SIGIR-94

Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval

Yiming Yang

Section of Medical Information Resources
Mayo Clinic/Foundation
Rochester, Minnesota 55905 USA

Abstract

Expert Network (*ExpNet*) is our new approach to automatic categorization and retrieval of natural language texts. We use a training set of texts with expert-assigned categories to construct a network which approximately reflects the conditional probabilities of categories given a text. The input nodes of the network are words in the training texts, the nodes on the intermediate level are the training texts, and the output nodes are categories. The links between nodes are computed based on statistics of the word distribution and the category distribution over the training set. *ExpNet* is used for relevance ranking of candidate categories of an arbitrary text in the case of text categorization, and for relevance ranking of documents via categories in the case of text retrieval. We have evaluated *ExpNet* in categorization and retrieval on a document collection of the MEDLINE database, and observed a performance in recall and precision comparable to the Linear Least Squares Fit (LLSF) mapping method, and significantly better than other methods tested. Computationally, *ExpNet* has an $O(N \log N)$ time complexity which is much more efficient than the cubic complexity of the LLSF method. The simplicity of the model, the high recall-precision rates, and the efficient computation together make *ExpNet* preferable as a practical solution for real-world applications.

1 Introduction

A major problem in retrieval of natural language texts is the vocabulary differences between humans. Related concepts can be represented by a variety of words. Therefore, retrieval methods based on shared words in queries and texts, which we call "surface-based matching", can be effective only if the authors who write the texts and the users who issue the queries happen to choose the same words for the related concepts; otherwise, poor retrieval results are unavoidable. A common technique for handling the vocabulary differences in practical databases is to use subject categories to index the content of texts. For surface-based matching methods, however, the problem of text categorization is not any easier than the problem of text retrieval, because the vocabulary gap between free texts and indexing categories may be even larger than the vocabulary gaps among the texts themselves. Manual assignment remains the dominant method of text categorization in practical databases. MEDLINE, for example, the world's largest and most frequently used on-line bibliographic database, spends over two million dollars per year for the manual categorization of about 350,000 new entries [1]. There is thus a strong motivation for automatic or semi-automatic text categorization.

It has been recognized that human knowledge is crucial for solving the vocabulary gap problem. The human knowledge can be either invoked from predefined terminology thesauri, or automatically learned from training samples where the related texts of different vocabularies are assigned by humans. A well-observed problem with thesaurus-based matching is that general-purpose thesauri often do not have sufficient vocabulary coverage crossing different applications. Furthermore, subjective decisions in thesaurus development often do not reflect the context sensitivity of word meanings and the application dependent relationship between words. As a result, no reliable improvements have been observed by using general-purpose thesauri compared to surface-based matching [2] [3] [4] [5]. In contrast to relying on human developed thesauri, learning the mapping between different vocabularies from training samples ("example-based matching") has the potential advantage to objectively reflect the application dependent features of word usage. Example-based matching methods include the relevance feedback approach [6], the Darmstadt Indexing Approach (DIA) [7], the Linear Least Squares Fit (LLSF) mapping [3][8], Bayesian belief networks [9] [10], and neural networks [11].

Relevance feedback has the user indicate texts related to a query in an initial retrieval based on surface matching. It then adds the words in the related texts to the query in order to reduce the

vocabulary gap between the query and the texts which are not retrieved by the initial query. Relevance feedback has shown significant improvement over surface-based matching and thesaurus-based matching in many cases of text retrieval. However, the limitation is that it requires relevance information for each query, and cannot use the information for predicting the answers to different queries. Such a restriction makes relevance feedback only rarely applicable to text categorization. For example, most documents (abstracts of articles including titles) in a bibliographic database differ from each other, so documents previously categorized by humans cannot be useful for relevance feedback in the categorization of new ones.

The DIA system probabilistically learns term-descriptor associations from manually indexed documents, where "term" means a word or phrase in a document, and "descriptors" are predefined indexing units in the DIA dictionary. DIA uses a technique known as "the least squares polynomial" (LSP) to obtain the term-descriptor associations. The LLSF mapping method is similar to the DIA/LSP approach, in the sense that it also learns from manual assignments of text categorization, and uses a least squares fit technique. A common feature of DIA/LSP and LLSF is that both allow training documents and testing documents to be different, so previously indexed documents can be used to index new documents. A major difference between LSP and LLSF is the context-sensitivity of the mapping. The term-descriptor associations in LSP are computed based on the occurrences of term/descriptor pairs in the training sample. The word-category associations in LLSF are computed based on the occurrences of word-combination/category-combination pairs. The LLSF mapping learned this way is therefore context-sensitive. Further discussions regarding this aspect can be found in a previous paper [12]. While the difference between LSP and LLSF in text categorization is rather subtle, the difference in text retrieval is more fundamental. LSP determines the relevance between a query and a document based on the shared terms, and ignores the non-shared terms. That is, it uses relevance judgements by humans to weight shared terms, not to connect the non-shared terms which are conceptually related. LLSF, on the other hand, uses the relevance judgements to establish connections between the query vocabulary and the document vocabulary (either free words or indexing categories), so the information within non-shared words are also effectively used in the retrieval. In our evaluations of text retrieval, LLSF has shown a performance comparable to relevance feedback, and significant improvements over surface-based matching and thesaurus-based matching [8]. In the evaluations of text categorization (where relevance feedback is not applicable), LLSF has significantly outperformed surface-based matching and thesaurus-based matching [3] [8] [13]. The current problem with LLSF is the cubic time complexity of training, which makes it expensive to apply this method to very large data collections.

