Hierarchical Indexing and Document Matching in BoW

Maayan Geffet and Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University, 91904 Jerusalem, Israel

Abstract

Bow is an on-line bibliographical repository based on a hierarchical concept index to which entries are linked. Searching in the repository should therefore return matching topics from the hierarchy, rather than just a list of entries. Likewise, when new entries are inserted, a search for relevant topics to which they should be linked is required. We develop a vector-based algorithm that creates keyword vectors for the set of competing topics at each node in the hierarchy, and show how its performance improves when domain-specific features are added (such as special handling of topic titles and author names). The results of a 7-fold cross validation on a corpus of some 3,500 entries with a 5-level index are hit ratios in the range of 89-95%, and most of the misclassifications are indeed ambiguous to begin with.

1 Introduction

An obvious and natural approach to organize a large corpus of data is a hierarchical index – akin to a book's table of contents. The type of corpus we deal with is a bibliographical repository, with entries from a limited domain (our prototype is on "parallel systems"). Given such an index, it is desirable that search results point to relevant locations in the hierarchy, rather than just providing a flat list of entries. This is useful not only to support user searching, but also as an aid suggesting possible places to link new entries that are inserted into the repository.

1.1 BoW – Bibliography on the Web

The goal of the BoW project [9] is to create a convenient environment for using and maintaining an on-line bibliographic repository. The key idea is that this be a communal effort shared by all the users. Thus every user can benefit from the input and experience of other users, and can also make contributions. In fact, the system tabulates user activity, so merely searching through the repository and exporting selected items already contributes to the ranking of items in terms of user interest.

The heart of the BoW repository is a deep (multi-level) hierarchical index spanning the whole domain. The nodes in the hierarchy are called *concept pages*. Pages near the top of the hierarchy represent broad concepts, while those near the bottom represent more narrow concepts. The depth of the hierarchy should be sufficient so that the bottommost pages only contain a handful of tightly related entries (as opposed to Web search engines and scientific literature databases like CORA [5] which contain a relatively shallow directory). A subtrees containing all the concept pages reachable from a certain (high level) concept page is referred to as a *topic*. Entries can be linked to multiple concept pages, if they pertain to multiple concepts. Likewise, they can be linked at different levels of the hierarchy, depending on their breadth and generality.

The index is navigated using a conventional browser. Normally three frames are available (Fig. 1). The first shows the hierarchical index, and the currently selected concept page. The second lists entries linked to this concept page, and allows for the selection of a specific entry. the third displays the surrogate of the chosen entry, including all the bibliographical data (authors, title, where and when published), user annotations, and additional links (e.g. to where the full text is available). Available operations on the current entry include marking it for export, adding an annotation, and adding links. This includes links from additional concept pages to the entry, links between this entry and related entries (e.g. from a preliminary version of a paper to the final version), and links to external resources such as the full text.

The index structure is created by the site editor. The vocabulary used in the index and annotations is uncontrolled by the system, and users also query the system using natural language [2]. Indexing is simplified by the fact that we use concise surrogates, rather than full text documents [13]. We make up for the reduction in data by enlisting users to verify indexing suggestions. Thus, when a user introduces a new entry, the system uses the text of the entry as a query, and finds concept pages that contain similar entries. But the actual decision to link the new entry to these concept pages is left to the discretion of the user.

The indexing described in this paper is based on lexical analysis of concept pages and entries linked to them. For each topic, we create a list of keywords that differentiate it from other topics that have the same parent. The indexing then proceeds from the root, choosing the most suitable sub-topic(s) at each point. As only contending topics are considered, the complexity of the search is reduced [15, 19].

1.2 Related Work

There are three basic approaches for textual documents processing [14]: lexical, syntactic, and semantic analysis. A number of systems using syntactic and semantic analysis have been developed and are being used for research, such as DR-LINK [17], CLARIT [8] and TREC [7, 29]. However, they are typically not significantly better than the best lexical analyzers. We will discuss various lexical analyzers throughout the paper, in relation to our work.

Very little has been done so far on hierarchical indexing. In general, it has been shown that hierarchical indexing methods outperform traditional flat algorithms [19, 15]. How-

