Xerox TREC 3 Report: Combining Exact and Fuzzy Predictors

Hinrich Schütze   Jan O. Pedersen   Marti A. Hearst

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
{schuetze,pedersen,hearst}@parc.xerox.com

1 Overview

We employed different processing techniques for the ad hoc and routing tasks, although both explored the combination of exact and fuzzy predictors.

For the routing task, we used a neural net classifier to estimate the probability of relevance of a new document to a given topic description. We experimented with a variety of document representations, finally settling on a hybrid feature set that combined a selected number of discriminating terms with a low dimensional local LSI [9] for generalization.

In the ad hoc task, we applied some of our content analysis techniques, in particular, automatic thesaurus construction and automatic topical segmentation of full-length texts. Working from the hypothesis that exact term match can be too restrictive, we used *thesaurus vectors* [13] to compute *context vectors* for full texts and text segments. We also used thesaurus vectors to decompose the terms contained in the topic descriptions into sets of *query factors*, where each query factor is intended to represent a semantically distinct component of the topic description. These factors constrain document search by imposing Boolean constraints as will be described below. We used *TextTiling* [6] to partition long documents into semantically motivated multi-paragraph segments called *tiles*. We adopted the logistic regression methodology of Cooper et al. [2] and Fuhr et al. [5] to model the probability of relevance given measured attributes of a query-document pair. This allowed us to combine standard predictors (such as number of match terms) with various assessments of tile match, query factor score, and other predictors.

For both tasks we first performed standard preprocessing (document parsing, tokenization, stop-list term removal) using the TDB system [3]. Our *terms* consisted of single words and two-word phrases that occur over five times in the corpus (where phrase is defined as an adjacent word pair, not including stop words). This process produced over 2.5 million terms, which were further processed as described below.

In the remainder of this report, Section 2 describes our routing experiments, Section 3 describes our ad hoc experiments, and Section 4 discusses our results and possible future experiments.

2 Routing

Figure 1 sketches the system architecture for routing. Our routing method combines a neural network classifier and a hybrid representation based on terms and local Latent Semantic Indexing (LSI) vectors [4, 9].

Routing is a statistical classification problem, although the classifiers that are typically employed have relatively restricted generalization capacity (e.g., logistic regression [2] and other linear methods). We explored the use of a more powerful family of classifiers: neural networks. They support both linear and non-linear architectures and can model complex functions (e.g., Boolean functions of arbitrary complexity).

The input to the classifier consists of two blocks of 100 units, one block for 100 highly discriminating terms, and one block for a local LSI vector. This representation allows for both exact term match (from the discriminating terms) as well as a fuzzy similarity measure (from the local LSI vector). For long documents, the terms and vectors are computed only for one subpart of the document, the segment with the highest vector space score (see Section 3 for a description of the segmentation procedure).

The highly discriminating terms were selected using a chi-square test. We created a subcollection of the 2000 documents closest to the topic description according to the vector space score (as computed Buckley et al. (1994) [1]). The topic description was Rocchio-expanded before running the selection procedure [1]. Five copies of the (unexpanded) topic were added to the 2000 documents resulting in a subcollection of 2005 items. Terms that occurred at least five times within