Bayesian belief networks and neural networks have also been studied for incorporating human assessments in solving the vocabulary differences. Haines and Croft [9] implemented relevance feedback on simplified inference networks for text retrieval; Tzeras and Hartmann [10] proposed a Bayesian belief network approach to text categorization. The Haines approach inherits the limitation of relevance feedback, i.e. human assessment must be given for every query, and predicting answers for different queries is impossible. Tzeras' evaluation showed a less effective performance of the belief network than the LSP approach, while the computation cost was far higher than LSP. Wong and Cai proposed a neural network [11] for learning term associations, and demonstrated its use on a small retrieval data set (32 queries in the training set, 4 queries in the testing set, and 82 documents in total). The effectiveness of this method on larger text collections is not yet clear, and the intensive computation remains a potential problem when adopting it to real world applications.

Next we will introduce a new example-based approach to text categorization and retrieval, the Expert Network. This method is as effective as the LLSF mapping, and has a time complexity of $O(N \log N)$, far more efficient than LLSF, Bayesian belief networks and neural networks.

2 Method

ExpNet can be applied to either text categorization or text retrieval. We will first describe the model for text categorization, and then introduce its use in text retrieval by an extension. The task of text categorization is to assign the related categories to texts based on their contents. Our strategy is to automatically learn the text-to-category mapping from human assignments. Texts categorized by experts are undoubtedly a rich resource for learning, and such training data are available in most databases. While obtaining the training data is not difficult, the crucial question is how to use a finite training set to predict the categories of arbitrary texts. A text requiring categorization may be different from the

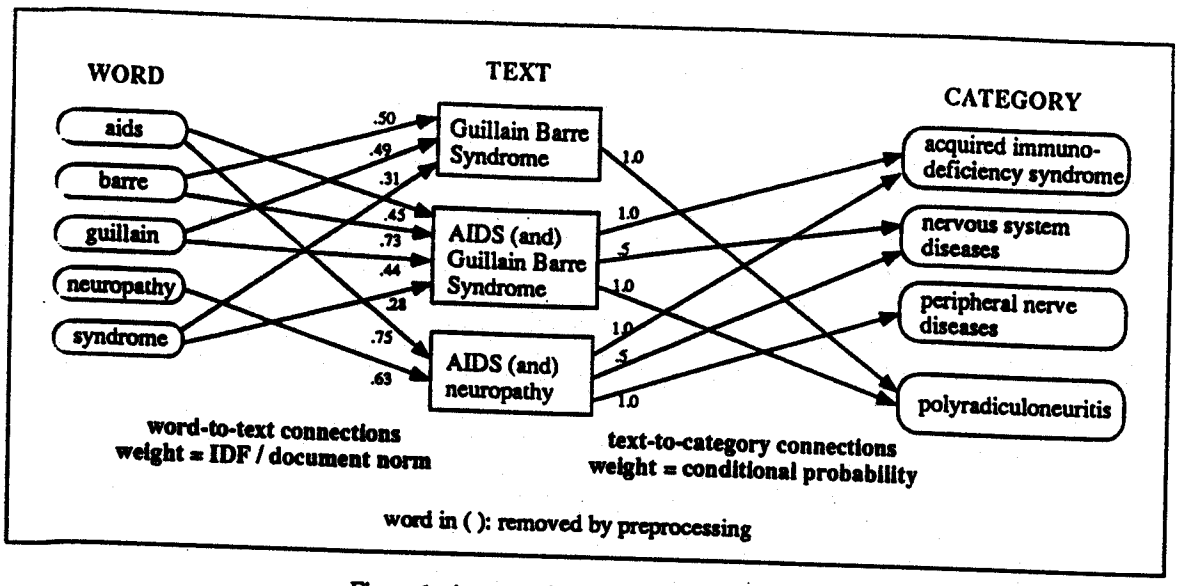

Figure 1. An example of the Expert Network

available training texts, so we must find a way to predict the categories of a new text from the partially matched training texts. For convenience, we will call the text requiring categorization "the request", and a training text "a document" if no specification is given. A request or a document is treated as a set of weighted words in our model.