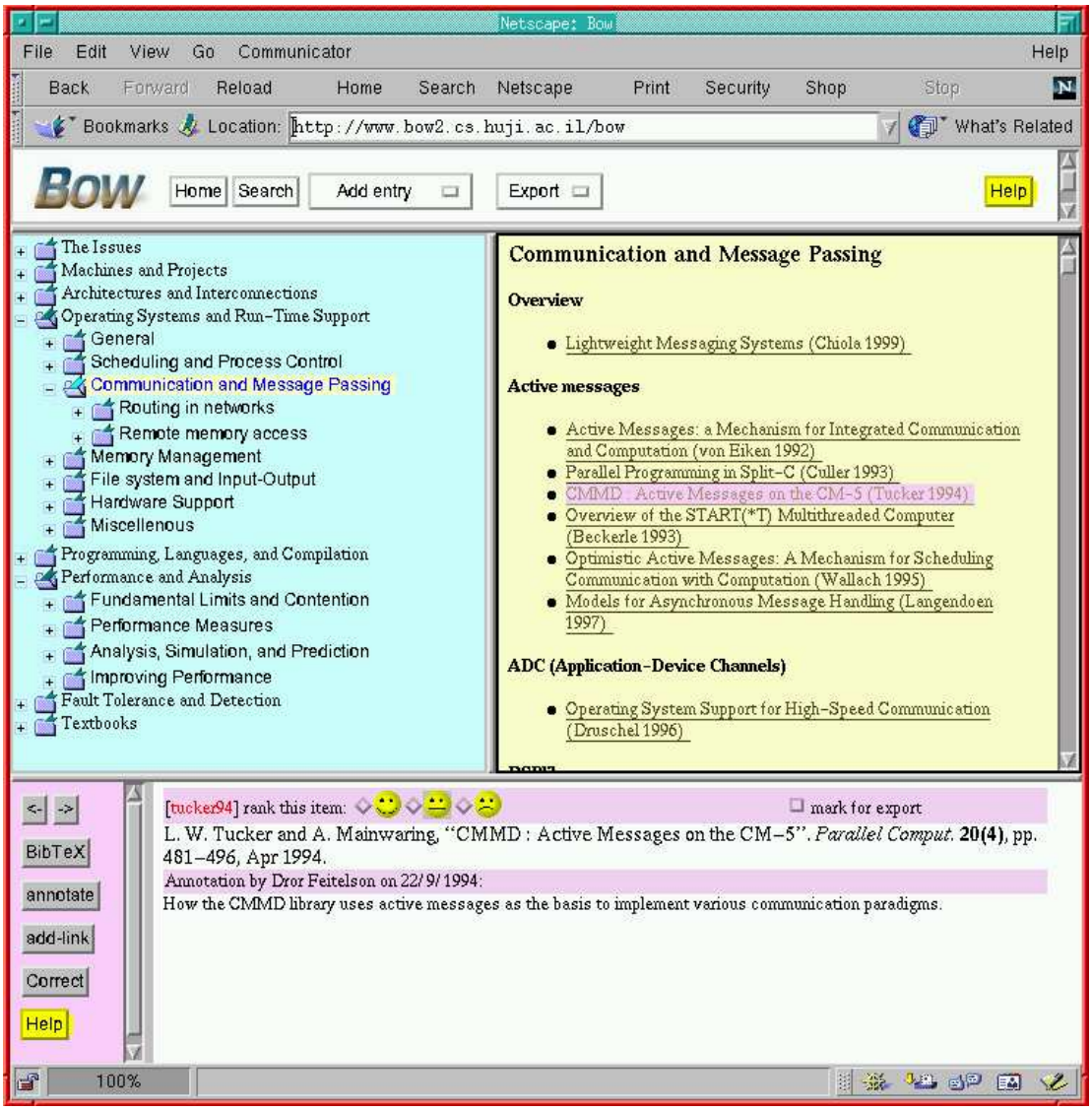

Figure 1: Screen dump of BoW showing partially opened hierarchical index.

ever, these studies were based on a very wide domain and a relatively shallow hierarchy (e.g. two levels). our work, in contrast, requires a very fine classification, as the bottom levels of the hierarchy only contain a small number of entries each.

2 Off-Line Preparation of Keyword Vectors

The hierarchical indexing mechanism consists of two parts. The first is an off-line traversal of the whole repository, repeated at regular intervals (e.g. once a day) in order to compute keyword vectors for all the topics. The second is a matching scheme that compares new entries or queries with these pre-computed keyword vectors.

Topic	Number of clusters	Hit ratio	
		5-grams	Whole words
Cooking recipes	10	87%	53%
Linux	16	85%	47%

Table 1: Clustering hits ratio for two given documents collections using 5-grams versus whole words.

The off-line part is executed recursively for every level of the index, top-down. The main idea is that each topic encompasses all the concept pages in a sub-tree of the index, therefore all of them should be taken into account while constructing its keywords vector. The group of sibling topics, located at the same level and having the same parent in the index are called a *competitive topics set*, since they compete for keywords with each other. The algorithm generates keywords vectors in five steps: *parse* all the pages in the topic's sub-tree, *merge* them into one vector, *unify* the resulting vectors to include the same words, *normalize* the weights of the words in all the vectors, and *choose* the most relatively frequent ones to represent the corresponding topics.

2.1 Parsing

The first stage is parsing the text of concept pages, with the goal of creating a vector of all the words in the given concept page [28, 31], denoted by Voc_{page}. This of course requires us to define "word".

The natural definition is a completely separated meaningful string. This has the well-known disadvantages of treating related words as being different, and the well-known solutions such as stemming (e.g. [22, 18]). An alternative is to use n-grams (substrings of length n of words: for example, "algorithm" will be turned into "algor", "lgori", "gorit", "orith", and "rithm") [1]. We prefer the latter, and specifically use 5-grams, based on a separate study[1] in which documents were clustered automatically based on similarity and this was compared with manual clustering (Table 1). But in order to avoid 5-grams that are largely based on common suffixes and therefore meaningless, we also use stemming first.

Note that longer words are represented by more 5-grams in the vocabulary vector than shorter ones, which gives them more weight in the comparisons. Thus it would be interesting to check if similar results would be obtained by using whole words, and weighting them according to length.

In any case, from now on the word "word" will mean a 5-gram.

[1] In cooperation with E. Boncheck.

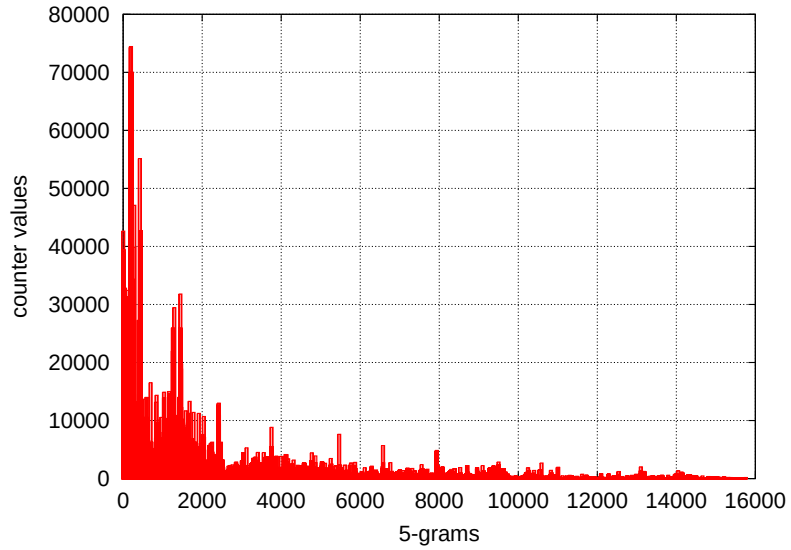

Figure 2: Example of counter values for 5-grams in the vocabulary of a top-level topic.

