Discovering and Comparing Topic Hierarchies

Dawn Lawrie and W. Bruce Croft
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA

Abstract

Hierarchies have been used for organization, summarization, and access to information, yet a lingering issue is how best to construct them. In this paper, our goal is to automatically create domain specific hierarchies that can be used for browsing a document set and locating relevant documents. We examine methods of automatically generating hierarchies and evaluating them. To this end, we compare and contrast two methods of generating topic hierarchies from the text of documents: one, subsumption hierarchies, uses subsumption relations found within document sets, and the other, lexical hierarchies, utilizes frequently used words within phrases. Our evaluation shows that subsumption hierarchies divide documents into smaller groups, allowing one to find all relevant documents without looking at as many non-relevant documents. However, such hierarchies are more likely to contain no path to a relevant document.

1 Introduction

Hierarchies are an intuitive way to describe information. One can find organizational systems that utilize hierarchies in the Library of Congress, Yahoo, and the personal file cabinet. They have been used as a tool for classification, as in the field of Biology. They are also the structural basis of newspapers. Because people find them easy to understand, finding ways to automatically generate hierarchies would be advantageous, since such hierarchies can be used for summarizing and browsing large document sets.

Currently most hierarchies are created manually. Because so much work goes into their generation, hierarchies are very general in order to meet the needs of a large number of users. Let us briefly consider an individual with interests in gold mining, who uses Yahoo! (YAHOO), a typical general-purpose hierarchy. There are two main locations in the hierarchy where interesting documents might be found "Home > Recreation > Hobbies > Rocks, Gems, and Minerals > Prospecting" and "Home > Business and Economy > Companies > Mining and Mineral Exploration > Precious Metals and Diamonds > Gold". The former is further classified into Organizations and Equipment. The latter is further classified into Organizations; however, there are 98 documents under "Gold", and only a short description, if that, to distinguish them. As a general hierarchy, Yahoo! cannot afford to spend a disproportionate amount of resources on organizing gold mining because that would mean another topic would have to be neglected. From the user's point of view, however, further classification would improve information search. Automatically generating the hierarchy from the "gold mining" documents is one solution to this problem.

Generating hierarchies is not a new goal for information retrieval, and there have been past attempts using automatic techniques. One example is Crouch (1988), which automatically generates thesauri; however, the generated thesaurus was not "human usable". Another example is Scatter/Gather (Cutting *et al.*, 1992) in which clustering is used to create document hierarchies. However, because of the nature of clustering, fully explaining the contents of each level in the hierarchy is difficult.

Recently, new types of hierarchies have been introduced that rely on the terms used by a set of documents to expose some structure of the document collection. One such technique is lexical modification (Nevill-Manning *et al.*, 1999; Anick & Tipirneni, 1999; Anick, 1999) and another is subsumption, (Sanderson

& Croft, 1999). These two techniques are promising because they give fairly complete summaries of the document set. However, the two methods use very different techniques to create the hierarchies. Lexical modification relies on phrases within a document, and creates a hierarchy by using frequently used terms within the phrases as parents. The full phrases are treated as children. Subsumption looks at both words and phrases in the document set. It determines the probability that one term co-occurs with another. If a term normally occurs with another term, then the latter term subsumes the former term.

These techniques work well for fairly homogeneous sets of documents, yet are less effective for larger document sets. This is especially true for lexical modification because the number of key terms can be immense. By choosing any most frequent subset, important relationships can be left out. However, when the document set is more homogeneous, this becomes less of a problem. Subsumption finds more subsuming relations in homogeneous groups of documents. This is probably because terms are more likely to be used in the same way within similar documents.

Clustering has long been used to group documents. In this paper, we investigate the use of clustering to improve hierarchies constructed with subsumption and lexical relationships.

Another focus of this paper is to address the issue of evaluation. As previous work has shown, evaluating such hierarchies is a challenging task (Sanderson & Croft, 1999). Although hierarchies are designed to be used by people, user studies generally give ambiguous results. Alternative methods of evaluation would be very useful. In this paper, we develop two metrics of evaluation. The first measures the speed with which relevant documents can be found, and the other quantifies the similarity among different techniques of generating hierarchies.

In the next section we discuss previous work on clustering techniques and topic hierarchies. In Section 3, an example is given of how topic hierarchies could be used to find relevant documents. In Section 4, the creation of the hierarchies is explained. Our evaluation methods are discussed in Section 5, and results are given in Section 6.

2 Background

2.1 Clustering methodologies

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 they co-occur (Willett, 1988). Clustering has been used within two different environments—browsing and retrieval. Scatter/Gather (Cutting *et al.*, 1992) 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 (Salton & McGill, 1983), 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 (1971) and Crouch, Crouch, & Nareddy (1990) are examples of using term clusters for retrieval. Sparck Jones (1971) uses term clusters to find similarities among keywords in order to identify terms that could be substituted in a query. In Crouch, Crouch, & Nareddy (1990) documents are first clustered using the content term vector, 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.

In this paper we use document clustering to create more homogeneous groups of documents. Two types of document clustering have been explored: 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 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 C clusters while minimizing the total within-cluster sum of squares for each cluster. One example of this technique is the k-means algorithm (Jain & Dubes, 1988). 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 (van Rijsbergen, 1979).

2.2 Use of hierarchies for browsing

As in clustering, hierarchies can be created that are document oriented or term oriented. 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 (Cutting *et al.*, 1992) 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. For example, in Cutting *et al.* (1992) three clusters are identified: one about the Gulf War, one about oil sales and stock markets, and a third about East and West Germany. When these three clusters are re-clustered, it is revealed that somewhere within one of these clusters are documents about Pakistan, Trinidad, South Africa, and Liberia.

