i) = F$ if $\overline{u}_i \in V'$. To see that this truth assignment satisfies each clause $c_j \in C$, consider the edges in E''_j. In order for the nodes representing the literals in the clauses to be dominated (none are in the dominating set), at least one of the variables in c_j must be in the dominating set. This implies that the corresponding literal, either u_i or $\overline{u}_i$, from clause c_j is true under the truth assignment t, and hence clause c_j is satisfied by t. Because this holds for every $c_j \in C$, it follows that t is a satisfying truth assignment for C.

Conversely, suppose that $t : U \rightarrow \{T, F\}$ is a satisfying truth assignment for C. The corresponding dominating set V' includes one vertex from each T_i. The vertex from T_i in V' is u_i if $t(u_i) = T$ and is $\overline{u}_i$ if $t(u_i) = F$. This ensures that all of the vertices connected by edges from each set E''_j are dominated, because t satisfies each clause c_j. Since each

variable has a truth assignment, all the vertices representing the alternative truth setting are also dominated. Therefore we have the desired dominating set.

A.2 The Hardness of Optimization

Since the decision problem for DSP is NP-Complete, the DOMINATING SET OPTIMIZATION (DSPO) problem is NP-Hard. DSPO asks the question: What is the smallest dominating set for graph G? Researchers have also explored how hard an approximation to DSPO is. Paz and Moran defined a genre of reduction that preserves the degree of approximability[53]. Much like the reduction of Section A.1, a reduction that preserves approximability shows that the target problem is approximable whenever the source problem is. Paz and Moran refer to such a reduction as a *measure preserving reduction*.

Paz and Moran show that *measure preserving reductions* exist between NP-optimization problems that have a quality they call *simple*. An optimization problem is *simple* if for a given instance of a minimization problem and a specific k, one can decide if a solution exists whose value is $\leq k$, in polynomial time in the size of the problem instance. This can be accomplished for all positive integers k. In the case of DSPO, given a graph with n vertices and k, one can decide if a dominating set of size k exists for the graph by enumerating all potential dominating sets of size k in $\binom{n}{k}$ time (or approximately n^k) and then checking each set to see if it is indeed a dominating set. DSPO is *simple*.

The key property of a *measure preserving reduction* is that the quality of the solution is preserved by the mapping that associates the two problems. This means that if the quality of the answer to a problem instance A is within a factor φ of optimal, and if there is a *measure preserving reduction* from B to A, then the quality of the answer for B is the same. Let us consider reducing an instance of MINIMUM SET COVER OPTIMIZATION (MSCO) to DSPO.

PROBLEM: MINIMUM SET COVER [17]

INSTANCE: Collection C of subsets of a finite set S, number of subsets $K \leq |C|$.

QUESTION: Does C contain a cover for S of size K or less, i.e., a subset $C' \in C$ with $|C'| \leq K$ such that every element of S belongs to at least one member of C'?

The reason such a reduction is of interest is that in Hochbaum [27] there are published bounds on the degree of approximability for MSCO. If a *measure preserving reduction* exists from MSCO to DSPO, then these bounds apply also to DSPO. The first part of the proof of a *measure preserving reduction* requires showing that MSCO is *simple*. The proof of this is very similar to that for DSPO. Given an instance C of MSCO and a k, in $\binom{|C|}{k}$ time, all possible subcollections of size k can be enumerated. The subset in each subcollection can then be checked to see if any actually covers the entire set S.

The second part of the proof requires that there is a reduction from MSCO to DSPO that preserves the size of the solution. Let us assume that the instance of MSCO is: $C = \{S_1, ..., S_n\}$, where $\bigcup_{i=1}^{n} S_i = S = \{x_1, ..., x_m\}$[53]. From the clauses and variables, a graph is constructed. For each clause $S_i \in C$, graph G has a *cover setting component* $C_i = (V_i, E_i)$, with $V_i = \{S_i\}$ and $E_i = \{(i, j) | i \leq j \leq n\}$. All set-vertices are connected to all other set-vertices, so any set-vertex, i, dominates all set-vertices.

For each variable $x_j \in S$, graph G has a *variable testing component* $T_i = \{V_j'\}$, with $V_j' = x_j$, a vertex for each variable and no edges.