2.2 Merging

After parsing all the concept pages in a topic's sub-tree, the resulting vocabulary vectors are merged. The resulting vector includes the complete vocabulary of the topic: $Voc_{topic} = Voc_{page1} \cup Voc_{page2} \cup ... \cup Voc_{pagen}$. The counters indicating how many times each word appears are summed as described below.

2.3 Unification

In order to compare a query with a set of competitive topics, the vocabulary vectors of these topics must span the same space. We therefore create a unified vocabulary that includes all the words that appear in any of the competitive topics: $Voc_{compet-set} = Voc_{topic1} \cup Voc_{topic2} \cup ... \cup Voc_{topick}$. We then normalize the vocabulary vectors of the individual topics to include all these words, by adding the missing ones with a count of zero. The resulting normalized vectors will be denoted by $NormVoc_{topic}$.

2.4 Counters Normalization

In order to select meaningful keywords, we need to consider the number of times each word appears in each topic. As shown in Fig. 2, these values vary considerably. But they also suffer from the scaling effect problem [14]: the counter values in "small" topics are generally lower than in "big" topics, leading to an assignment of all the keywords to the bigger topics. To compensate for this, we need to normalize the counters based on the size of each topic.

The simplest approach is to divide the counter values by the total number of the words in the topic. However, according to Zipf's formula [32], $rank \times count \approx constant$ (where

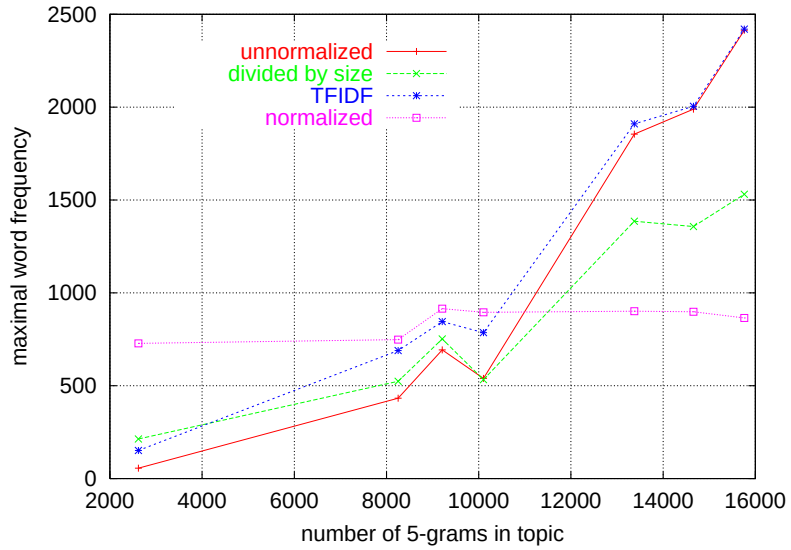

Figure 3: The influence of various normalization strategies on the 5-grams frequencies in the top level competitive set of 7 topics.

the words in the text are ranked in order of decreasing count), so the number of distinct words in the text grows much slower than their counts. Practically, about 50% of the regular text content consists of the same 250 words [14]. Therefore this method does not lead to good normalization (Fig. 3).

The most popular algorithm is TFIDF (Term Frequency Inverse Document Frequency) [25, 24, 6]. However, this technique does not take into account the frequency of term occurrences in other documents in the collection, based on the assumption that there are very many documents. In our case, we are trying to distinguish between a small set of topics, so as adjustment is needed. When applied directly, TFIDF did not produce good results (Fig. 3).

Our chosen approach is to normalize the counters on-the-fly during the previous three steps. Since we are interested defining a topic's vocabulary, words which occur frequently in one particular entry within it should not have a higher weight. Thus, we count each word only once for every entry containing it in the concept page. For example, given a topic with 5 entries, the maximal weight of a word is 5 if it appears in all the entries, but if it appears twice in one entry and three times in another, its weight will only be 2. This normalization is implemented as part of the parsing algorithm. To deal with the fact that concept pages have different sizes, the counters are further normalized by dividing by the number of entries in a page or topic. This is done as part of the merging and unification.

The comparative results of this method are illustrated in Fig. 3. As shown in the graph the maximal weights have reached the uniform distribution irrespective of the topic size.

2.5 Keywords Selection Heuristic

A keyword is a word that characterizes a concept and differentiates one topic from others [14]. Thus, in order to decide whether a word is a keyword of some topic, one should consider its frequency (weight) in this topic, and also compare with its weights in all the competitive topics. The basic idea is that if a word is extremely frequent in one particular topic and relatively rare in others, then we may use it as a keyword for this topic. If a word has similar weight in all the topics, then it does not represent any of them, even if its weight is high [27].

One way to assess the discriminatory power of a word is based on the difference between its maximal and minimal counter values in different topics in the competitive set. More formally, the algorithm is as follows (where $NormVoc_t(w)$ denotes the counter value for word w in the normalized vector of topic t):

1. For each topic in the competitive set, find those words that achieve their maximal counter value in this topic: $Max_t = \{w | \forall i,\, i \neq t : NormVoc_t(w) > NormVoc_i(w)\}$.
2. For these words, find the range of counter values: $\forall w,\, w \in Max_t$, $Dif(w) = max_{i \neq t}\{(NormVoc_t(w) - NormVoc_i(w))\}$.
3. sort the words in Max_t according to $Dif(w)$ in a descending order.
4. Choose the top 10% of the words (those with the biggest difference values) and place them in the keywords vector $Tkeys_t$.