Term-oriented hierarchies have been much more common and make up a large portion of the manually created hierarchies such as MeSH (Medical Subject Headings) and Yahoo!. One area of research has been finding ways to automatically index documents once the hierarchies are created. Examples of such work include Koller & Sahami (1997) and Fuhr *et al.* (1993). Koller & Sahami (1997) uses Bayesian classifiers to classify documents under pre-existing topics in the hierarchy. Fuhr *et al.* (1993) determines the terms that should be used to index a document.

A drawback of using predefined hierarchies is that they are not 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, and not be predefined. Term-oriented hierarchies seem better suited to this task.

2.3 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 only occur in 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 Sanderson & Croft (1999) 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 descendents so that transitivity holds,

A → B (pcfgs)
B → probabilistic C
C → D grammars
D → context free

Figure 1: Production rules for "probabilistic context free grammars (pcfgs)".

- a child may have more than one parent.

These characteristics are achieved by defining subsumption as

$$P(x|y) \leq 0.8^{1} \text{ and } P(y|x) < P(x|y). \quad (1)$$

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 most 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.

Once one has defined a notion of subsumption, the candidate terms must be selected. Sanderson & Croft (1999) 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 from an automatic expansion are a very good way to focus the hierarchy in this situation, since they describe the interest of the user. Local Context Analysis(LCA) (Xu & Croft, 1996) is used for automatic expansion. 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 score, and the top N terms are designated for use in the subsumption hierarchy. Subsumption relationships are found using O(n^2) comparisons. 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 two or fewer documents.

2.4 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.* (1999) and Anick & Tipirneni (1999). 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 & Tipirneni (1999) 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."

There are many ways of selecting the phrases that will be used for candidates in the lexical hierarchy. Nevill-Manning *et al.* (1999) 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 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.* (1999), sequences averaged 9.6 non-terminals.

[1]The threshold 0.8 was determined empirically.

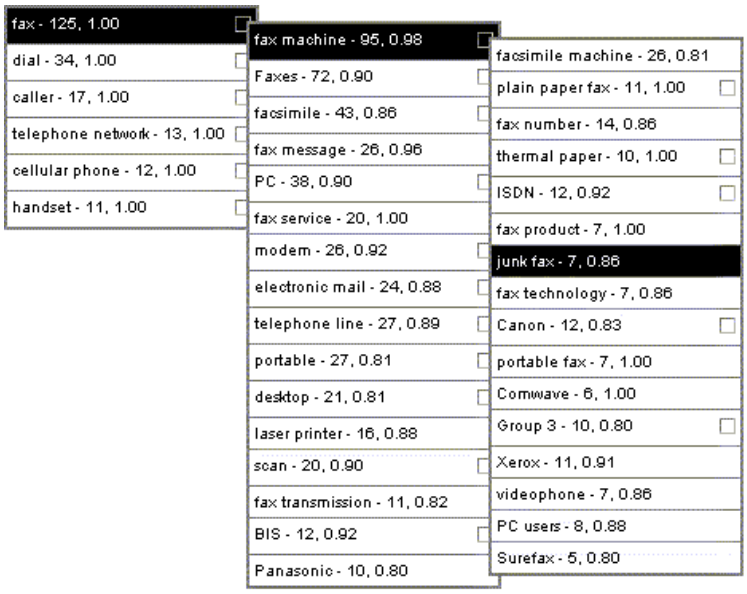

Figure 2: Subsumption hierarchy generated from first cluster of documents for TREC topic 317. The first of the two numbers associated with a term is the number of documents in which the term occurs. The second number gives the frequency that the term is subsumed by its parent.

Another way to locate sequences is to match the pattern $\{?adjective\ noun+\}$ of two to four terms in length, as was done in Anick & Tipirneni (1999).

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 & Tipirneni (1999) 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 & Tipirneni (1999) 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.

3 Comparing topic hierarchies based on a single query

Since we would like to use hierarchies to find relevant documents, we compare topic hierarchies as a demonstration of how they would be used. However, we realize that a user study is required to show the extent to which these types of hierarchies can be used to find relevant documents. The following is a proof of concept.

In this illustration, we will compare the hierarchies generated from documents retrieved for the TREC query (Voorhees & Harman, 1997) 317: "Unsolicited Faxes Description: Have regulations been passed by the FCC banning junk facsimile (fax)? If so, are they effective?". The reason this particular example is chosen is that articles using the phrase "junk fax" are an easy indication of some of the relevant documents. Five hundred documents were retrieved for this query. They were then clustered to create more homogeneous document groups. Figure 2 shows a subsumption hierarchy. In order to find relevant documents, one can follow a path from "fax" to "fax machine" to "junk fax". For the same group of documents, Figure 3 shows a lexical hierarchy. One would expect to follow the path "fax" to "junk fax" in order to find relevant documents. Unfortunately, the level underneath "fax" does not contain the phrase "junk fax". It turns out that this is due to the way that phrases are ranked in the lexical hierarchy. Priority is given to phrases with high dispersion. Since "junk fax" is not part of any larger phrase, it only has a