The Expert Network provides empirical linkages between free words and categories via training documents. Figure 1 shows a simple example of ExpNet. The network consists of three levels of nodes. The input nodes are the unique words in the training documents. The nodes in the intermediate level are the training documents. The output nodes are the categories of the training documents. A word node and a document node are connected if the word occurs in the document. Similarly, a document node and a category node are connected if the category is assigned to the document by humans. These connections facilitate the relevance ranking of categories given an arbitrary request, as described in the following sections.

2.1 Category Ranking via ExpNet

The category ranking of ExpNet consists of two steps. The first step is to measure the similarities between a request and training documents. The second step is to rank the categories based on the similarity scores of the documents, and the conditional probabilities of categories given a document.

We use the conventional similarity measurement, that is, the cosine value of the vectorized request and document, defined as:

$$sim(X, D_j) \stackrel{\text{def}}{=} \frac{\sum_{t_i \in (X \cap D_j)} x_i \times d_{ij}}{\|X\|_2 \times \|D_j\|_2} \tag{1}$$

where

X is a request; D_j is a document; t_i is a word shared by X and D_j;

x_i is the weight of word t_i in request X; d_{ij} is the weight of word t_i in document D_j;

$\|X\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots}$ is the norm of X;

$\|D_j\|_2 = \sqrt{d_{1j}^2 + d_{2j}^2 + d_{3j}^2 + \ldots}$ is the norm of D_j.

For word weighting, we adopted the commonly used schemes as options, including binary weights, term frequency (TF), Inverse Document Frequency (IDF), and the combination TF×IDF [14]. The first layer of ExpNet is constructed for the similarity computation. The weight of the link from word t_i ($i = 1, 2, \ldots, m$) to document D_j ($j = 1, 2 \ldots, n$) is set to the normalized weight $\frac{d_{ij}}{\|D_j\|_2}$. When a request is sent to ExpNet, the word weights (also normalized) in the request are passed through the

input nodes, multiplied by the weights of word-to-document links and summed at the document nodes. Such a computation results in a similarity score of each document as specified in Formula (1).

The second step is the novel part of the method. The idea is to estimate the likelihood suggested by each of the training documents and then integrate the local estimates. We define $P_r(c_k|D_j)$ as the conditional probability of category c_k being related to document D_j by human judgement. Given a training sample of categorized documents, the conditional probability can be estimated as

$$P_r(c_k|D_j) \approx \frac{\text{number of times category } c_k \text{ is assigned to document } D_j}{\text{number of times document } D_j \text{ occurs in the training sample}} \quad (2)$$

where $D_1, \ldots, D_n$ are unique training documents, and $C_1, \ldots, C_l$ are unique categories. Note that a document may have more than one occurrence in the training sample. Diagnoses in patient records, for example, often repeat. Abstracts of articles, as another example, are unlikely to repeat in a bibliographic database; however, some may become identical if an aggressive "stoplist" is applied in a preprocessing to remove unimportant or redundant words. Consequently, the number of times a category assigned to a document may also be more than one.

Given a request X, the relevance score of each category is estimated as the weighted sum of the conditional probabilities,

$$rel(c_k|X) \approx \sum_{j=1}^{n} sim(X, D_j) \times P_r(c_k|D_j) \quad (3)$$

Note that $sim(X, D_j)$ is a cosine value, not a probability. Consequently, $rel(c_k|X)$ is not the probability of category c_k being related to request X. Nevertheless, the relevance scores have the functionality of ranking documents similar to what the probabilities would do. The second layer of ExpNet is constructed for computing the relevance scores. The weight of the link from document D_j to category c_k is set to the conditional probability $P_r(c_k|D_j)$. Before the computation, the default score of each category is set to zero. After the similarity values of documents are computed in the first layer of ExpNet, they are propagated to the second layer, multiplied by the weights of the document-to-category links, and summed at the category nodes. This results in a relevance score of each category as specified in Formula (3).

2.2 Optimization of Category Ranking

While using a weighted sum of the conditional probabilities in category ranking is a reasonable idea, further questions can be raised regarding optimal integration. For example, should we count the contributions from all the documents in the relevance ranking? Or should we just count a few top-ranking documents and ignore the remaining ones? Would a threshold setting for document cutoff optimize the category ranking?

To answer these questions, we tested the effect of document cutoff on the category ranking. That is, we sorted the training documents according to their similarity values, and took only n' ($n' \leq n$) top-ranking documents into account in the relevance estimation. Formula (3) is therefore modified as below,

$$rel(c_k|X) \approx \sum_{D_j \in \{n' \text{ top-ranking documents}\}} sim(X, D_j) \times P_r(c_k|D_j) \quad (4)$$

We used MEDLINE documents for the tests, including 586 training documents and 1758 testing documents (refer to Section 3 for details). There is no overlap between the training set and the testing set. For each of the cutoff thresholds n' = 1, 5, 10, 20, 30, ..., 586 (no cutoff at all), we computed a 10-point average precision. That is, we computed the precision values at recalls of 10%, 20%, ..., 100% and averaged them for a global measure. Figure 2(a) shows the testing results. The interesting points are:

(1) the poorest result (12.6%) occurred when $n' = 1$, i.e. only the top-ranking document was selected;

(2) the best result (34.1%) occurred when $n' = 30$, i.e. the 30 top-ranking documents were selected;

(3) for $n' > 30$, the performance slowly decreased and converged to the level of no cutoff ($n' = 586$).