A possible problem with this definition is that the difference can be large because the minimal value is very small. An alternative is therefore to use the difference between the two top counter values in step 2. The definition then becomes $Dif(w) = NormVoc_t(w) - max_{i \neq t}\{NormVoc_i(w)\}$. This version selects the words with significantly greater weight in one particular topic than in all the others, but may miss cases in which a word has a high count in 2 or 3 topics (which may happen as shown in Fig. 4). Specifically, in the BoW corpus the gap between the two highest values is the largest in 65-79% of the cases, but the gap between the 2nd and 3rd is the largest in another 15-22%.

Another disadvantage of this heuristic is the percentage of words to be chosen as the most significant: we decided to choose an empirically-determined 10% threshold, but maybe for other repositories it will be reasonable to use another threshold. An alternative is to choose the most significant words according to their statistics. Specifically, we propose to select those words whose counter value is larger than the average plus one standard deviation:

1. For each word calculate the average counter value: $average(w) = \frac{1}{n}\sum_{1 \leq i \leq n} NormVoc_i(w)$ (where n is the number of topics in the competitive set).
2. Calculate the standard deviation: $std\_dev(w) = \frac{1}{n}\sqrt{\sum_{1 \leq i \leq n}(NormVoc_i(w) - average(w))^2}$.

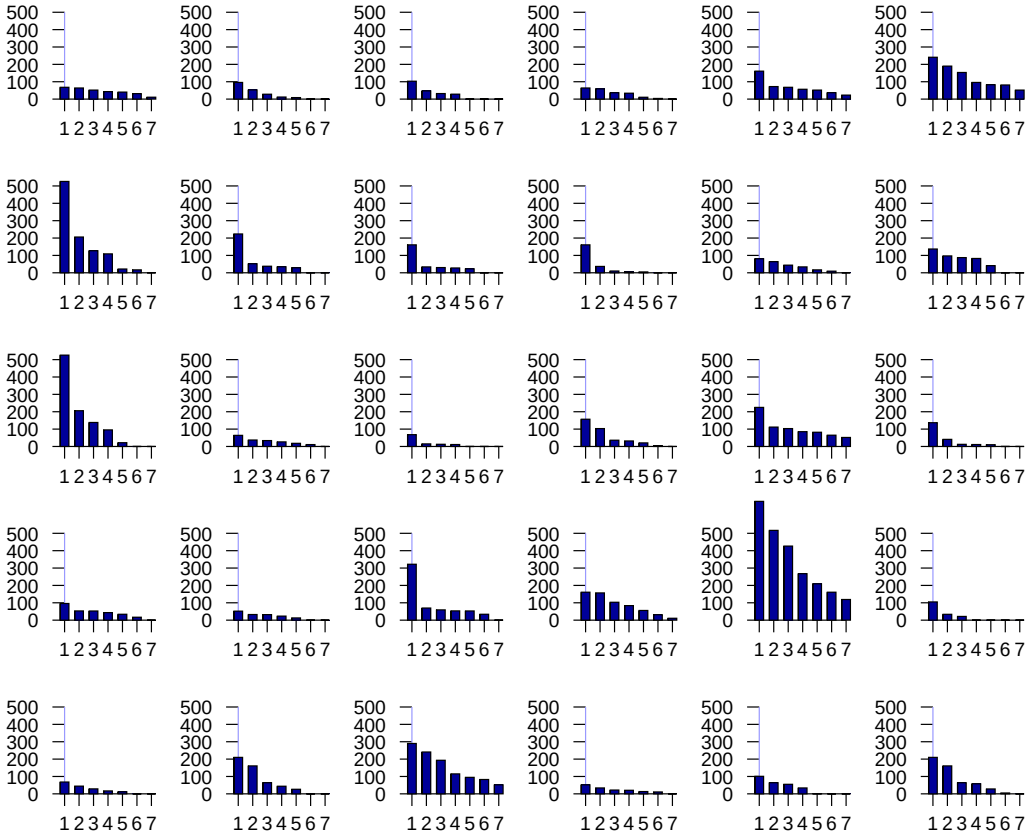

Figure 4: Distribution of word counts in the top level 7 topics for 30 selected 5-grams. The counters are sorted in a descending order.

3. If $max\_weight(w) > average(w) + std\_dev(w)$ then the word w is a keyword of the maximal weight topic, otherwise it does not represent any topic since it is almost equally frequent in all of them.

To check if the word should be a keyword for other topics as well, the highest value is removed and the procedure repeated for the remaining topics.

To compare the above heuristics we used them to classify 200 entries from the BoW prototype repository. The results are shown Table 2, and indicate that the last heuristic (using the average and standard deviation) is the best.

2.6 Optimizations

2.6.1 Stop-lists

A well-known optimization in classifications based on lexical analysis is the definition of a stop-list – a list of common words that should be ignored. In order to generate the list

Heuristic used	Hit ratio
extreme values differences	48%
two highest values differences	62%
if (max_weight ¿ avg+std_dev)	87%

Table 2: Correct classification rate when using alternative heuristics for keyword selection.

automatically, a threshold distinguishing the most common words should be found. Numerous studies of documents show that 30% of general English text encompassing millions of words is made up of only 18 distinct words [14]. Usually, stop-lists contain about 250-300 terms [30, 23, 10]. However, our repository is limited to a focused scientific domain, so its language is rather limited, and may vary among topics. Thus, the stop-list should contain only those words which are common in all the topics in the competitive set. This leads us to the following method for stop words identification: $stop\_list = \{w | \forall i, NormVoc_i(w) > \tau\}$, where τ is an empirically selected threshold on the counter values.