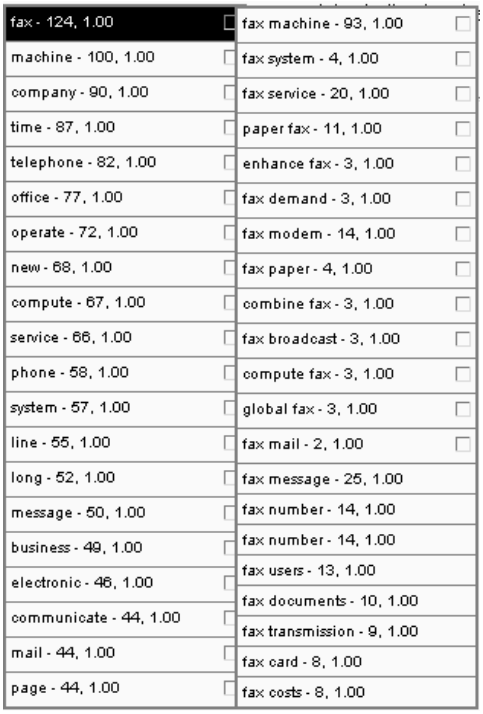

Figure 3: Lexical hierarchy generated from first cluster of documents for TREC topic 317. In the lexical hierarchy the first number associated with a term is also the number of documents in which the term appears. The second number is always 1.00 because a child cannot occur without its parent since the parent term is part of the child.

dispersion level of one. Another problem is that phrases of the same dispersion level are ranked in order of the number of documents in which they appear. The phrase "junk fax" appears in only six documents, giving it a rank of thirty. It is, therefore, too low in the ranking to be displayed.

In the second cluster where documents relating to the California State Legislature are grouped, finding relevant documents is not as straight forward. Figure 4 shows a second subsumption hierarchy. The top level has only two choices. Since we are trying to find relevant documents as quickly as possible with human intuition, "facsimile" is chosen over "D Los Angeles, Sen, SB[2]". At the second level, the choices are "Sen, SB" and "misdemeanor". Neither seems to be a likely candidate, but given that there are only two choices, both can be explored quickly. As it turns out "Sen, SB" leads to "Junk Fax", which

[2]This cluster contained articles relating to Senate Bills (SB) in the California State Legislature. Within articles that talked about a bill sponsored by a Democratic (D) Senator (Sen) from Los Angeles were ones that discussed a vote on junk faxes.

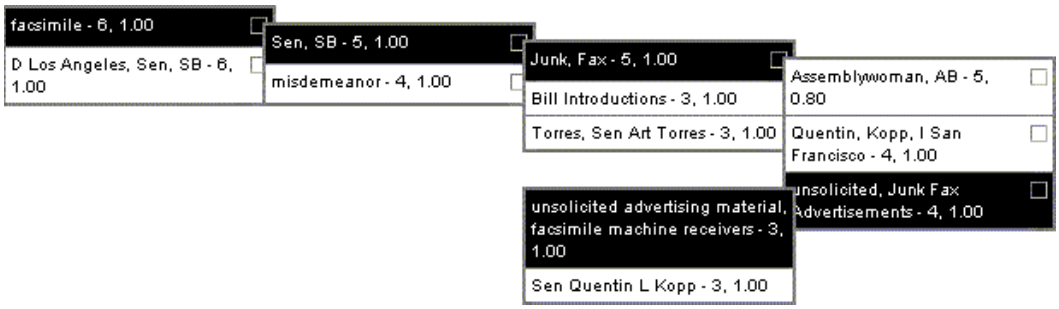

Figure 4: Subsumption hierarchy generated from second cluster of documents for TREC topic 317.

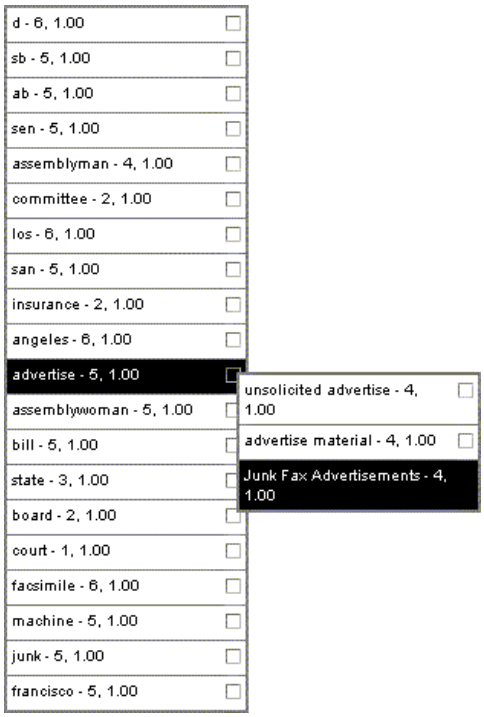

Figure 5: Lexical hierarchy generated from second cluster of documents for TREC topic 317.

means that relevant documents have been found. Figure 5 shows a hierarchy generated from the lexical algorithm using the same document set. In this case, the hard choice is the first one. However, someone interested in junk faxes would probably think that "advertise" is a promising candidate since junk faxes are in fact advertisements, even though they did not think of using that word when phrasing their query. Once "advertise" is chosen, "Junk Fax Advertisements" appears, and relevant documents are found.

4 Creation of Hierarchies

In order to create a hierarchy, several steps must be taken. First, the document set of interest must be identified. This does not necessarily have to be a retrieved document set. Instead, it could be a personal collection of papers or even email. These documents are clustered to provide homogeneous document sets from which topic hierarchies can be made. Candidate terms are then selected. Finally, a hierarchy is created that can be browsed by people.