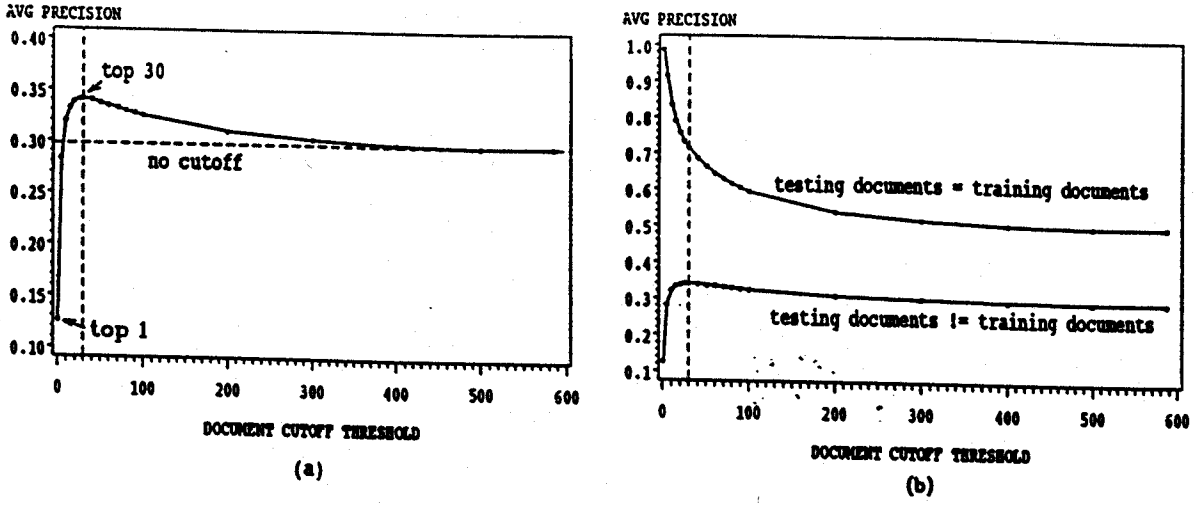

Figure 2. The effect of document cutoff in text categorization.

Our analyses based on the observations are:

(1) The poor result of selecting only the top-ranking document is not surprising. Given that a request and the top-ranking document are partially matched, it is reasonable to assume that they would have some categories in common, but it is misleading to assume that they have exactly the same categories in all cases.

(2) The category ranking based on a few top-ranking documents is relatively reliable, because the categories of these documents, especially the shared categories among them, are informative about the contents of the request.

(3) After a certain point, counting more documents with lower similarity values only contributes noise.

Note that the condition of the above test is that the training documents and the testing documents are different. This is typically true in bibliographic databases where documents are unlikely to be identical to each other. However, in some other applications such as diagnoses or procedure reports in patient records, the texts are relatively short and often repeatedly used. The commonly used texts are likely to occur in a training set. Does the above analysis still hold? To answer this question, we tested a case where every request was included in the training set, that is, we used the MEDLINE training documents as the testing set. Figure 2(b) compares the results of this test and the previous test. The effect of document cutoff in this test is obviously different:

(1) The best result (98.7%) occurred when $n' = 1$, i.e. only the top-ranking document was selected;

(2) for $n' > 1$, the performance monotonically decreased and converged to the level of no cutoff.

This test indicates that when a training document is found to be exactly the same as the request, it is often not helpful to further refer to other training documents which are partially matched. However, the testing condition is rather extreme, where every request is covered by the training set. In reality, a mixture of requests is more likely, that is, some requests are covered by the training set and some are not. For such a situation, a multiple threshold setting would be reasonable:

(1) set the threshold to 1, if the maximal similarity score of the training documents is 1;

(2) set the threshold to constant n' ($n' \leq n$) otherwise.

We are not going to include further details of the optimization. The point is, the optimal threshold(s) can be empirically determined, based on the statistical features of applications.

3 Evaluation

The MEDLINE documents mentioned in Section 2.2 are used for the evaluation. The data set was originally designed for an evaluation of MEDLINE retrieval [15] and has been used for evaluations of

other retrieval and categorization systems [5] [8] [12] [13]. The categories of the documents are assigned by MEDLINE indexers. There are about 168 words and 17 categories per document on average. We arbitrarily split the documents (2344) into a training set (586 documents) and a testing set (1758 documents). There are no duplicates in the entire collection, and consequently, no overlap between the training documents and testing documents. The training set contains 7813 unique words and 1832 unique categories. The testing set contains 14339 unique words and 3430 unique categories; about 42% of these words and 46% of the categories are covered by the training set. The total number of unique categories including the training set and the testing set is 4020. So the chance of a random assignment being correct is $\frac{17}{4020}$ or 0.4%.