The only part of the construction that depends on which variables occur in which sets is the collection of *communication edges*. For each $S_i \in C$, the communicating edges connect the set to the variables that comprise the set, $E_i'' = \{(S_i, x_t) | x_t \in S_i\}$.

The construction of our instance of DSPO is completed by setting $G = (V, E)$ where

$$V = (\bigcup_{i=1}^{n} V_i) \cup (\bigcup_{j=1}^{m} V_j')$$

and

$$E = (\bigcup_{i=1}^{n} E_i) \cup (\bigcup_{j=1}^{n} E_i'').$$

All that remains to be shown is that C' is a set cover of S if and only if G has a dominating set V' where $|C'| = |V'|$. First, suppose that $V' \subseteq V$ is a dominating set for G.

In a minimal solution to DSPO, V' will be made up of all set-vertices, since each set-vertex dominates all other vertices, and variable-vertices can dominate only set-vertices. Thus we can use the way in which V' intersects with the cover setting component to choose which sets to include in the C'. $S_i \in C'$ if $S_i \in V'$. To see that C' is a cover of S, consider the edges in E_i''. In order for a variable-vertex to be dominated, there must exist an S_i that contains that variable, so for each variable-vertex, x_j, there exists an S_i that contains x_j. Therefore C' is indeed a set cover and by construction $|C'| = |V'|$.

Conversely, suppose that C' is a set cover of S. The corresponding dominating set V' includes one vertex for each $S_i \in C'$, so if $S_i \in C'$, then $S_i \in V'$. Since V' is made up entirely of set-vertices, all set-vertices are dominated. Since all variables are covered in C', there is an edge in E_i'' that connects each variable-vertex, x_j, to a vertex, $v \in V'$. Therefore V' is indeed a dominating set and by construction $|V'| = |C'|$.

Since we assume an efficient solution to DSPO, there is a unique C' for each dominating set V' and vice-versa. Throughout this discussion we have said nothing about C' or V' being a minimum set. Since the solutions may be approximations, as long as the optimal solutions are the same size, then the approximations are of the same quality. We have already shown that the approximations are the same size. An optimal solution V' for a particular instance of DSPO, when mapped to the corresponding instance of MSCO will yield an optimal solution for C'. To prove this, let us assume that C' is not optimal, so there is a set S_i that can be removed from C' while C' remains a set cover. This means that the corresponding vertex in V' can also be removed. Since V' is optimal, Si cannot exist, so C' must be optimal. The converse proof is the same. Since the optimal solutions are the same size, the approximations are of the same quality.

Finally, since a *measure preserving reduction* exists from MSCO to DSPO, the best possible approximation factor of DSPO is the same as that of MSCO. One cannot approximate MSCO to within a factor $(1 - \delta)\ln N$ [27], so at best an approximation factor of ln

N can be found for DSPO. Moreover, this factor is, indeed, achievable through a greedy approximation[30, 39].

APPENDIX B

INTERPRETING A BOX PLOT

Figure B.1 contains a standard box plot used to visualize data, and to provide the ability to compare data sets as is done in Section 5.2.2. The main features of the box plot are the box, which outlines the 25th and 75th percentiles; the median, which is a horizontal line indicating the median value; and the whiskers, which extend to at most to 1.5 times the interquartile range. Any boxes outside the whiskers represent individual data points.