4.1 Clustering

We chose to use two different clustering algorithms in order to determine the effectiveness of clustering documents before creating the hierarchy, and also to see if the type of clustering used had any effect on the hierarchies created. The two algorithms we chose were a modified k-means and an average-link hierarchical agglomerative clustering (HAC) algorithm. These were chosen primarily because of their popularity and ease of implementation.

The k-means algorithm required modification because of its tendency to form singleton and small clusters. Extremely small clusters make it difficult to create subsumption hierarchies. This is because terms must co-occur in at least three documents to be considered a valid relationship. If there are no interesting words that meet this restriction, no hierarchy will be created for the cluster. In order to make sure that a subsumption hierarchy could always be created, a minimum size of seven documents was used. Twelve documents were then randomly chosen as the seeds for clusters. A threshold was introduced so that documents that were very different from all cluster centers did not have to be placed within a cluster.

	K-means	*HAC*
Average max size	159.1	200.9
Average min size	11.6	5.4
Highest concentration of relevant docs	29.3%	41.3%

Table 1: Characteristics of the average cluster created using k-means and HAC.

After examining the similarities of documents, 2.75 was chosen empirically because an overwhelming majority of documents fell within this similarity, but it prevented very dissimilar documents from being included in a cluster. The k-means clustering was repeated up to ten times. With each iteration, clusters that were too small were added to the leftover cluster and new cluster seeds were chosen from among the documents in the leftover cluster. All documents were reclustered using the surviving clusters and the newly seeded clusters. The iteration would stop if no clusters were created that were too small. Table 1 contains information about the average clusters formed.

The HAC clustering algorithm was implemented in the way that is described in Sahami (1998). Each document began as a singleton cluster and clusters were joined until the correct number of clusters was formed. In this case, thirteen clusters were created for each document set. Although singleton clusters were never formed, some larger clusters did not yield any subsuming relationships, in which case no hierarchies were created. However, 83% of these clusters without hierarchies also did not have any relevant documents. A lexical hierarchy could always be created, regardless of the number of documents in the cluster. More information about the average clusters can be found in Table 1.

Overall, HAC did a better job of grouping relevant documents together. Some 55.4% of clusters contained no relevant documents when the HAC algorithm was used, and 47.1% of clusters contained no relevant documents when the k-means algorithm was used. Unfortunately, 3.5% of the clusters created using HAC contained no subsuming relations. All clusters created using the k-means algorithm yielded a subsumption hierarchy.

4.2 Generating Structures

After the clusters are created, a subsumption hierarchy and a lexical hierarchy are formed for each cluster. Both hierarchies rely on phrases extracted. We use a phrase identification process created for 'in-house' use at the CIIR, University of Massachusetts. These phrases are similar to the ones extracted by Anick & Tipirneni (1999) but do not limit phrases to four words. The subsumption hierarchies created in Sanderson & Croft (1999) include LCA terms and single terms as well. Because we are interested in creating hierarchies in situations where a query is not available, we also create hierarchies that do not make use of LCA terms to measure the contribution of these terms.

Neither Anick & Tipirneni (1999) nor Nevill-Manning *et al.* (1999) form hierarchies of the type in which we are interested. In order to form a hierarchy that is more than two levels, we employ a method similar to what Nevill-Manning *et al.* (1999) does when creating the rules for a context free grammar. In our case, single words are located at the highest level in the hierarchy. In the second level, all combinations of two word phrases are examined. If any other phrase contains the same two-word combination, the phrases are conflated and appear together at the next level of the hierarchy. All phrases at a given level are displayed in order of their dispersion. If multiple phrases have the same dispersion level, they are ranked by the number of documents that include the phrase.

The subsumption hierarchies are created in the same way as described in Section 2.3: candidate terms are identified, subsumption relations are found, and the relationships are organized into a hierarchy. The only difference that we employ is in the use the LCA terms. For some hierarchies we exclude the LCA terms from the candidate terms.

```
for each relevant document d
    if d seen before?
        d.path length ← 0
    else find all leaves with d
        if (num leaves > 0)
            lm ← menu with max # rel docs
            d.path length ← lm.new menus +
                            lm.total new docs
        else find all menus with d
            if (num menus >0)
                m ← menu with min # total
                docs
                d.path length ← m.new menus
                                +m.total new
                                docs
            else
                d.path length ← -1  #no path
```

Figure 6: Algorithm assigns a path length to each relevant document.

Once all the relationships are determined, the hierarchies need to be displayed in such a way that people can easily view them. We chose to use the hierarchical menu system (DHTMLAB) that was used in Sanderson & Croft (1999).

5 Evaluation

We evaluate the hierarchies from two perspectives. First, we examine how quickly one could find all relevant documents if one knew which nodes in the hierarchy held relevant documents. This is done because the intended use of the hierarchies is to find relevant documents. Second, we examine the similarities of the different hierarchies. By quantifying the similarity, we can learn more about the strengths and weaknesses of the two methods of creating topic hierarchies.

5.1 Scoring the Hierarchies

The scoring algorithm estimates the time it takes to find all relevant documents by calculating the total number of menus that must be traversed and the number of documents that must be read. The algorithm aims to find an optimum route through the hierarchy traveling to nodes that hold the greatest concentration of relevant documents. Since we begin with the knowledge of where the documents are located, 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 non-zero document paths.