A preprocessing was applied to the documents for the removal of punctuation, numbers and the non-informative words on a stoplist, and for changing uppercase letters to lowercase; no stemming was applied. For weighting the links of ExpNet, we tested the word weighting options including binary weights, TF (Term Frequency), IDF (Inverse Document Frequency), and the combination TF×IDF. We will refer to the result of the optimal weighting (using IDF) in the comparison of ExpNet with other methods. The document cutoff threshold of ExpNet was set to 30.

We tested the following methods on the MEDLINE testing set for comparison.

(1) **LLSF** is an example-based categorization/retrieval method described in Section 1. It is theoretically different from ExpNet; however, it uses the same training data as ExpNet.

(2) **STR** (STRing matching) is our implementation of word-based matching in which documents and category descriptions are represented using vectors, and words are represented using binary weights. The relevance score of a category with respect to a document is the cosine value of the vectorized document and category description. We use STR to test the effectiveness of basic word-based matching in text categorization.

(3) **SMART**, developed by Salton's group [2], is one of the most representative retrieval systems. We use SMART to test the effectiveness of retrieval techniques in text categorization. The documents and categories are represented using vectors in the same way as in STR, except the binary word weighting is replaced by the statistical word weighting schemes of SMART. We ran the SMART system for category ranking based on the cosine-similarity between documents and category descriptions. The default parameter setting of SMART was used, and word weighting options TF and TF×IDF were tested. We observed a better result of using TF×IDF than using TF, and will refer to the better result in the comparison of SMART with other methods. Relevance feedback of SMART is not applicable to this test because the training documents and the testing documents are different.

(4) **RFW** ("relevance forward") is our modification of relevance feedback, in order to make it possible to use the training data. Recall that relevance feedback adds words in the related documents of each query to the query. RFW takes a symmetric approach of adding words in the training documents to the descriptions of the related categories. We ran the SMART system for relevance ranking of the expanded categories with respect to the testing documents. We observed that using TF weights had a better result than using TF×IDF, and will refer to the better result in the comparison of different methods.

Figure 3 (a) compares the recall-precision curves of ExpNet and the alternative methods. Figure 3 (b) shows the 10-point average precisions. STR had an average precision of 10.0%. SMART had better performance (16.3%) than STR, indicating the advantage of its statistical word weighting over the binary weighting of STR. RFW had a significant improvement over SMART, showing the benefit of employing human experience in text categorization. ExpNet and LLSF had superior performance over RFW, giving evidence of better ways of using human experience. Instead of simply adding document words to category descriptions as RFW did, ExpNet and LLSF used the statistical information in the training sample to optimize the category ranking; this made the difference. By setting SMART as the base of the comparison, the relative improvements are 114.4% for ExpNet, 113.8% for LLSF, 49.0% for RFW, and -32.8% for STR.

It is interesting to see the very close performance of ExpNet and LLSF, as their results differ only by 0.1% in precision on average. Although they are theoretically and mathematically different methods, the testing results show that both made effective use of the statistical information in the training sample. We

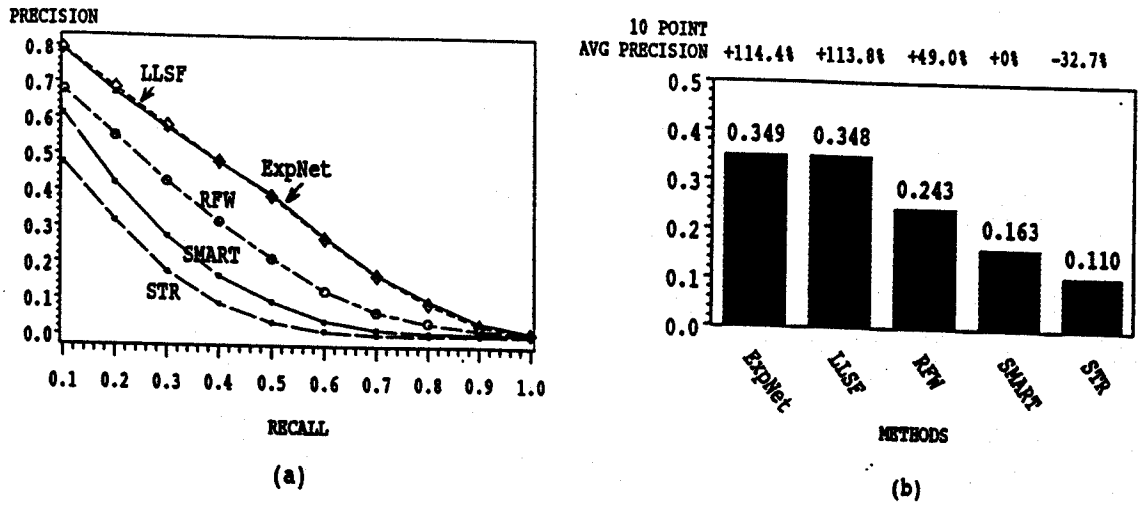

Figure 3. The results of different methods in text categorization of MEDLINE documents.