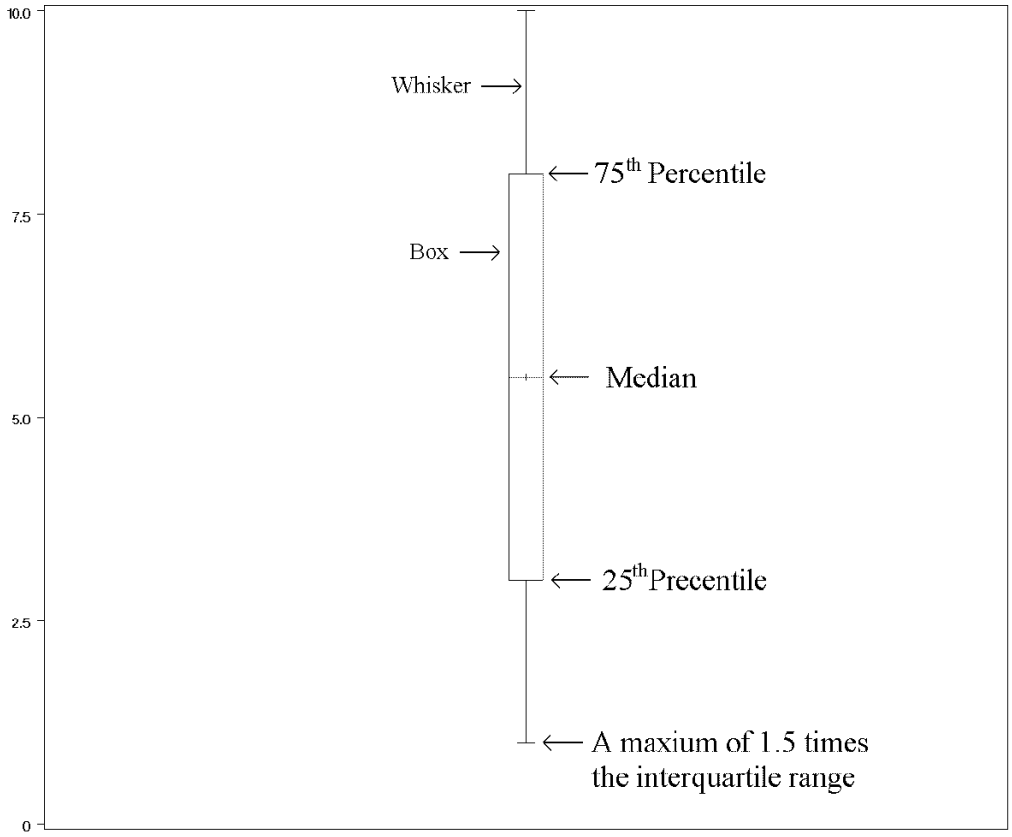


Figure B.1. Illustrates a standard box plot and labels the interesting statistical points.

APPENDIX C

MODIFICATIONS TO TREC TOPICS FOR WEB RETRIVAL

The following is a list of queries issued to the Google Search engine to create the snippet document sets, for TREC topics where we were unable to use the title of the topic provided by TREC. For TREC topics 201-250, there was no title, so we created brief descriptions of the topic to be used as queries. Any topics that fall outside this set are a shortend form of title which was required to retrieve a minimum of 500 documents.

TREC topic	*Query Used*
201	Au Pair child care
202	nuclear proliferation treaty
203	Economic recycling tires
204	nuclear power plants
205	paramilitary activity U.S.
206	third U.S. political party
207	Quebec independence
208	bioconversion
209	Social Security broke
210	medical waste
211	DWI intoxicated driving
212	fail protect copyrights
213	DNA testing
214	self-induced hypnosis
215	infant mortality United States

TREC topic	*Query Used*
216	reduce osteoporosis
217	extraterrestrial life intelligence
218	mini steel mills
219	auto imports exports
220	crossword puzzle makers
221	youth drug gang warfare
222	capital punishment deterrent
223	responsible Microsoft computer industry
224	lower blood pressure
225	Federal Emergency Management Agency
226	lottery gambling state economy
227	friendly fire training accidents
228	environmental recovery pollution
229	schizophrenia
230	electric powered automobile
231	U.S. support Endowment Arts
232	near-death experience
233	co generation electric power
234	fuel cell technology
235	U.S. legalizing drugs
236	sea laws
237	alternative sources energy automobiles
238	cost U.S. National Parks
239	cancer regions United States
240	combating terrorism

TREC topic	*Modified Title for Query*
241	doctor lawyer malfeasance
242	affirmative action construction industry
243	restrict fossil fuels
244	trade balance Japan
245	retirement communities
246	U.S. arms exports
247	economic chunnel
248	electronic technology blind
249	rain forest world weather
250	correlation sale firearms crimes
252	Combating people Smuggling
253	Cryonic suspension
254	Non-invasive procedures heart ailments
261	Soviet Union nuclear terrorist
262	Seasonal affective disorder syndrome
306	African Civilian Death
326	Ferry Sinking
328	Pope Beatification
338	Aspirin adverse effect
342	Diplomatic classified industrial projec
344	Abuse of E-Mail