Figure 6 shows the path length algorithm. Given that documents often belong to more than one menu, it is necessary to choose which of these will be used when calculating the path. To do this, we break the menus into two groups. The first group consists of leaf menus. These types of menus are favored because they tend to have a smaller number of documents associated with them. Smaller document groups are also likely to be more homogeneous. From among these leaf menus, we favor the menu with the most relevant documents because we are computing an optimal path. If there are no leaf menus, then all menus containing the document are considered. In this case, we favor menus that contain a small number of documents. The path to a relevant document is composed of the previously unexplored menus that are traversed to reach it, and the unread documents associated with the final menu. Since the documents belonging to a particular menu item are not sorted in any way, it is assumed that users will have to read all new documents in the group in order to find the relevant one(s).

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 despite the expected 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.

5.2 Quantifying Similarity

In order to find similarity, we examine all parent-child pairs within two hierarchies. Since the term pairs define the hierarchy, this comparison provides a way to measure how similar one hierarchy is to another. It is fairly straightforward to count the number of term pairs that two hierarchies have in common. The difficulty comes when trying to express this number in a meaningful way. The problem is that no two hierarchies have the same number of pairs, so comparing the raw number of overlapping pairs is meaningless. Instead, a ratio comparing the number of overlapping pairs to the total number of pairs in one of the hierarchies is used. This means that if hierarchy A has 1000 term pairs, hierarchy B has 1500 term pairs, and the hierarchies share 500 pairs, then the ratio with regard to A is $\frac{1}{2}$ and the ratio with regard to B is $\frac{1}{3}$. The problem with this method is that in order to know how each hierarchy relates to all other hierarchies, one would have to perform n^2 comparisons. However, even without comparing every single hierarchy to every other, one can get an idea of how much the different types of hierarchies have in common, and how different groupings of document sets affect the hierarchies.

6 Results

Our experiments are designed to reveal two main characteristics about the hierarchies involved. First, we want to determine the difference between subsumption and lexical hierarchies. Second, we want to determine what effect clustering the document sets has on the hierarchies. We are also interested in two other finer details within the creation of the hierarchies. We want to examine the contribution of the expanded query terms (LCA) to the subsumption hierarchies, and to determine if any preference is given to the type of clustering performed.

Our experiments make use of TREC topics 301-350 and associated relevance judgments. We have retrieved 500 documents using InQuery (Callan, Croft, & Harding, 1992) for each of the 50 queries. We treat a set of 500 documents for a given query as a document set. A number of hierarchies are generated for each document set. These include hierarchies of three different groupings of the document sets: one clustered using k-means, one clustered using HAC, and the other left as a single document group. For each of these three groupings, subsumption hierarchies are created that make use of LCA terms and hierarchies that do not use LCA terms. Lexical hierarchies are also created for the three document groups.

Hierarchies are assigned a path length score using the algorithm described in Section 5.1. A lower score denotes a superior hierarchy. We compare our hierarchies to those formed through a random subsumption process, as well as to each other. Random hierarchies are formed in the same manner as subsumption hierarchies (as described in Section 2.3) except that when all terms are compared to all other terms, random selection is used to form parent-child pairs instead of the subsumption criteria from Equation 1.

Once all the hierarchies are scored, they are compared on a basis of the average path to a document. This is used instead of doing a straight comparison of the total path length because it is possible that some relevant documents are unreachable. The total path length for a particular hierarchy could end up being shorter simply by leaving out relevant documents. By using the average path length, we neither reward nor penalize a hierarchy for excluding relevant documents. It was found empirically that randomly generated hierarchies were more likely to leave relevant documents out of the hierarchy than the other hierarchies, except when a single subsumption hierarchy was generated for the entire document set without using LCA. This particular group of hierarchies needs the LCA terms because the document set is less homogeneous than when the document set is first clustered. The average percentages of relevant

Hierarchy	*% no path*
Lexical	5.1%
Clustered Subsumption with LCA	2.5%
Clustered Subsumption without LCA	11.2%
Unclustered Subsumption with LCA	1.90%
Unclustered Subsumption without LCA	28.70%
Random	13.8%

Table 2: 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. The unclustered subsumption hierarchy without LCA leaves out a substantial number of the relevant documents because the document set is so large that it requires the LCA terms to focus it.

	Subsumption		*Lexical*		
	Smaller	*Avg. difference*	*Smaller*	*Avg. difference*	*Equal*
K-means - LCA	38	5.04	11	1.63	1
K-means - no LCA	34	5.03	14	1.63	2
HAC - LCA	39	5.08	10	1.34	1
HAC - no LCA	36	4.28	12	1.50	2
Single - LCA	44	14.93	5	3.40	1
Single - no LCA	40	16.71	9	3.69	1

Table 3: The number of times one hierarchy (subsumption or lexical) had a smaller path length and how much shorter the path length usually was. For each type of grouping (k-means, HAC, and single), a lexical hierarchy is compared to both a subsumption hierarchy created using LCA and one that did not use LCA.

documents in the hierarchies that contain no path to a relevant document are found in Table 2. These percentages are based on the number of relevant documents excluded compared to the total number of relevant documents.

6.1 Comparing topic hierarchies

The evaluation of the hierarchies based on the average path lengths reveals extreme differences among the methods used to create topic hierarchies. Using this measure, lexical hierarchies gave remarkably poorer results. When comparing clustered document groups, the subsumption hierarchy outperformed the lexical hierarchy for at least 34 queries. When the subsumption hierarchy had a smaller path, the difference in average path length was significantly greater than when lexical hierarchies had a smaller document path length. Exact results can be found in Table 3.