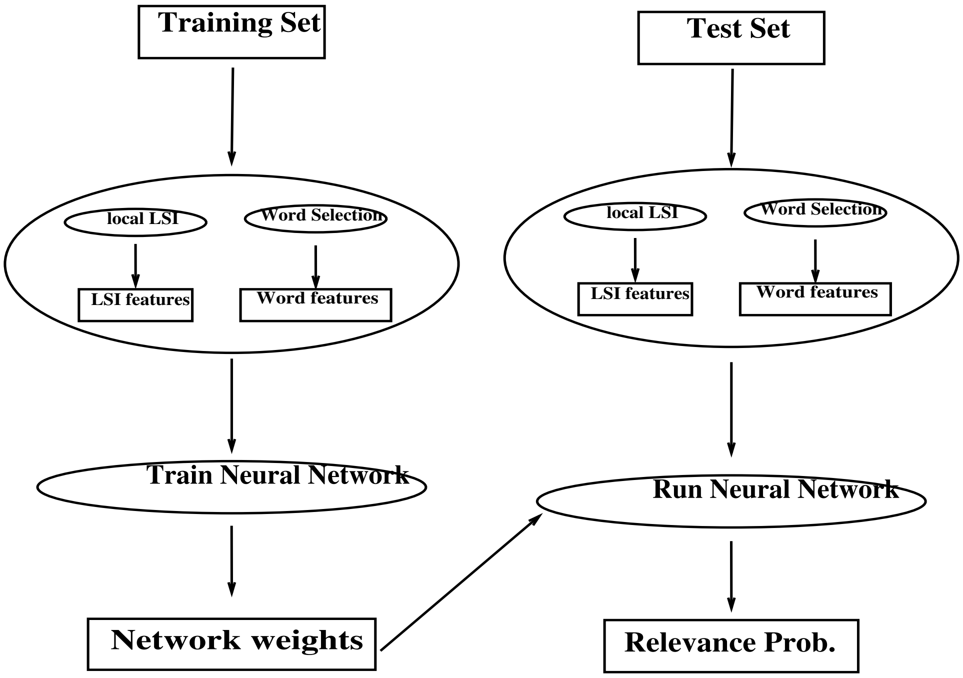

Figure 1: Architecture for routing.

this subcollection were ranked, using the chi-square test, according to how significantly they discriminated relevant from non-relevant documents. The top 100 terms were chosen as the input terms for the classifier.

We settled on chi-square as a selection criterion after initial experiments comparing it with terms selection according to raw frequency of occurrence and according to the ratio of relevant and non-relevant documents a term occurs in. These alternative measures were outperformed by the chi-square test.

Next, the top 1000 terms and the 2000 documents were used to define a 1000-×-2000 matrix that was decomposed using a singular value decomposition. The first 100 components for each document correspond to the second block of input units for the classifier.

For both linear and nonlinear networks, a different network is built for each topic description, and a single output unit indicates whether or not a given document is judged relevant to that network's topic. In the linear architecture, each input unit is connected to the output unit with no hidden units. This is roughly equivalent to logistic regression, although the fitting method differs (maximum likelihood for logistic regression versus backpropagation for the neural net). In the non-linear architecture, the input units are again connected to the output unit, but there is an additional layer of three hidden units between the input units and the output unit. Input units and hidden units are fully connected.

For each topic description, the classifier was trained with backpropagation [10] on two thirds of the 2000 preselected documents. The remaining third was used as a validation set. In order to avoid overfitting, the neural network was trained until the error on the validation set increased. To ensure that the classifier would be applied to a test set with a similar distribution as the training set, we considered only documents with a vector space score above a cutoff.

Table 1 gives results for the two classifiers and three input configurations on the TREC 2 data and TREC 3 data, respectively. For both collections, the best performance is achieved for a combination of terms and local LSI vectors. This result indicates that each of the two modes of representation, terms and local LSI, provides information about relevance that is not present in the other mode.

The results in Table 1 are less consistent with respect to the potential benefit of nonlinearity. The nonlinear classifier outperforms the linear one only in TREC 2 whereas it shows worse performance in TREC 3. A possible explanation is that the number of relevant documents for topics with few relevant documents is smaller in TREC 3 than in TREC 2. For example, 16 topics have less than 100 relevant documents in TREC 3. In contrast, there are only 6 such topics in TREC 2. It seems plausible that the kind of complex generalizations that a nonlinear classifier can learn will only be sound if there is enough training material, and this condition does not seem to hold in TREC 3.

Our routing results can be summarized as follows:

- Of the two document representations, terms and local LSI vectors, each contributes information that

TREC 2		TREC 3		
input	precision average	input	precision average	precision at 100
linear classifier		linear classifier		
terms	0.4253	terms	0.4109	0.473
local LSI	0.3901	local LSI	0.3963	0.473
terms + local LSI	0.4213	terms + local LSI	0.4207	0.484
nonlinear classifier		nonlinear classifier		
terms	0.4260	terms	0.4095	0.474
local LSI	0.3829	local LSI	0.3981	0.474
terms + local LSI	0.4359	terms + local LSI	0.4145	0.482

Table 1: Routing performance for TREC 2 (topics 51-100, 100 selected terms) and TREC 3 (topics 101-150, 200 selected terms). The term "average" refers to non-interpolated average precision. These results were obtained after the official NIST submission for TREC 3.

cannot be extracted from the other.