According to our observation of the common words values distribution, the upper value at the top level is greater than at lower levels. The best τ value is 80 for the highest level of the hierarchy, 60 for the second one, and 50 for the rest, where the average maximal counters are 320, 240 and 200, respectively. Thus the empirically obtained rule is that a stop-words lower bound threshold is a quarter of the average maximal frequencies for the given competitive set of topics.

Another interesting question is whether the stop-lists at various locations in the hierarchy will differ. It is reasonable to expect that words like "network", "software", and "language" will be important at the highest level of the hierarchy, since each of them leads to an appropriate broad topic, such as "architecture and interconnections", "operating systems and run-time support", and "programming, languages, and compilation" (see Fig. 1). Obviously, inside the topic "programming, languages, and compilation" the words "programming" and "languages" should be the first ones to go to the stop-list. However, our observation of Parallel Systems repository has shown that most of the stop-words at all the levels were the same, whereas for every lower level several additional common stop-words were added. The total number of stop-words is around 200 with slight differences for various competitive sets.

2.6.2 Special Treatment for Selected Fields

Another means for optimization is using domain-specific knowledge. In our case the domain is a bibliographical repository, which is classified into topics. Thus special fields like authors names and topic titles may carry special significance.

For example, the topics and sub-topics title fields may be expected to reflect the contents of the topic, and this is based on a semantic understanding by a human editor. It is therefore desirable to use these words as keywords, even if the counter-based algorithm described above does not recognize them as such.

The special treatment of author names is founded on the assumption that usually

scientists tend to concentrate their work in a rather narrow area of research. Therefore if several of the given author's publications appear in one specific topic of the competitive set, but not in the others, then it is sensible to suggest that the new article will also belong to this topic. As most of the author names appear too rarely and thus do not survive the keyword filtering process, special treatment is required. Just as in the case of topic titles, we simply treat author names explicitly as keywords. For this purpose, the first and last names are concatenated and treated as a single term.

2.6.3 Thesauri

The final major problem to be considered here is the use of similar or related terms (synonyms). Thus the use of thesauri in order to recognize variants or to control the vocabulary has been suggested [3]. A specific feature of our index is that it contains a lot of names of projects, systems, and tools, which are often referred to by acronyms. Text observations show that typically such terms occur in one of the following formats at least once:

1. The full term words with capital letters and then the acronym consisting of the same first capital letters in parenthesis.

2. The acronym is followed by the parenthesized full term words interpretation.

Based on this we developed a thesaurus-builder which is responsible for lexical text analysis and extracting the full expressions and their acronyms, and used it to construct a dictionary of acronyms. This was used during parsing to check if the acronym or its interpretation occur in any particular concept vocabulary, then if so it was explicitly entered into the keywords vector. User queries are also checked against the thesaurus, and expanded in a similar manner.

3 On-Line Searching

Given the keyword vectors for all the repository's topics, those matching queries can be found. This is done in two cases: when a user issues a search by specifying authors and/or keywords, and when a user inserts a new entry into the repository. In this latter case, the goal is to recommend topics to which the new entry may be linked.

An important goal is that a retrieved set will be of "reasonable" size — large enough to give the user a choice but not too large. BoW therefore doesn't retrieve a set of individual documents in response to a query. Instead, it returns whole concept pages. Moreover, if many of these concept pages belong to the same higher-level topic, that topic is returned rather than listing the lower level ones.

3.1 Matching and Ranking

Matching and ranking go together — we want to find the topics that match the query to the highest degree. Several methods for such ranking exist [20]. The most popular are

based on the TFIDF algorithm described in section 2.4 [24, 28, 26, 25] and will be rejected here for the same reasons. An alternative approach which is usually used in clustering (e.g. in Isodata Clustering) is to compute the distances between the keyword vectors. This can be applied in our case, by comparing the distances between the query vector and the competitive set vectors. However, the query is typically so short that it is not reasonable to weight its terms [11] so the terms relative frequencies distance between the query and the index vectors is not useful in our model. Thus we have to use a boolean ranking method [16], rather than a vector space algorithm.

Our matching process works as follows:

1. Check the query data against the acronyms thesaurus, and insert both acronyms and their full interpretation into the initially empty query vocabulary vector $QVoc$.

2. Parse the query (of new entry) and insert the resulting 5-grams into the vocabulary vector $QVoc$ (with no terms weights considerations).

3. Starting from the highest level topics, measure the similarity of the query to all topics in the competitive set by counting the number of common words in the vectors: $score_{topic} = \mid QVoc \cap TKeys_{topic} \mid$.

4. Select the topics with the highest score, and continue recursively to lower levels. The selection criterion is that the score be higher than the average plus a standard deviation, as was done in section 2.5. This gives good results because in 84%-91% of the queries the biggest gap is between the highest and the next topic, or between the second highest and the third one (Fig. 5).

Note that we don't examine all the tree branches, but only those which survive the filtering criteria, thus reducing the computational cost. This technique, called tree pruning, was also employed by others [15, 19], except that they choose only the single most suitable sub-topic at each level. The main disadvantage of such aggressive "single-path" pruning is that a failure at one of the higher levels will cause all the classification process to fail, whereas pruning that keeps two or three branches for further examination attains almost the same accuracy as full tree evaluation. Therefore, our ranking scheme does not suffer from the irrecoverable errors occurrence problem. Choosing more than one also meets our expectation that an article may refer to several categories in the bibliography.