APPENDIX D

OFFLINE ANALYSIS RESULTS

This appendix is a sample of the results discussed in Chapter 6. For each hierarchy, we computed three measures: EMIM, Reachability, and Access to Relevant Documents. The tables below indicate the group of hierarchies that are tested by query set ("Query"), database ("DB"), and number of topics asked for at a given level of the hierarchy ("Lev. Sz."). The Tukey Grouping is then indicated. Different letters indicate significant differences between the different types of hierarchies that are being tested. In this particular sample, we show the results of the parameter "All words". After the Tukey Grouping is the mean value of the score produced by the particular test. This is followed by the number of hierarchies that were used to determine the mean. This number is generally 50, which are the number of queries in each set. The last column of the table refers to the setting of the parameter. In this case, a "1" indicated that "All Words" were used to generated the hierarchy, and a "0" did not use all the words. Please refer to Section 6.2.1 for a description of the parameter and a discussion of these results.

D.1 EMIM

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
201-250	comp	20	A	0.538820	50	0
201-250	comp	20	B	0.500597	50	1
201-250	news	20	A	0.487116	50	0
201-250	news	20	B	0.450413	50	1
201-250	web	20	A	1.42393	50	1
201-250	web	20	B	1.35101	50	0
201-250	comp	15	A	0.526238	50	0
201-250	comp	15	B	0.486463	50	1
201-250	news	15	A	0.461967	50	0
201-250	news	15	B	0.429721	50	1
201-250	web	15	A	1.280627	50	1
201-250	web	15	B	1.218993	50	0
201-250	comp	10	A	0.493015	50	0
201-250	comp	10	B	0.462076	50	1
201-250	news	10	A	0.424318	50	0
201-250	news	10	B	0.398480	50	1

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
201-250	web	10	A	1.095200	50	1
201-250	web	10	B	1.033071	50	0
201-250	comp	5	A	0.436910	50	0
201-250	comp	5	B	0.418934	50	1
201-250	news	5	A	0.352201	50	0
201-250	news	5	B	0.336311	50	1
201-250	web	5	A	0.740545	50	1
201-250	web	5	B	0.709087	50	0
251-300	comp	20	A	0.414677	49	0
251-300	comp	20	B	0.369354	48	1
251-300	news	20	A	0.558250	50	0
251-300	news	20	B	0.477338	50	1
251-300	web	20	A	1.420275	50	1
251-300	web	20	B	1.346647	50	0
251-300	comp	15	A	0.409152	48	0
251-300	comp	15	B	0.351600	50	1
251-300	news	15	A	0.536734	50	0
251-300	news	15	B	0.457329	50	1
251-300	web	15	A	1.302202	50	1
251-300	web	15	B	1.242275	50	0
251-300	comp	10	A	0.383961	50	0
251-300	comp	10	B	0.340643	49	1
251-300	news	10	A	0.492365	50	0
251-300	news	10	B	0.426503	50	1
251-300	web	10	A	1.124876	50	1
251-300	web	10	B	1.068436	50	0
251-300	comp	5	A	0.355141	50	0
251-300	comp	5	B	0.317073	50	1
251-300	news	5	A	0.401132	50	0
251-300	news	5	B	0.364976	50	1
251-300	web	5	A	0.764703	50	1
251-300	web	5	B	0.724029	50	0
301-350	comp	20	A	0.416148	50	0
301-350	comp	20	B	0.373971	50	1
301-350	news	20	A	0.431706	50	0
301-350	news	20	B	0.385673	50	1
301-350	web	20	A	1.44917	50	1
301-350	web	20	B	1.37167	50	0
301-350	comp	15	A	0.404994	50	0
301-350	comp	15	B	0.367228	50	1
301-350	news	15	A	0.414809	50	0
301-350	news	15	B	0.372655	50	1
301-350	web	15	A	1.309719	50	1
301-350	web	15	B	1.248318	50	0

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
301-350	comp	10	A	0.383512	50	0
301-350	comp	10	B	0.349900	33	1
301-350	news	10	A	0.388327	50	0
301-350	news	10	B	0.354843	50	1
301-350	web	10	A	1.127758	50	1
301-350	web	10	B	1.070134	50	0
301-350	comp	5	A	0.341830	50	0
301-350	comp	5	B	0.330963	50	1
301-350	news	5	A	0.335553	50	0
301-350	news	5	B	0.322581	50	1
301-350	web	5	A	0.740242	50	1
301-350	web	5	B	0.717231	50	0