- There is evidence that a non-linear classifier performs better than a linear classifier if there is enough training information.

3 Ad Hoc

Documents are preprocessed for the ad hoc task by computing thesaurus vectors and segmenting long texts with TextTiling, topic descriptions are subdivided into query factors, and the various attributes (or predictors) are combined using logistic regression. This section describes this process in more detail. Figure 2 sketches the architecture for our ad hoc retrieval system.

3.1 Document Preprocessing

3.1.1 Thesaurus Induction

Our automatically constructed thesaurus maps each term to a vector that represents its pattern of cooccurrences with other terms throughout the document collection. The goal is to capture the similarity between terms as measured by the commonality of their neighbors. The technique is related to Latent Semantic Indexing [4], but relies on cooccurrence of terms rather than occurrence of terms in documents [12].

The thesaurus induction is motivated by a term-by-term matrix in which each entry of the matrix is a count of the number of times that the row term and the column term occur near each other in the corpus (where near is defined as within the same paragraph). To the extent that similar local context is a good measure of topical similarity, closeness in the row (or column) space of this matrix reflects topical similarity among terms. Since the full matrix is unworkably large, we approximate it in a staged computation. First, the matrix for a small subvocabulary of frequent terms is collected and the induced term similarity used to cluster these terms into basis classes. These basis classes are then used to compute a new matrix for an extended vocabulary. The process can be iterated until the full vocabulary is covered. The final matrix is reduced in dimensionality by the application of a singular value decomposition. For more details see [13].

Thesaurus vectors induce a similarity between terms which we use to factor queries. This similarity measure can be extended to sets of terms, and hence to documents, by averaging the included terms' vectors to form context vectors. Context vectors offer a fuzzy (or soft) match criteria for judging the similarity between a query and a document.

3.1.2 Motivated Segmentation

The second preprocessing step is concerned with the treatment of long documents (those comprised of more than two or three paragraphs). In previous work we have developed TextTiling, an algorithm for partitioning full-length text documents into coherent multi-paragraph units, called tiles [6]. These units are meant to represent the subtopic structure of the texts; according to evaluations done with human judges, subtopical discussions often span several paragraphs. TextTiling uses term frequency analyses to determine the extent of the tiles. The result of tiling is a representation of the original document augmented with tile boundary locations.

3.2 Topic Preprocessing: Factorization

A distinctive feature of our approach to the ad hoc task is that we overlay a conjunctive Boolean constraint on top of an additive similarity scoring scheme, which is intrinsically disjunctive in nature. That is, because the underlying scoring scheme is additive, there is no guarantee that a high scoring document will include a high