3.2 Output Representation

Observe that the total number of selected topics may grow exponentially while descending the tree, if most subtopics are selected at each stage. To avoid showing the user such a long list of hits, we replace them all by their shared father. As the result, the more general (higher level) topic will be returned to the user. The condition for such output compression is that at least 50% of the particular topic's children and more than two of them are in the resulting list. The compressing routine is performed recursively from the

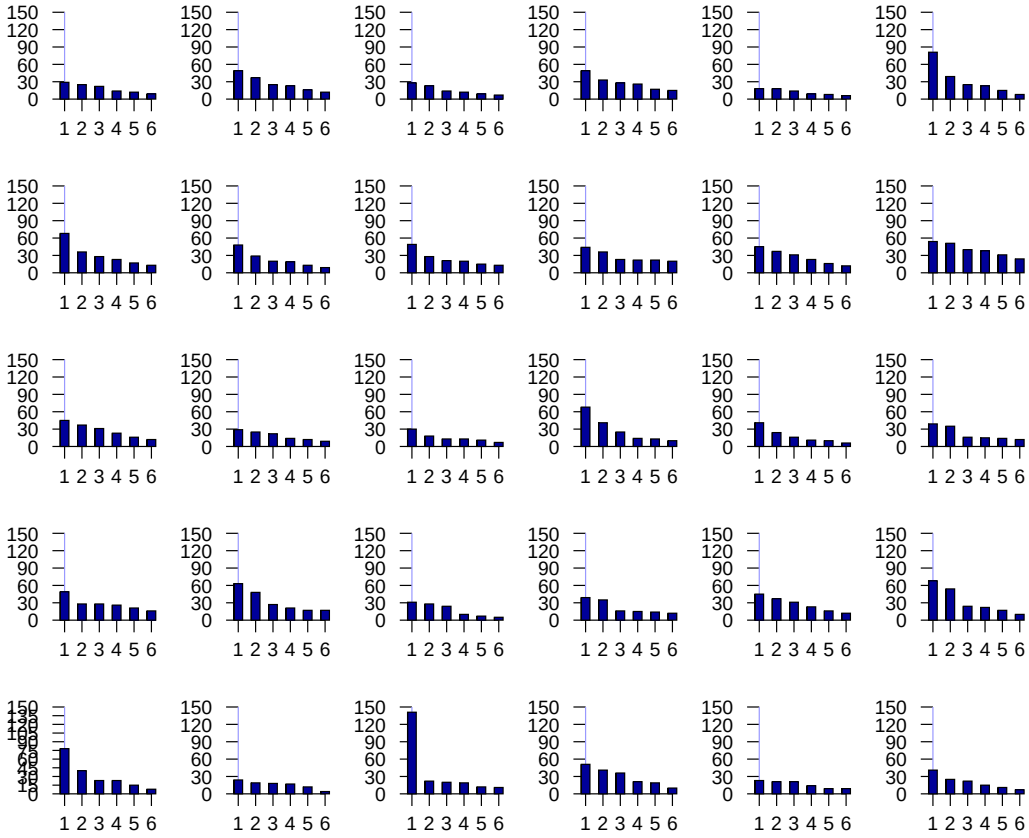

Figure 5: Distribution of topics scores (in a competitive set of 6 topics) for 30 sample queries. The topics are sorted in a descending order for each one.

bottom to the root of the index. The results of output compression are demonstrated in Fig. 6. The output was compressed for about 25% of the queries, where the majority of the compressed output sets were those including 14 links and more, only 10% of them remained untouched. On the other hand, only 10% of smaller sets (up to 13 links) were compressed. The compression ratio is quite big, and the size of compressed output sets was decreased by half in average.

Given the topics selected by the ranking process, and remaining after output compression, the question is how to display them on the screen. The dilemma is how to reconcile two contradicting considerations: keep both the concept pages' topological locations in the hierarchy (as in the Berkeley Cha-Cha Search Engine [4]), and their respective ranking with regard to this query (as is typically done in search engines, e.g. Northernlight [21]). Our solution is to display the original index tree, with the selected links opened and marked with different colors and font sizes according to their relevance to the query.

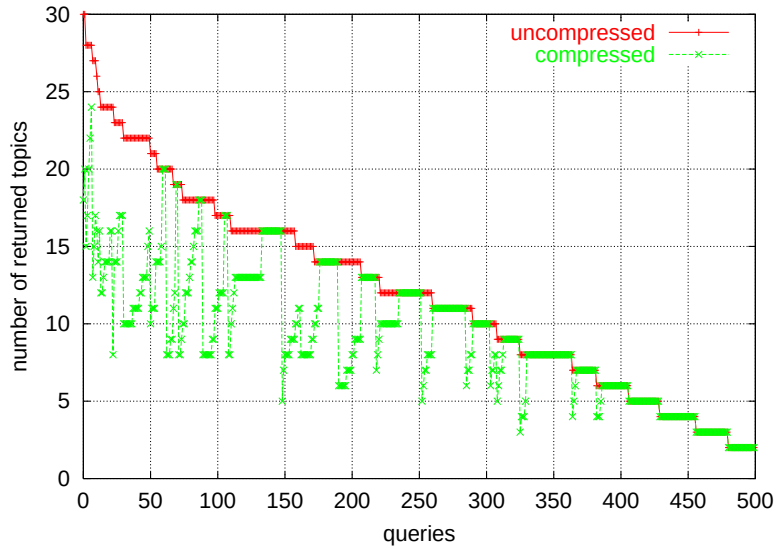

Figure 6: Results of the compression procedure for 500 queries from one of the 7-fold cross validation experiments (described below), sorted in descending order by the uncompressed output sets sizes.

4 Evaluation

In order to check the final algorithm performance we have conducted a sequence of 5 experiments employing 7-fold cross validation over a corpus of about 3,500 bibliographic entries. The corpus is focused on the domain of parallel systems, with an index that has an average depth of 5 and an average branching factor of 6. Every experiment was based on about 500 randomly chosen entries, which were extracted from the repository. The automatic off-line indexing was performed on the remaining 3,000 entries, and the resulting keyword vectors used to re-insert the 500 entries that were extracted. The hit ratio for each case was computed by comparing the algorithm's classification of these entries with their original manual classifications (Fig. 7). Manually checking those that were misclassified revealed that in many cases they were indeed ambiguous, and had very short annotations that only included very general terms.