We performed ANOVA (ANalysis Of VAriance) on the twelve variations of hierarchies including random hierarchies. To linearize the data for the ANOVA, we performed a loglog transform on the average path length. All multiple comparisons were done using Tukey's Honest Significant Difference (HSD). Figure 7 shows how the hierarchies were ranked both absolutely and where significance was found ($p < 0.05$). It should be noted that TREC topic 305 was left out of all ANOVA analysis because there were no relevant documents in the document set. Because this gave a path length of zero, it would only add noise to the data. See Appendix A.1, Table 7 for ANOVA table.

From the ANOVA analysis, it can be seen how poorly lexical hierarchies perform at this task. The problem is that lexical hierarchies do not always create small document groups at its leaves. Since the average path length looks for a leaf cluster with the most relevant documents, it is more likely to pick larger document clusters, even though the algorithm chooses the smallest cluster from among the clusters with the most relevant documents. This factor also causes lexical hierarchies to perform worse than those that were randomly generated. Although random hierarchies consist of randomly related pairs, pairs are

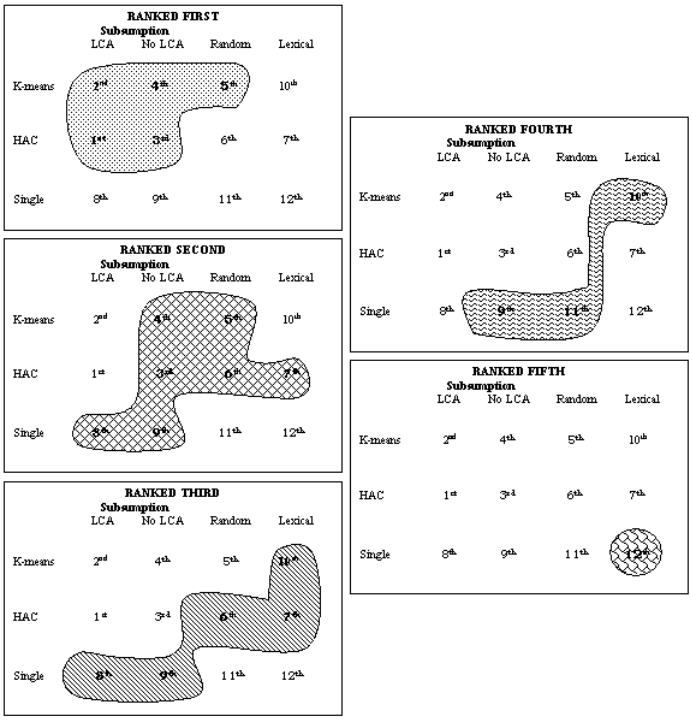

Figure 7: The groups indicate those hierarchies which ANOVA found indistinguishable for $p<0.05$.

still ordered on frequency of occurrence within document sets as in the subsumption hierarchies and documents are assigned to the hierarchy correctly. This means that clusters at the leaves are smaller for random hierarchies than they are for lexical hierarchies. Comparing lexical hierarchies to a random hierarchy that had similar leaf characteristics to the lexical hierarchy would yield more interesting results.

Figure 7 also shows that the clustered subsumption hierarchies are not significantly better than the random hierarchy when k-means clustering is used. Further analysis was done comparing the four types of hierarchies when k-means clustering is used. An ANOVA analysis revealed that the hierarchy with LCA is significantly better ($p < 0.00005$) than random; however, the hierarchy without LCA was indistinguishable, although slightly better ($p < 0.05$). The reason that random hierarchies performed nearly as well as subsumption hierarchies without LCA is that the random hierarchy still divide the document set up into smaller groups that enable the average path length to perform fairly well. It should be noted that the a comparison of parent-child pairs reveal almost no similarity between random and non-random subsumption hierarchies, as shown in Figure 8. The random hierarchy is dividing the document set differently, even though it is equivalent to subsumption hierarchies without LCA as far as the average path length is concerned. However, when all types of hierarchy groupings are combined, there is a significant difference between random and the two types of subsumption hierarchies. See Appendix A.2, Table 8 for ANOVA table.

When comparing the similarity of the relations used in the two types of hierarchies, there is very little overlap between subsumption and lexical hierarchies. For all comparisons of lexical hierarchies to subsumption hierarchies, there was less than a 10% similarity shown in Figure 8.

6.2 Effectiveness of clustering

Figure 7 shows that for almost all variations, clustering does significantly better than the single group hierarchy created in the same way. In fact, according to the ANOVA analysis that divided the data among

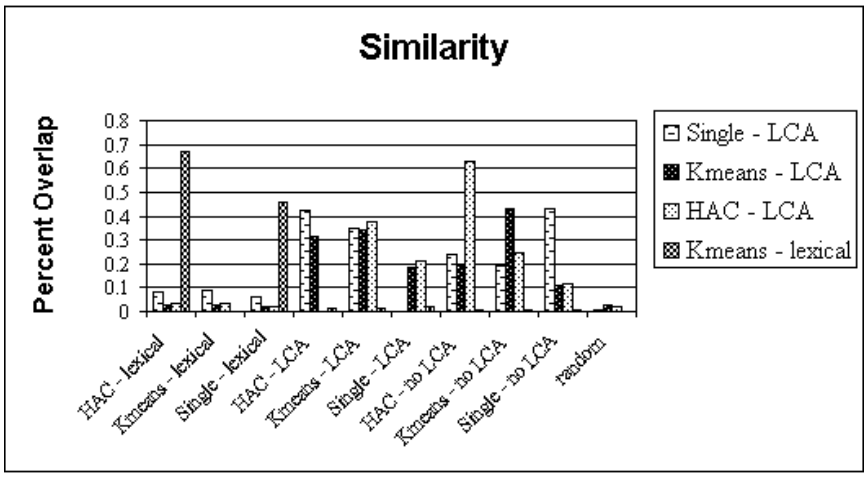

Figure 8: This represents the percent overlap when one hierarchy type is compared to another.