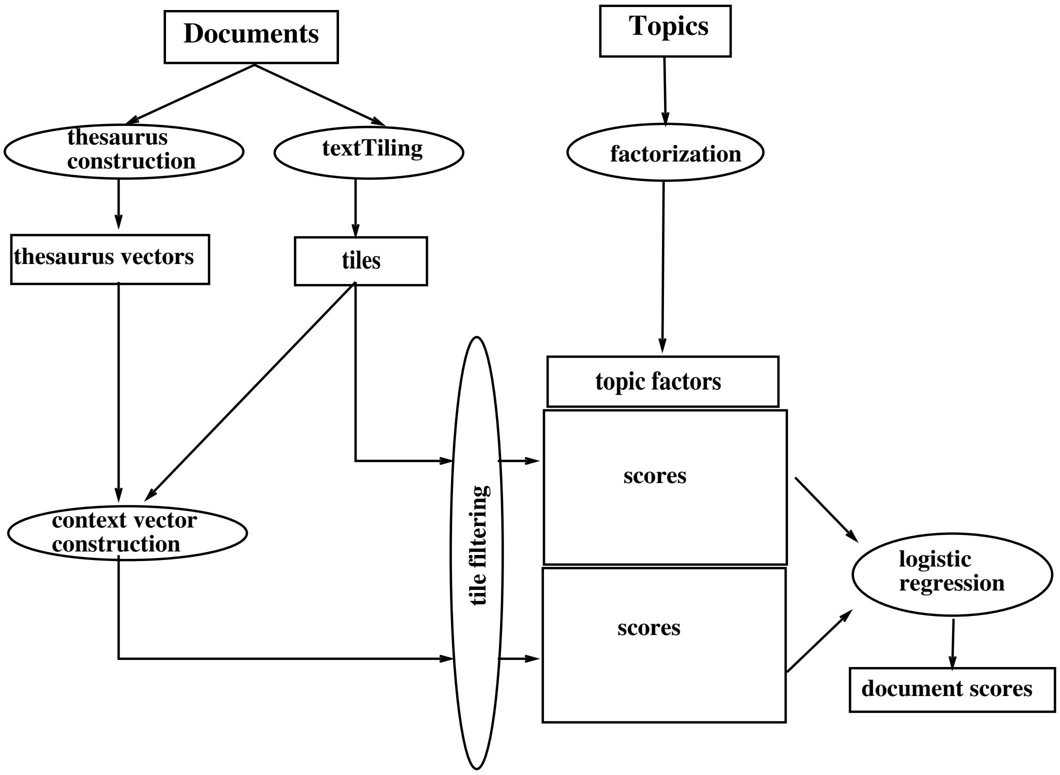

Figure 2: Architecture for ad hoc.

percentage of the terms in the query; it may be sufficient to include only a few with high intensity. This can be counterproductive if the meaning of the query requires disjoint sets of terms all to appear simultaneously. We explored both manual and automatic means to recover appropriate Boolean constraints. However, in the following we concentrate on automatic topic analysis since the manual experiment, described below, was unsuccessful.

Our automatic method, called topic *factorization*, subdivides the topic into a number of orthogonal *factors* or groups of words that are associated with independent themes of the topic. Factorization bears some similarity with extended Boolean retrieval [11] in that it combines Boolean and similarity-based retrieval. In contrast to [11], we induce the Boolean constraint automatically instead of relying on the user to define disjunctions or conjunctions. The key idea is to exploit the information about term relations in the thesaurus. Topics often contain several distinct groups of closely related terms. A case in point is Topic 51, which is about subsidies to the European aircraft industry. Two groups of terms or factors in this topic that are clearly delineated from each other and from other terms are terms related to subsidies and terms related to the aircraft industry. Consider two documents with the same number of terms matching Topic 51, the first with terms that are related to only one of these factors, the second with some terms related to the first factor and others related to the second factor. Intuitively, we want to give preference to the document that covers two of the crucial subthemes of the topic. (The importance of relating the themes of a query to the various subthemes of the documents being retrieved against are also discussed in [8].) In these experiments we attempted to achieve this by giving each document the retrieval status value of the factor that scores the lowest, thus imposing a conjunctive constraint on the query.

To find the factors of a topic automatically, words are represented by their thesaurus vectors and partitioned into a pre-determined number of classes via group-average agglomerative clustering. This step exploits the semantic information in the thesaurus to arrive at coherent groups of terms. Since topic descriptions in the TREC collection have similar length, we use a fixed number of four clusters per topic. For example, here are the four clusters for topic 51:

1. industrie airbus;industrie airbus aircraft;-consortium european;aircraft construcciones;-aeronauticas construcciones aerospatiale aeronauticas aeronauticas;s.a. british;aerospace boeing messerschmitt-boelkow-blohm messerschmitt-boelkow-blohm;gmbh mcdonnell;douglas mcdonnell aircraft;producer aerospace;plc s.a. aerospace aircraft u.s.;aircraft douglas plc producer consor-

input	TREC2		TREC3	
	precision		precision	
	average	at 100	average	at 100
documents	0.3018	0.444	0.2458	0.351
best tile	0.3038	0.452	0.2532	0.354
best tile + phrases	0.3154	0.469	0.2647	0.369
best tile + phrases + factors	0.3192	0.475	0.2718	0.379

Table 2: Scores for ad hoc runs on TREC 2 and TREC 3 data. These results were obtained after the official NIST submission for TREC 3.

tium gmbh

2. government u.s.;government dispute subsidy sanction federal;subsidy government;assistance agreement retaliation finance aid;loan aid dispute;trade loan controversy assistance;aid group

3. trade;dispute european;government trade;tension trade;controversy gatt european airbus;subsidy u.s. spanish;government trade;policy trade;gatt german aircraft;code british trade anti-dumping german;-british french anti-dumping;duty spanish tension french;german

4. duty federal assistance objection petition policy complaint countervail cite countervail;duty duty;-petition tariff loan;finance document narrative code

Although factor 4 is fairly nondescript, the other clusters do correspond to major themes present in the topic (which is about government subsidies to Airbus and the implications for U.S. companies): the aircraft industry (factor 1), government subsidies (factor 2), and international trade (factor 3). The intuition is that a relevant document should score highly against all selected query factors simultaneously. This imposes a conjunctive constraint that would otherwise not be present.