have tested ExpNet and LLSF in categorization of another text collection which includes 6134 surgical procedures from Mayo patient records [12]. The results were 87.6% for ExpNet and 88.6% for LLSF in the 10-point average precision. The point is, no significant difference between ExpNet and LLSF has been observed in their categorization effectiveness.

4 Computational Complexity

A major attraction of ExpNet is its computational efficiency. ExpNet does not require complex learning or intensive computation. The "learning" of ExpNet is simply the computations of within-document word weights (Section 2.1) and the conditional probabilities of categories (Section 2.2). These computations are based on counting word occurrences and category occurrences in a training set, so the time is proportional to the number of documents, assuming that the average numbers of words and categories per document are constant. In the big-O notation, the time complexity is $O(n)$, where n is the number of training documents. In the construction of ExpNet, we also sort the input nodes (the words) in alphabetical order. The sorting time is $O(m \log m)$ when using a Quick Sort algorithm, where m is the number of unique words in the training set. Together, the time for constructing ExpNet is $O(n) + O(m \log m)$.

Another aspect of the computation is the time for category ranking using ExpNet. For a given query, the category ranking consists of the computations of weighted sums in layers 1 and 2 of the network. The time complexity is $O(N \log N)$, where $N = \max(m, n, l)$, and l is the number of unique categories. This analysis can be explicitly seen in the following algorithm.

initiate the similarity score of documents, $sim(D_j) := 0$ *for* $j = 1, \ldots, n$; — *line 1*

initiate the relevance score of categories, $rel(c_k) := 0$ *for* $k = 1, \ldots, l$; — *line 2*

foreach *word in the request* **do** { — $\# O(\log m) + O(\frac{n}{m})$ — *line 3*

find the corresponding node t_i *for the request word using a binary search;* — $\# O(\log m)$ — *line 4*

foreach *link from node* t_i *to document node* D_j — $\# O(\frac{n}{m})$ — *line 5*

update the similarity score $sim(D_j) = sim(D_j) + \frac{x_i}{\|X\|_2} \times \frac{d_{ij}}{\|D_j\|_2}$; — *line 6*

}; — *line 7*

sort $\{D_1, D_2, \ldots, D_n\}$ *by the similarity scores and select the* n' *top-ranking documents;* — $\# O(n \log n)$ — *line 8*

foreach *link from document* D_j *in the* n' *top-ranking documents* — $\# O(\frac{n}{l})$ — *line 9*

update the relevance score $rel(c_k) = rel(c_k) + sim(D_j) \times Pr(c_k|D_j)$; — *line 10*

sort $\{C_1, C_2, \ldots, C_l\}$ *by the relevance scores;* — $\# O(l \log l)$ — *line 11*

The computation time of each step in the above algorithm is indicated after the # sign. In line 4, for example, the binary search through a sorted list of m input nodes takes $O(\log m)$ time. In lines 5-6, the computation of the inner foreach loop is proportional to the average number of links per word, which is equal to the average number of unique training documents that contain the word. Let constant k_w be the average number of words per document, then the average occurrences per word is approximately $\frac{k_w \times n}{m}$. Consequently, the upper bound of the average links per word is $O(\frac{n}{m})$. In the outer foreach loop of lines 3-6, the time is the product of the number of request words and the time of the inner steps. Since we assume the average number of words per request is constant, the time of the outer loop is $O(\log m) + O(\frac{n}{m})$. The sorting in line 8 takes $O(n \log n)$ time. The computations of the rest of the lines can be estimated in a similar way. To summarize, the time of the category ranking of ExpNet is $O(\log m) + O(n \log n) + O(l \log l) = O(N \log N)$, assuming $O(\log m) \leq O(N \log N)$.

In Section 3, we have shown that ExpNet and LLSF are equally effective in text categorization. From the computational point of view, however, the difference between these two methods is rather significant. Our current implementation of LLSF uses the LINPACK algorithm for singular value decomposition, which requires roughly $O(N^3)$ time [8] (our on-going research shows that the cubic time complexity can be reduced to quadratic). So the training time of LLSF is $O(N^3)$. The categorization time of LLSF is $O(N \log N)$. In our experiments on the training set of 586 MEDLINE documents (containing 7813 unique words and 1832 unique categories), the training of LLSF took 2.25 hours of CPU time on a SUN SPARCstation 10, while the training (network construction) of ExpNet took only 16 seconds. The categorization was 5 seconds per document by LLSF and 0.4 seconds per document by ExpNet. Note that the $O(N \log N)$ time is a necessary cost for any algorithm which requires relative ranking of categories or documents. Hence, ExpNet is optimal.