D.2 Reachability

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
201-250	comp	20	A	0.871212	500	0
201-250	comp	20	A	0.869272	500	1
201-250	news	20	A	0.819960	500	1
201-250	news	20	A	0.815360	500	0
201-250	web	20	A	0.0124832	500	0
201-250	web	20	A	0.0122540	500	1
201-250	comp	15	A	0.815740	500	0
201-250	comp	15	A	0.810412	500	1
201-250	news	15	A	0.757820	500	1
201-250	news	15	A	0.757200	500	0
201-250	web	15	A	0.0105079	500	0
201-250	web	15	B	0.0095979	500	1
201-250	comp	10	A	0.707116	500	0
201-250	comp	10	B	0.693856	500	1
201-250	news	10	A	0.637232	500	0
201-250	news	10	A	0.633780	500	1
201-250	web	10	A	0.0071958	500	1
201-250	web	10	A	0.0068924	500	0
201-250	comp	5	A	0.421788	500	0
201-250	comp	5	A	0.415112	500	1
201-250	news	5	A	0.369008	500	0
201-250	news	5	B	0.361808	500	1
201-250	web	5	A	0.0045079	500	1
201-250	web	5	A	0.0045044	500	0
251-300	comp	20	A	0.848232	500	0
251-300	comp	20	B	0.826812	500	1
251-300	news	20	A	0.836200	500	1
251-300	news	20	A	0.828908	500	0

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
251-300	web	20	A	0.02000	500	0
251-300	web	20	A	0.02000	500	1
251-300	comp	15	A	0.794676	500	1
251-300	comp	15	B	0.782916	500	0
251-300	news	15	A	0.770656	500	1
251-300	news	15	A	0.763656	500	0
251-300	web	15	A	0.02000	500	0
251-300	web	15	A	0.02000	500	1
251-300	comp	10	A	0.702236	500	0
251-300	comp	10	B	0.671816	500	1
251-300	news	10	A	0.644804	500	1
251-300	news	10	B	0.632980	500	0
251-300	web	10	A	0	500	0
251-300	web	10	A	0	500	1
251-300	comp	5	A	0.410640	500	0
251-300	comp	5	B	0.400036	500	1
251-300	news	5	A	0.366268	500	0
251-300	news	5	A	0.365968	500	1
251-300	web	5	A	0	500	0
251-300	web	5	A	0	500	1
301-350	comp	20	A	0.877956	500	0
301-350	comp	20	B	0.870740	500	1
301-350	news	20	A	0.845024	500	1
301-350	news	20	B	0.834932	500	0
301-350	web	20	A	0	500	0
301-350	web	20	A	0	500	1
301-350	comp	15	A	0.820628	500	0
301-350	comp	15	B	0.812852	500	1
301-350	news	15	A	0.777572	500	1
301-350	news	15	B	0.769880	500	0
301-350	web	15	A	0	500	0
301-350	web	15	A	0	500	1
301-350	comp	10	A	0.710684	500	0
301-350	comp	10	B	0.695032	500	1
301-350	news	10	A	0.646904	500	1
301-350	news	10	A	0.643200	500	0
301-350	web	10	A	0	500	0
301-350	web	10	A	0	500	1
301-350	comp	5	A	0.419968	500	0
301-350	comp	5	B	0.406312	500	1
301-350	news	5	A	0.371020	500	0
301-350	news	5	A	0.365720	500	1
301-350	web	5	A	0	500	0
301-350	web	5	A	0	500	1