3.3 Combining Predictors

To combine scoring information for factors and tiles we adopted a probabilistic retrieval approach that models the logit of the probability of relevance given the query as a linear function of the available predictors [2, 5]. The model was fit using topics 51–100 as a training set and tested on topics 101–150 (the validation set). Since running the logistic regression on all 750,000-×-50 document-topic pairs is not feasible, we selected a subset of *privileged* tiles: for each topic description, only those tiles with the 2000 highest vector space scores were retained. The privileged tiles were used to select documents (or portions of documents) for further processing in each of the following ways:

- documents that contain at least one privileged tile
- the best privileged tile of a document
- the aggregate (= concatenation) of all privileged tiles of a document

Factors were used in the computation of *minmax* predictors: the minimum score for the n "best" factors of a topic. The "goodness" of factors was assessed either manually or automatically. In the manual method, one of us simply ranked the factors according to his intuition about how important they were relative to each other. In the automatic method, factors were ranked according to "soft" correlation with the center of the thesaurus space, i.e. according to the correlation of the context vector of the factor (the sum of the thesaurus vectors of its terms) and the vector representing the center of the space occupied by the thesaurus. The less descriptive a factor, the closer it is to the center of the thesaurus space. Terms like "outcome" and "attempt", that don't have strong topical characteristics, cluster near the center of thesaurus space. In other words, the more distant a factor's terms were from the thesaurus center, the higher it was ranked. The center of the thesaurus space was computed as the centroid of the 10,000 most frequent terms.

Given the privileged tiles and the ranked factors, the following predictors were computed in the experiments. Except for the first two, each was computed separately for the document as a whole, for the best tile (where best is defined according to the vector space score), and for the aggregate of the best tiles. We also computed these predictors separately for words and phrases.

1. number of tokens in query
2. number of tiles in aggregate
3. number of tokens (in document, tile or aggregate)
4. number of matches with query
5. sum over all term matches: square root document frequency (the occurrence count of that term in the document, tile or aggregate)
6. sum over all term matches: product square root query frequency (the occurrence count of the term in the query) and square root document frequency

pred. prob.	# matches:	2.50	5.50	7.00	9.00	12.00	29.00
63	emp. prob.	0	38	46	40	37	31
	# documents	2	64	254	800	1600	2326
24	emp. prob.	43	26	23	23	20	30
	# documents	51	296	1131	2079	2578	1301
16	emp. prob.	27	21	20	16	15	28
	# documents	177	1033	1876	2777	2632	998
11	emp. prob.	19	18	15	11	9	28
	# documents	605	2169	3290	3127	2007	797
8	emp. prob.	13	12	9	6	6	25
	# documents	2663	5266	4545	2563	1210	536
3	emp. prob.	3	4	3	1	1	11
	# documents	35116	9225	3205	1034	410	257

Table 3: Predicted and actual probability of relevance. Documents are binned according to number of matches and predicted probability. For each cell, the empirical probability and the number of documents in the cell are given.

7. sum over all term matches: product square root document frequency and log collection frequency (the number of documents the term occurs in)
8. minimum of soft correlations with 3 best factors (automatic selection)
9. minimum of soft correlations with 3 best factors (manual selection)

We ran extensive tests to find the optimal combination of predictors over the validation set. In general, it was found that adding predictors to the basic Cooper-style predictors (5–7) did not improve performance. Since many of the predictors are highly correlated (e.g., predictor 5 for tiles and for documents capture similar information), this result is not surprising in many cases. For some combinations we were disappointed by the failure of predictors to make a difference. For example, the manually selected factors did not do better than the automatically selected ones.

Table 2 displays results for predictors 5–7 for documents and separately for best tiles. Best tiles perform slightly better than documents. We found that simply adding predictors 5–7 calculated for phrases instead of words did not improve performance. However, if we fitted models separately for words and phrases and then combined them in a second-stage regression we did achieve better results (as quoted in the table). Finally, if we added predictor 8 to the second-stage regression mentioned above we achieved another improvement in performance.