5 Extension for Text Retrieval

The ExpNet model for text categorization can be easily extended for text retrieval by adding two layers of nodes and links on the network. Figure 4 illustrates such an extension.

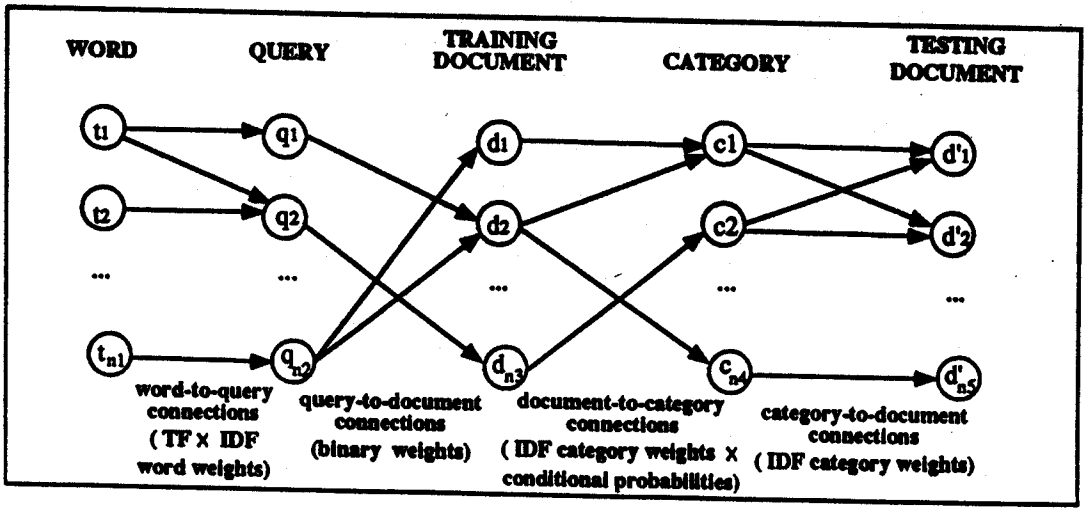

Figure 4. The Expert Network for document retrieval.

The input nodes of the extended ExpNet are the unique words in training queries. The second column of nodes are the training queries. The third and fourth columns are the training documents and their categories, which are the same as in the categorization model. The output nodes are the testing documents which may or may not overlap with the training documents. The first layer of links relates words to training queries, using normalized within-query word weights. The second layer relates training queries to training documents according to human judgements in retrieval. A query-to-document link is established if and only if the document is assigned by humans to the query as being related; all the links have an equal weight of 1. The third layer relates training documents to training categories; the weight of a document-to-category link is the product of the IDF category weight [8] and the conditional

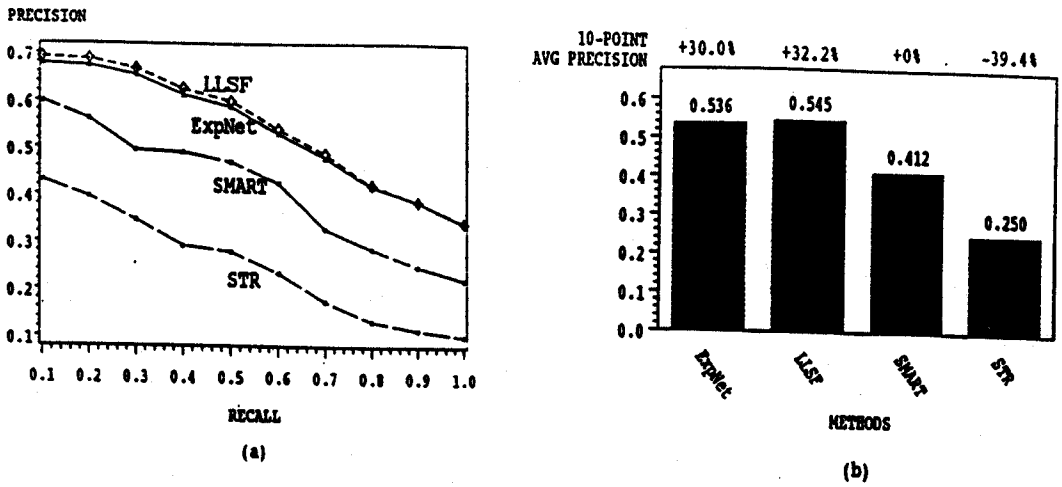

Figure 5. The results of different methods in retrieval of MEDLINE documents.

probability of the category given the document. The fourth layer connects the categories to the testing documents according to human judgements in categorization. A category-to-document link is established if and only if the category is assigned to the document by humans, and is weighted using the IDF value of the category. Such an ExpNet facilitates the relevance ranking of testing documents with respect to an arbitrary query.