Our experimental results have corroborated those of McCallum et al. that larger vocabulary sizes generally perform better. For larger branches of the index our algorithm selects more keywords, and the classification reached its highest accuracy (near 100%). For example, the "Operating Systems and Run-Time Support" topic, which is one of the biggest topics in the repository with over 7,700 distinct five-grams vocabulary, got 100% hit ratio, whereas "Algorithms and Applications" which is a smaller topic, containing about 3,700 keywords, attained only 92% hit ratio. Another evidence is the decrease in hits percentage for lower levels, due to the smaller number of entries and therefore the smaller number of keywords, as shown in Fig. 8.

Generally, the results indicate that the more information is available about each concept

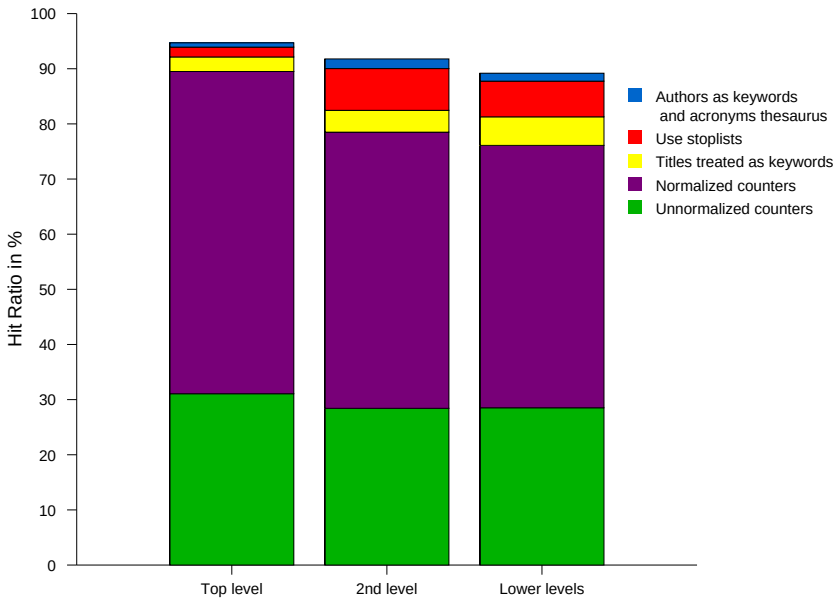

Figure 7: The hit ratios achieved at different levels of the index hierarchy, and how they depend on different parts of the classification algorithm.

and each query, the better the matching that is achieved. However, we find that even a relatively short annotation of 2-3 lines is enough for a reasonably good classification.

5 Conclusions

We have developed and presented the details of a data classification algorithm for effective concept-based storage and retrieval of scientific papers in multi-level hierarchical repositories. The three main features of the algorithm are its universality, scale independence, and self-updateability. The algorithm is universal in that it produces good results at all levels of the hierarchical index, and does not depend on the index depth. It is scale independent due to careful normalization of the keyword vectors, resulting in fair judgments for various-sized concept pages. It updates the keyword vectors regularly, thus keeping them current and adjusting to changes in the repository contents. This is done at selected intervals, rather than on-line for each new entry, because every local change in an individual concept page causes changes in the entire topic's vocabulary, and so in the selection of keywords across the entire competitive set; moreover, this effect can propagate up the hierarchy.

Results of experimentation with the BoW prototype repository on parallel systems are very promising. At the top level, nearly 95% of the entries were classified correctly, and this dropped to just under 90% for the lowest levels. Remarkably, this was achieved with only the entry details (mainly title and authors), and very short annotations typically between one and three sentences long. There was no access to or use of full text. The entries that

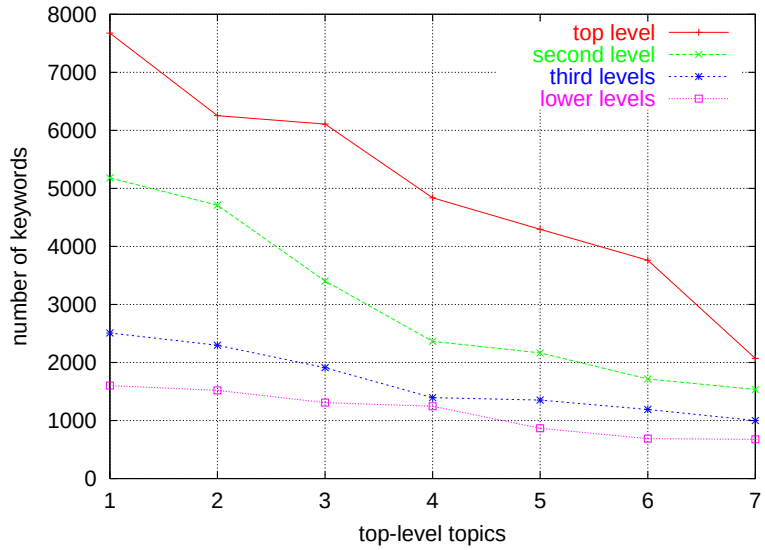

Figure 8: Distribution of keywords under the 7 top-level topics (which are sorted by size).

were misclassified were found to be ambiguous and had short or missing annotations.

In the future we hope to test our algorithm on additional repositories. Possible extensions include automatic construction of full thesaurus for all the words and phrases in the given corpus. A bigger challenge is automatic index creation from scratch. Our suggestion is to use one of the hierarchical clustering methods [12] combined with the described automatic indexing algorithm.

Acknowledgements

This work was supported by the Ministry of Science, Culture & Sport under the program to develop scientific and technological infrastructure.

References

[1] Adamson, G. and Boreham. 1974. "The use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles," *Information Storage and Retrieval*, 10, 253-260.

[2] Blair, David. C. 1990. *Language and Representation in information retrieval.* N.Y.: Elsevier.