	Cluster		*Single*		
	Smaller	*Avg. difference*	*Smaller*	*Avg. difference*	*Equal*
K-means - LCA	42	3.04	7	0.82	1
K-means - no LCA	34	3.78	15	1.80	1
HAC - LCA	42	3.37	6	2.54	2
HAC - no LCA	36	3.81	11	4.95	3
K-means - lexical	45	13.21	3	2.07	2
HAC - lexical	44	13.41	3	2.88	3

Table 4: The number times a hierarchy (clustered or single) had a smaller path length and how much shorter the average path length was.

document grouping methods at a significance of $p < 0.01$, clustering outperformed a single hierarchy for both subsumption and lexical hierarchies. See Appendix A.2, Table 8 for ANOVA table. Clustered hierarchies had a shorter average path length a majority of the time and usually by a wider margin. These results are shown in Table 4.

When comparing the similarity of the single and clustered hierarchies, Figure 8 shows that single hierarchies have more overlap with clustered hierarchies than clustered hierarchies have with single hierarchies. This implies that clustered hierarchies discover more relations than single hierarchies.

6.3 LCA contribution

Figure 7 shows only insignificant differences between using LCA and not using LCA. When hierarchies were compared using ANOVA analysis of only other hierarchies that used the same document groupings, the only grouping that portrayed a significant difference ($p < 0.04$) was k-means. Both single and HAC revealed there was no significant difference for $p < 0.05$. Table 5 shows that the number of times that one hierarchy has a shorter average path length than another is more evenly distributed than in previous examples and the difference in the average path length is not as great.

When comparing the similarity of subsumption hierarchies created with the two variations, Figure 8 reveals that two types of hierarchies share at least 40% of the same relations, which is more than when hierarchies are compared across the different groupings of documents. In fact, 63% of the same relations are found when HAC is used. This is the most similarity of any comparisons of subsumption hierarchies.

The only significant difference between the two techniques is the number of relevant documents included in the hierarchy. For single hierarchies, over a quarter of the relevant documents are excluded. This is

	LCA		*no LCA*		
	Smaller	*Avg. difference*	*Smaller*	*Avg. difference*	*Equal*
K-means	25	1.29	21	0.34	4
HAC	29	1.92	18	0.39	3
Single	23	3.41	24	3.07	3

Table 5: The number times a hierarchy (created using LCA or not) had a smaller path length and how much shorter the average path length was.

	K-means		*HAC*		
	Smaller	*Avg. difference*	*Smaller*	*Avg. difference*	*Equal*
LCA	20	1.06	27	0.99	3
No LCA	23	2.43	25	1.48	2
Lexical	25	3.71	32	3.73	3

Table 6: The number times a hierarchy (clustered using k-means or HAC) had a smaller path length and how much shorter the average path length was.

particularly unsatisfying. However, when clustering is used this falls to a tenth of the relevant documents. This is still a fairly large number of relevant documents. Perhaps developing a hybrid of subsumption and lexical hierarchies would help with the inclusion of more relevant documents

6.4 Comparing clustering methods

The method of clustering also was found to have little effect on the quality of the hierarchies created. The variation of the number of relevant documents left out of a hierarchy was less than 2% for both. Figure 7 reveals very little difference based on the type of clustering used. In fact, there was no significant difference when ANOVA analysis was performed on clusters only. HAC was ranked first for each variation, but not with significance of $p < 0.05$. Table 6 shows the number of times that one hierarchy has a shorter average path length than another. It is very evenly distributed and the difference in the average path length is small.

The overlap found when comparing one clustering method to another is moderately high in all three of the variations used to generate hierarchies. Both methods of clustering find roughly the same number of comparisons. This is illustrated in Figure 9.

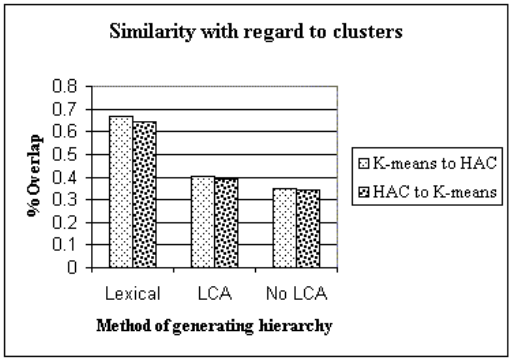

Figure 9: The amount of overlap between methods of clustering when the type of method used to generate the hierarchy is held constant.

7 Future Work

There remain many open research questions with regards to this work. One is how useful these hierarchies are for a person to find relevant documents. Given the example in Section 3, these hierarchies can be useful some of the time, but a user study would need to be conducted in order to evaluate the usefulness of the hierarchies in general.

A second question involves the integration of subsumption and lexical hierarchies. These two hierarchies emphasize different relationships. Each could be improved by using information gained from the other hierarchy. For example, the ordering of the second level of the hierarchy shown in Figure 3 should have had "Junk Fax" ranked higher so that it would not have been left out of the hierarchy when that particular level was truncated. Using the information in the subsumption hierarchy might have prevented this from happening. There might also be way to create a completely merged hierarchy.