D.3 Access to Relevent Documents

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
201-250	comp	20	A	6.6231	48	0
201-250	comp	20	A	6.4089	48	1
201-250	news	20	A	8.1808	48	0
201-250	news	20	A	7.5824	48	1
201-250	comp	15	A	8.4113	48	0
201-250	comp	15	A	7.4783	48	1
201-250	news	15	A	9.2490	48	0
201-250	news	15	A	8.8348	48	1
201-250	comp	10	A	13.378	48	0
201-250	comp	10	A	11.973	48	1
201-250	news	10	A	14.678	48	1
201-250	news	10	A	12.223	48	0
201-250	comp	5	A	16.0860	48	0
201-250	comp	5	A	15.4867	48	1
201-250	news	5	A	18.9728	48	0
201-250	news	5	A	17.8630	48	1
251-300	comp	20	A	10.7385	45	0
251-300	comp	20	B	7.8416	42	1
251-300	news	20	A	5.1394	46	0
251-300	news	20	A	4.9265	46	1
251-300	comp	15	A	11.3248	46	1
251-300	comp	15	B	9.7967	43	0
251-300	news	15	A	8.134	46	0
251-300	news	15	A	5.816	46	1
251-300	comp	10	A	14.0856	46	0
251-300	comp	10	B	12.2019	44	1
251-300	news	10	A	12.004	46	0
251-300	news	10	A	9.083	46	1
251-300	comp	5	A	20.099	46	0
251-300	comp	5	A	19.370	46	1
251-300	news	5	A	16.996	46	0
251-300	news	5	A	15.522	46	1
301-350	comp	20	A	6.1716	49	1
301-350	comp	20	A	5.7477	49	0
301-350	news	20	A	6.9494	50	0
301-350	news	20	A	6.0234	50	1
301-350	comp	15	A	8.6406	49	1
301-350	comp	15	A	8.5569	49	0
301-350	news	15	A	8.5637	50	0
301-350	news	15	A	8.0685	50	1

Query	DB	Lev. Sz.	Tukey Grouping	Mean	N	AW
301-350	comp	10	A	10.3279	49	0
301-350	comp	10	A	10.2254	49	1
301-350	news	10	A	11.5699	50	0
301-350	news	10	A	11.2117	50	1
301-350	comp	5	A	16.2570	49	1
301-350	comp	5	A	15.6864	49	0
301-350	news	5	A	19.2863	50	0
301-350	news	5	A	17.8593	50	1

BIBLIOGRAPHY

[1] Agrawal, R., Bayardo, R., and Srikant, R. Athena: Mining-based interactive management of text databases. In *Proceedings of the 7th Conference on Extending Database Technology* (2000), pp. 365–379.

[2] Allan, J., Gupta, R., and Khandelwal, V. Temporal summaries of news topics. In *Proceedings on the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), pp. 10–18.

[3] Anick, P.G., and Tipirneni, S. The paraphrase search assistant: Terminological feedback for iterative information seeking. In *Proceedings on the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), M. Hearst, F. Gey, and R. Tong, Eds., pp. 153–159.

[4] Attardi, G., Marco, S. Di, and Salvi, D. Categorisation by context. *Journal of Universal Computer Science 4:9* (1998), 719–736.

[5] Bacon, F. *Of the proficience and advancement of learning, by Francis lord Verulam, ed. by B. Montagu, esq.* London, W. Pickering, 1851.

[6] Berger, A., and Mittal, V. OCELOT: A system for summarizing web pages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (2000), pp. 144–151.

[7] Carbonell, J., and Goldstein, J. Use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), pp. 335–336.

[8] Chen, H., and Dumais, S. Bringing order to the web: Automatically categorizing search rresults. In *Proceedings of the the Human Factors in Computing, CHI 2000* (2000), pp. 145–152.

[9] CNN. Scientists map suspected SARS virus genome. www.cnn.com/2003/HEALTH/04/14/sars.sequence/index.html, April 2003.

[10] Conroy, J., and O'Leary, D. Text summarization via hidden markov models. In *Proceedings on the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), pp. 406–407.

[11] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*. Wiley-Interscience, New York, New York, 1991.

[12] Cronen-Townsend, S., and Croft, W.B. Quantifying query ambiquity. In *Proceedings of HLT* (2002), pp. 94–98.

[13] Crouch, C., Crouch, D., and Nareddy, K. The automatic generation of extended queries. In *Proceedings on the 13th annual international ACM SIGIR conference on Research and development in information retrieval* (1990), pp. 369–383.

[14] Cutting, D.R., Karger, D.R., Pedersen, J.O., and Tukey, J.W. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR conference on Research and development in information retrieval* (Copenhagen Denmark, 1992), pp. 318–329.

[15] Feng, F., and Croft, W.B. Probabalistic techniques for phrase extraction. *Information Process Management 37*, 2 (March 2001), 199–220.

[16] Fuhr, N., Hartmann, S., Lustig, G., Tzeras, K., Knorz, G., and Schwantner, M. Automatic indexing in operation: The rule-based system air/x for large subject fields. Tech. rep., Technische Hochschule Darmstadt, 1993.

[17] Garey, M., and Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Wlt Freeman and Company, 1979.