4 Discussion and Future Work

In analyzing the ad hoc results, we consider three avenues of investigation:

- What do individual predictors contribute to the estimation?
- To what extent does the regression model estimate the probability of relevance correctly?
- What kind of interaction takes place between query factors and tiles?

To investigate the first two questions, we have found it useful to bin the set of documents according to their value on a particular predictor and according to estimated probability.[1] Table 3 gives an example from the "best tile" results of Table 2. The estimated probabilities of all 100,000 documents used in training (2000 for each of the 50 queries), are binned into six bins such that each bin has the same number of relevant documents. Similarly, the same procedure is applied to the number of matches of these documents, i.e. six bins are constructed such that each bin has the same number of relevant documents. The cross-product of the two binnings gives us 36 cells for which we can compute the actual probability of relevance by dividing the number of relevant documents in the cell by the total number of documents in the cell. Table 3 gives for each bin the center of the interval covered by it (probability for rows, number of matches for columns), and, for each cell, the empirical probability of relevance and the number of documents.

Our regression model assumes a linear relationship between predictor and logit of probability. In the lower rows of Table 3, this relationship holds to a good degree of approximation for documents with up to 12 matches. However, the assumption of linearity is grossly inadequate for the "outlier" bin labeled '29.00' (the last column). Since regression analysis is sensitive to outliers,

[1] This method of analysis was suggested to us by John Tukey.

this is an important problem to investigate. It may be the case that for this particular predictor, the model should be modified to include a transformation of the predictor "number of matches" or a special processing step for documents that have high match scores.

As for the second question (accuracy of estimation), the table shows that high probabilities are overestimated (top row) and that low probabilities tend to be underestimated (rows corresponding to predicted probabilities "11" and "8"). The incorrect estimation of the high probabilities is especially problematic since the evaluation measure of average precision depends mainly on the top part of the document ranking. The reason the top part of the ranking is less accurately estimated than the bottom part seems clear: there are many more non-relevant than relevant documents so that the logistic regression will concentrate its "capacity" on the low-probability end. To address this problem we plan to experiment with cost functions that are more closely tied to the evaluation measure.

For the third question, we are in the process of using a new user interface paradigm, called TileBars [7], to assess in more detail how the terms in the query factors are distributed in the longer texts. We suspect that the characteristics of the best subset of tiles to represent a long document's contents will vary with each query; for example, for some topic descriptions the best tile will work well, but for others, the set of tiles with the most query term overlap might be most appropriate.

Acknowledgments. We would like to thank David Hull and John Tukey for discussions of the material presented in this report and Mike Berry for making SVDPACK available to us.

References

[1] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART: TREC 2. In Donna Harman, editor, *The Second Text REtrieval Converence (TREC-2)*. U.S. Department of Commerce, Washington DC, 1994. NIST Special Publication 500-215.

[2] William S. Cooper, Fredric C. Gey, and Aitoa Chen. Probabilistic retrieval in the TIPSTER collections: An application of staged logistic regression. In D.K. Harman, editor, *The Second Text REtrieval Converence (TREC-2)*, Washington DC, 1994. U.S. Department of Commerce. NIST Special Publication 500-215.

[3] Douglass R. Cutting, Jan O. Pedersen, and Per-Kristian Halvorsen. An object-oriented architecture for text retrieval. In *Conference Proceedings of RIAO'91, Intelligent Text and Image Handling, Barcelona, Spain*, pages 285–298, April 1991. Also available as Xerox PARC technical report SSL-90-83.

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] Norbert Fuhr and U. Pfeifer. Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions. *ACM TOIS*, 12(1), Jan 1994.

[6] Marti A. Hearst. Multi-paragraph segmentation of expository discourse. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics*, June 1994.

[7] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *To appear in Proceedings of CHI 1995*, Denver, CO, May 1995.

[8] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 59–68, Pittsburgh, PA, 1993.

[9] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR '94*, pages 282–289, 1994.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations.* The MIT Press, Cambridge MA, 1986.

[11] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[12] Hinrich Schütze. Dimensions of meaning. In *Proceedings of Supercomputing '92*, 1992.

[13] Hinrich Schütze and Jan O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIAO*, pages 266–274, Rockefeller University, New York, 1994.