Besides using the topic hierarchies to locate relevant documents, many other uses may be found. One way to utilize the hierarchies would be to use the exposed relationships in specific information tasks where characterizing a document set is necessary.

8 Conclusion

Hierarchies provide a convenient way to browse a document collection. Automatically generating a topic hierarchy brings to light information that is specific to the domain of the document set, as opposed to a manually generated hierarchy, which needs to suit all users and thus must be general. Given that the methods explored perform better using homogeneous document sets, clustering provides an alternative to using a ranked list, and also allows one to use these hierarchies in instances where a query is not present.

The evaluation metrics presented in this paper provide a way to begin evaluation of the hierarchies without requiring user input until well-formed hierarchies have been created, thus enabling a user study to yield less ambiguous results. The results presented in this paper show that subsumption and lexical relations are very different, exposing few of the same relations. The strength of subsumption lies in separating documents into small groups, whereas lexical hierarchies do a much better job of including all documents in the hierarchy. This research provides some of the foundation needed to continue developing hierarchies that best allow users to locate relevant documents.

9 Acknowledgments

We would like to thank Mark Sanderson, Jinxi Xu, and Mehran Semri for the use of their algorithms in this research. We would also like to thank Russell Swan for his help in the experimental analysis, and Craig Allen for his helpful comments on previous drafts.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. This material is also based on work supported in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Anick, P. (1999). *Automatic construction of faceted terminological feedback for context-based information retrieval*. Ph. D. thesis, Brandeis University.

Anick, P. & S. Tipirneni (1999). The paraphrase search assistant: Terminological feedback for iterative information seeking. In M. Hearst, F. Gey, & R. Tong (Eds.), *Proceedings on the 22nd annual*

international ACM SIGIR conference on Research and development in information retrieval, pp. 153–159.

Callan, J., W. Croft, & S. Harding (1992). The inquery retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pp. 77–83.

Crouch, C. (1988). A cluster-based approach to thesaurus construction. In *Proceedings on the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 309–320.

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

Cutting, D., D. Karger, J. Pedersen, & J. Tukey (1992). 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, pp. 318–329.

DHTMLAB. Dhtmlab. www.dhtmlab.com.

Fuhr, N., S. Hartmann, G. Lustig, K. Tzeras, G. Knorz, & M. Schwantner (1993). Automatic indexing in operation: The rule-based system air/x for large subject fields. Technical report, Technische Hochschule Darmstadt.

Jain, A. & R. Dubes (1988). *Algorithm for Clustering Data*. Engelwood Cliffs, N.J.: Pretice Hall.

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

Nevill-Manning, C., I. Witten, & G. Paynter (1999). Lexically-generated subject hierarchies for browsing large collections. *International Journal on Digital Libraries 2(2+3)*, 111–123.

Sahami, M. (1998). *Using Machine Learning to Improve Information Access*. Ph. D. thesis, Stanford University.

Salton, G. & M. McGill (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.

Sanderson, M. & B. Croft (1999). Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 206–213.

Sparck Jones, K. (1971). *Automatic Keyword Classification*. Butterworths.

van Rijsbergen, C. (1979). *Information retrieval* (second ed.). London: Butterworths.

Voorhees, E. M. & D. K. Harman (Eds.) (1997). *The Sixth Text REtrieval Conference (TREC-6)*. Department of Commerce, National Institute of Standards and Technology.

Willett, P. (1988). Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management 24(5)*, 577–587.

Xu, J. & W. Croft (1996). 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*, pp. 4–11.

YAHOO. Yahoo. www.yahoo.com.

A ANOVA analysis

A.1 ANOVA for all hierarchies

	DF	*SS*	*MS*	*F*	*P-value*
CONSTANT	1	384.03	384.03	5387.81237	0
qf	48	166.15	3.4615	48.56453	0
sysf	11	34.82	3.1654	44.41008	7.9515e-08
ERROR1	2037	145.19	0.071277		

Table 7: Summary for the ANOVA analysis for the comparison of all hierarchies. Model used is loglog.

Paired Comparison on all hierarchies, p = 0.05, HSD

|	-0.174	HAC, LCA
|	-0.167	K-means, LCA
| |	-0.129	HAC, no LCA
| |	-0.126	K-means, no LCA
| |	-0.0939	K-means, random
| |	-0.0387	HAC, random
| |	0.00613	HAC, lexical
| |	0.00795	Single, LCA
| | |	0.0331	Single, no LCA
| |	0.0773	K-means, lexical
|	0.15	Single, random
	0.453	Single, lexical

A.2 ANOVA for split hierarchies

	DF	*SS*	*MS*	*F*	*P-value*
CONSTANT	1	384.03	384.03	5332.29411	0
qf	48	166.15	3.4615	48.06411	0
clusf	2	25.197	12.598	174.92976	5.1514e-14
expanf	3	7.6792	2.5597	35.54247	7.989e-08
ERROR1	2043	147.14	0.072019		

Table 8: Summary for the ANOVA analysis for the comparison when data is split by document grouping (K-means, HAC, and single) and by hierarchy type (subsumption and lexical). Model used is loglog.

Paired Comparison on document grouping, p = 0.05, HSD

|	-0.0886	K-means
|	-0.0587	HAC
	0.147	Single hierarchy (no clustering)

Paired Comparison on hierarchy type, p = 0.05, HSD

|	-0.105	subsumption - LCA
|	-0.0662	subsumption - no LCA
	-0.000998	random
	0.172	lexical