[18] Glover, E., Pennock, D., Lawrence, S., and Krovetz, R. Inferring hierarchical descriptions. In *Proceedings of the 11th International Conference on Informaiton and Knowledge Management* (2002), pp. 507–515.

[19] Goldstein, J., Kantrowitz, M., Mittal, V., and Carbonell, J. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of the 22nd Annual International ACM SIGIR conference on Research and development in information retrieval* (Berkeley, California, 8 1999), pp. 121–128.

[20] Goldstein, J., Mittal, V., Carbonell, J., and Callan, J. Creating and evaluating multi-document sentence extract summaries. In *Proceedings of the 9th International Conference on Informaiton and Knowledge Management* (2000), pp. 165–172.

[21] GOOGLE. Google. www.google.com.

[22] Grobelnik, M., and Mladenić, D. Fast text categorization. In *Proceeding of the Text Mining Workshop on ECML* (1998).

[23] Harabagiu, S., and Maiorano, S. Multi-document summarization with GISTEXTER. http://www.languagecomputer.com/papers/lrec2002.pdf, 2002.

[24] Hardy, H., Shimizu, H., Strzalkowski, T., Ting, L., Wise, G. B., and Zhang, X. Cross-document summarization by concept classification. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 121–128.

[25] Harman, D., Ed. *Proceedings of the 2001 Document Understanding Conference* (2001), Department of Commerce, National Institute of Standards and Technology.

[26] Hersh, W., and Over, P. Trec-8 interactive track report. In *The Eighth Text REtrieval Conference (TREC-8)* (1999), pp. 57–64.

[27] Hochbaum, D., Ed. *Approximation Algorithms for NP-Hard Problems*. PWS publishing Company, 1997, ch. Hardness of Approximations, pp. 399–446.

[28] Hofmann, T. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Proceedings of International Joint Conference on Artificial Intelligence 1999* (1999), pp. 682–687.

[29] Jain, A., and Dubes, R. *Algorithm for Clustering Data*. Pretice Hall, Engelwood Cliffs, N.J., 1988.

[30] Johnson, D. S. Approximation algorithms for combinatorial problems. *Journal of Computer System Science 9* (1974), 256–274.

[31] Kan, M., and McKeown, K. Information extraction and summarization: Domain independence through focus types. Tech. Rep. CUCS-030-99, Columbia University, 1999.

[32] Koller, D., and Sahami, M. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Leaning* (1997), pp. 170–178.

[33] Kosovac, B., Vanier, D., and Froese, T. Use of keyphrase extraction software for creation of an aec/fm thesaurus. *Electronic Journal of Information Technology in Construction 5* (2000), 25–36.

[34] Kupiec, J., Pederson, J., and Chen, F. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1995), pp. 68–73.

[35] Lavrenko, V., and Croft, W.B. Relevance-based language models. In *Proceedings on the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), p. to appear.

[36] Lawrie, D., and Croft, W.B. Discovering and comparing topic hierarchies. In *Proceedings of RIAO 2000 Conference* (2000), pp. 314–330.

[37] Leuski, A. *Interactive Information Organization: Techniques and Evaluation*. PhD thesis, University of Massachusetts Amherst, 2001.

[38] Lin, C., and Hovy, E. Automated multi-document summarization in NeATS. In *Proceedings of the DARPA Human Language Technology Conference* (2002), pp. 50–53.

[39] Lovász, L. On the ratio of optimal integral and fractional covers. *Discrete Mathematics 13* (1975), 383–390.

[40] Lowe, H., and Barnett, G. Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. *Journal of the American Medical Association 271(4)* (1994), 1103–1108.

[41] Luhn, H. The automatic creation of literature abstracts. *IBM Journal of Research and Development 2* (1958), 159–165.

[42] Maedche, A., Pekar, V., and Staab, S. Ontology learning part one – on discovering taxonomic relations from the web. In *Web Intelligence*. Springer, 2002, ch. 1.

[43] Mani, I., and Bloedorn, E. Multi-documemt summarization by graph search and matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence* (1997), pp. 622–628.

[44] Manning, C., and Schutze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[45] McKeown, K., Klavans, J., Hatzivzssiloglou, V., Barzilay, R., and Eskin, E. Generating summaries of multiple news articles. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and Development in Information Retrieval* (1995), pp. 74–82.

[46] McKeown, K., Klavans, J., Hatzivzssiloglou, V., Barzilay, R., and Eskin, E. Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999), pp. 453–460.