[3] Chen, H., Martinaz, J., Kirchhoff, A., Ng, T. D., and Schatz, B. R.. "Alleviating search uncertainty through concept association: automatic indexing, co-occurrence analysis, and parallel computing." *J. Am. Soc. Inf. Sys.* 49(3), 216-206.

[4] Chen, M., Hearst, M. Hong, J. and Lin, J., 1999. "Cha-Cha: A System for Organizing Intranet Search Results". In *2nd USENIX Symp. Internet Technologies & Systems.*

[5] CORA - Computer Science Research Paper Search Engine, http://cora.whizbang.com

[6] Doszkocs, T.E. 1982. "From Research to Application: The CITE Natural Language Information Retrieval System," in *Research and Development in Information Retrieval*, eds. G. Salton, and H. J. Schneider, pp. 251-62. Berlin: Springer-Verlag.

[7] Dumais, Susan, T. 1995. "Latent semantic indexing (LSI): TREC-3 report." In *Overview of 3rd Text Retrieval Conference (TREC-3).* Donna K. Harman, ed. 1995. Washington, D. C.: Nist Special Publication.

[8] Evans, David A., Robert G. Lefferts, Gregory Gregenstette, S. Henderson, William Hersh, and A. Archbold. 1993. "CLARIT TREC design, experiments, and results." In *1st Text Retrieval Conference (TREC-1).* ed. Donna K. Harman, pp. 251-286. 1993. Washington, D. C.: Nist Special Publication, 500-207.

[9] Feitelson, D. G., 2000. "Cooperative Indexing, Classification, and Evaluation in BoW". In *7th IFCIS Intl. Conf. Cooperative Information Syst.,* O. Etzion and P. Scheuermann (eds.), Springer-Verlag LNCS Vol. 1901, pp. 66-77.

[10] Fox, C. 1990. "A Stop List for General Text". *SIGIR Forum.*

[11] Frakes, W. B., Baeza-Yates, R. eds. 1992. *Information Retrieval - Data Structures and Algorithms*, Englewood Cliffs, N. J.: Prentice Hall.

[12] Jardine, N. and C. J. vanRijsberngen. 1971. "The Use of Hierarchic Clustering in Information Retrieval." *Information Storage and Retrieval*, 7(5), 217-40.

[13] Kerner, C. J., and T. F. Lindsley. 1969. "The value of abstracts in normal text searching." In *The information bazaar: Proc. 6th Ann. Nat'l Colloq. Information Retrieval*, Philadelphia, pp. 437-440.

[14] Korfhage, R. R., 1997. *Information Storage and Retrieval*, N.Y.: John Wiley and Sons.

[15] Koller, D., and Sahami, M. 1997. "Hierarchically classifying documents using very few words". In *Proc. 14th Int'l Conf Machine Learning (ML-97)*, pp. 170-178, Nashville, Tennessee.

[16] Lee, Joon Ho, Myoung Ho Kim, and Yoon Hoon Lee. 1993. "Ranking documents in thesaurus-based Boolean retrieval systems." *Information Processing & Management* 30(1), 79-91.

[17] Liddy, Elizabeth D., and Sung H. Myaeng. 1993. "DR-LINK's linguistic-conceptual approach to document detection." In *1st Text Retrieval Conference (TREC-1).* Donna K. Harman, ed. Washington, D. C.: Nist Special Publication, 500-207, pp. 113-130.

[18] Lovins, J. B. 1968. "Development of the Stemming Algorithm." *Mechanical Translation and Computation Linguistics*, 11(1-2), 22-23.

[19] McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. Y. 1998. "Improving Text Classification by Shrinkage in a Hierarchy of Classes". In *Proc. 15th Int'l Conf Machine Learning (ML-98)*, Madison, Wisconsin.

[20] McGill, M. et al. 1979. *An Evaluation of Factors Affecting Document Ranking by information retrieval systems.* Project report. Syracuse, New York: Stracuse University School of Information Studies.

[21] The Northernlight Search Engine, http://www.northernlight.com.

[22] Paice, Chris. 1990. "Another stemmer." *SIGIR Forum* 24(1), 53-61.

[23] Salton, G., and M. McGill. 1983. *Modern Information Retrieval.* New-York: McGraw-Hill.

[24] Salton, G. and C. S. Yang. 1973. "On the Specification of Term Values in Automatic Indexing." *J. Documentation* 29(4), 351-72.

[25] Salton, G. 1971. *The SMART Retrieval System - Experiments in Automatic Document Processing.* Englewood Cliffs, N. J.: Prentice Hall.

[26] Salton, G. and C. Buckley. 1988. "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management* 24(5), 513-23.

[27] Salton, G., H. Wu, and C. T. Yu. 1981. "The Measurement of Term Importance in Automatic indexing." *J. Am. Soc. Inf. Sys.* 32(3), 175-86.

[28] Salton, G. and J. Allan. 1994. "Text retrieval using the vector processing model." *Proc. 3rd Ann. Symp. Document analysis and information retrieval*, Las Vegas, Nevada, pp. 9-22.

[29] Smeaton, Alan F., R. O'Donnel, and F. Kelledy. 1995. "Indexing structures derived from syntax in TREC-3: system description". In *Overview of 3rd Text Retrieval Conference (TREC-3).* Donna K. Harman, ed. 1995. Washington, D. C.: Nist Special Publication.

[30] vanRijsberngen, C. J. 1975. *Information Retrieval.* London: Butterworths.

[31] Wong, S.K.M., W. Ziarko. 1985. "On generalized vector space model in information retrieval." *Annals of the Society of Mathematics of Poland, Series 4: Fundamentals of information* 8(2), 253-267.

[32] Zipf, George Kinglsey. 1949. *Human behavior and the principle of least effort.* Cambridge, Massachusetts: Addison-Wesley.