We tested the ExpNet retrieval model on the same set of MEDLINE documents which we used in the categorization evaluation. In addition, we used the retrieval queries (75) and the query-document relevance judgements which were available with the documents. We split the data into a training set and a testing set; 88% of the testing queries were covered by the training queries, and none of the testing documents are covered by the training documents. We omit the details of the data set construction due to the space limit of this paper; more description can be found in previous papers [8][12]. Figure 5 shows the retrieval results of ExpNet in comparison with the alternative retrieval methods. ExpNet was only 0.9% less than LLSF in average precision, and significantly outperformed the alternative methods.

6 Discussion

To summarize this paper, a major problem in text categorization and text retrieval is the vocabulary differences between documents and categories, or between user queries and documents. ExpNet, with the capability of learning from human experiences, effectively solves the vocabulary difference problem with a low computation cost. The power of ExpNet comes from the utilization of human judgements about text categorization. The human assignments provide empirical linkages between free vocabularies of documents and a controlled vocabulary of categories. These linkages make ExpNet more effective than word-based matching methods. Furthermore, using categories as semantic units in representing document content is presumably less noisy than using arbitrary words. This would be another reason for the superior performance of ExpNet over word-based matching in document retrieval.

Another characteristic of ExpNet is its use of lexical similarity. In word-based matching, lexical similarity between a query and a document, or a document and a category, is used as the criteria of relevance judgement. In contrast, ExpNet uses lexical similarity for allocating a request in a "neighborhood" of training texts which offer human assigned connections to the target space (categories or documents). In other words, the lexical similarity is used for relating requests to available human knowledge about training texts, and for integrating local relevance estimates into a global measure. Such a use of lexical similarity makes the relevance ranking of ExpNet much simpler than other methods using probabilistic relevance ranking. In Bayesian belief networks, for example, the conditional probabilities for all potential word permutations require an exponential time and space complexity, and independence assumptions therefore have to be made to avoid the intensive computation, i.e. trading context sensitivity for tractability. ExpNet, on the other hand, does not assume word independence in computing the conditional probabilities.

In conclusion, the simplicity of the model, the effective use of human knowledge, and the optimal computational efficiency together make ExpNet a preferable solution for categorization and retrieval of natural language texts. Exploring the capacity of this approach in very large databases and potential limitations remain the focus of our further research.

Acknowledgement

I would like to thank Dr. Kent Bailey for fruitful discussions, and my colleagues in the Natural Language Processing Group at the Section of Medical Information Resources, Mayo Clinic, for help with this project. This work is supported in part by NIH Research Grant LM-05416 to Mayo Clinic, and National Library of Medicine Training Grant LM-07041 in Medical Informatics to the University of Minnesota.

References

1. Hersh WR, Haynes RB. Evaluation of SAPHIRE: an automated approach to indexing and retrieving medical literature. *Proc 15th Ann Symp Comp Applic Med Care* 1991; 15:808–812

2. Salton G. Development in Automatic Text Retrieval. *Science* 1991; 253:974-980

3. Yang Y, Chute CG. A Linear Least Squares Fit mapping method for information retrieval from natural language texts. *Proc 14th International Conference on Computational Linguistics (COLING 92)* 1992; 447–453

4. Harman D. Overview of the first TREC Conference. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval* 1993; 36-47

5. Hersh WR, Hickam DH, Leone TJ. Words, concepts, or both: optimal indexing units for automated information retrieval. *Proc 16th Ann Symp Comp Applic Med Care* 1992; 16:644–648

6. Salton G, Buckley C. Improving retrieval performance by relevance feedback. *J Amer Soc Inf Sci* 1990; 41(4): 288–297

7. Fuhr N, Hartmann S, Lustig G, et al. AIR/X - a rule-based multistage indexing systems for large subject fields. *Proceedings of the RIAO'91* 1991; 606-623

8. Yang Y, Chute CG. An application of Least Squares Fit Mapping to text information retrieval. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval* 1993; 281-290

9. Haines D., Croft B. Relevance Feedback and inference networks. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval* 1993; 2-11

10. Tzeras K, Hartmann S. Automatic indexing based on Bayesian inference networks. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval* 1993; 22-34

11. Wong SKM, Cai YJ. Computation of term associations by a neural network. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval* 1993; 107-115

12. Yang Y, Chute CG. An example-based mapping method for text categorization and retrieval. ACM Transaction on Information Systems 1994; in press

13. Yang Y, Chute CG. Words or Concepts: the Features of Indexing Units and their Optimal Use in Information Retrieval. *Proc 17th Ann Symp Comp Applic Med Care* 1993; 17:685–689

14. Salton G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Pennsylvania, 1989

15. Haynes R, McKibbon K, Walker C, Ryan N, Fitzgerald D, Ramsden M. Online access to MEDLINE in clinical settings. *Ann. Int. Med.* 1990; 112:78–84