[47] Mittal, V., Kantrowitz, M., Goldstein, J., and Carbonell, J. Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999), pp. 467–473.

[48] National Public Radio. Severe acute respiratory syndrome (SARS). www.npr.org/news/specials/sars/index.html, April 2003.

[49] Nevill-Manning, C., Witten, I., and Paynter, G. Lexically-generated subject hierarchies for browsing large collections. *International Journal on Digital Libraries 2(2+3)* (1999), 111–123.

[50] OCLC Forest Press. Introduction to the dewey decimal classification. http://www.oclc.org/dewey/about/about_the_ddc.htm, 2002.

[51] Over, P. Trec-6 interactive track report. In *The Sixth Text REtrieval Conference (TREC-6)* (1997), pp. 73–82.

[52] Over, P. Trec-7 interactive track report. In *The Seventh Text REtrieval Conference (TREC-7)* (1998), pp. 65–72.

[53] Paz, A., and Moran, S. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science 15(3)* (1981), 251–277.

[54] Radev, D., Blair-Goldensohn, S., and Zhang, Z. Experiments in single and multi-document summarization using MEAD. In *Proceeding of the Document Understanding Conference* (2001).

[55] Robertson, S. E. *The Probability Ranking Principle in IR*. Morgan Kaufmann Publishers Inc., San Francisco, California, 1997, pp. 281–286.

[56] Salton, G., and McGill, M. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[57] Salton, G., Singhal, A., Mitra, M., and Buckley, C. Automatic text structuring and summarization. *Information Processing and Management 33* (1997), 193–207.

[58] Sanderson, M. Personal communication, September 2002.

[59] Sanderson, M., and Croft, W. B. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval* (1999), pp. 206–213.

[60] Schiffman, B. Producing biographical summaries: Combining linguistic and knowledge with corpus statistics. In *Proceedings of the 39th Association for Computational Linguistics* (2001), pp. 450–457.

[61] Schiffman, B., Nenkova, A., and McKeown, K. Experiments in multidocument summarization. In *Proceedings of the DARPA Human Language Technology Conference* (2002), pp. 44–49.

[62] Sparck Jones, K. *Automatic Keyword Classification*. Butterworths, 1971.

[63] Sun Microsystems, Inc. Java 2 sdk, standard edition version 1.4.1. http://java.sun.com/j2se/1.4.1/, 2002.

[64] Utiyama, M., and Hasida, K. Multi-topic multi-document summarization. In *Proceedings of COLING* (2000), pp. 892–898.

[65] van Rijsbergen, C.J. *Information retrieval*, second ed. Butterworths, London, 1979.

[66] Voorhees, E. M., and Harman, D. K., Eds. *The Sixth Text REtrieval Conference (TREC-6)* (1997), Department of Commerce, National Institute of Standards and Technology.

[67] White, M., and Cardie, C. Multidocument summarization via information extraction. In *Proceedings of the Summarization Workshop at the Association of Computational Linguistics Conference* (2002), p. XXX.

[68] White, M., Korelsky, T., Cardie, C., Ng, V., Pierce, D., and Wagstaff, K. Multidocument summarization via information extraction. In *Proceedings of the DARPA Human Language Technology Conference* (2001), pp. 143–146.

[69] Willett, P. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management 24(5)* (1988), 577–587.

[70] Witbrock, M., and Mittal, V. Ultra-summarization: A statistical approach to generating highly condensed non-extractive summaries. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), pp. 315–316.

[71] Witten, I., Paynter, G., Frank, E., Gutwin, C., and Nevill-Manning, C. Kea: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM conference on Digital Libraries* (1998), pp. 254–255.

[72] Xu, J., and Croft, W.B. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* (1996), pp. 4–11.

[73] YAHOO. Yahoo. www.yahoo.com.

[74] Yand, Y., Pierce, T., and Carbonell, J. A study on retrospective and on-line event dection. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval* (1998), pp. 28–36.

[75] Zamir, O., Etzioni, O., Madani, O., and Karp, R. Fast and intutive clustering of web documents. In *Proceeding of the 3rd International Conference on Knowledge Discovery and Data Mining* (1997), pp. 287–290.

[76] Zamir, Oren, and Etzioni, Oren. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999) 31*, 11–16 (1999), 1361–1374.

[77] Zha, H. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 113–120.