A Neural Network Approach to Topic Spotting

Erik Wiener[1], Jan O. Pedersen[2], and Andreas S. Weigend[3]
[1]Xerox Palo Alto Research Center and Dept. of Computer Science, CB 430, University of Colorado at Boulder, Boulder, CO 80309, wiener@cs.colorado.edu
[2]Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304, pedersen@parc.xerox.com
[3]Institute of Cognitive Science and Dept. of Computer Science, Campus Box 430, University of Colorado at Boulder, Boulder, CO 80309, andreas@cs.colorado.edu, http://www.cs.colorado.edu/~andreas/Home.html

Abstract

This paper presents an application of nonlinear neural networks to topic spotting. Neural networks allow us to model higher-order interaction between document terms and to simultaneously predict multiple topics using shared hidden features. In the context of this model, we compare two approaches to dimensionality reduction in representation: one based on term selection and another based on Latent Semantic Indexing (LSI). Two different methods are proposed for improving LSI representations for the topic spotting task. We find that term selection and our modified LSI representations lead to similar topic spotting performance, and that this performance is equal to or better than other published results on the same corpus.

1 Introduction

Topic spotting is the problem of identifying which of a set of predefined topics are present in a natural language document. More formally, given a set of n topics and a document, the task is to output for each topic the probability that the topic is present. These probabilities can subsequently be used for higher level analysis or decision making, in the simplest case through thresholding to yield binary presence decisions.

Topic spotting may be applied to automatically assign subject codes to newswire stories, to filter electronic mail and on-line news, and to prescreen documents in information retrieval and data extraction systems.

While several successful text categorization systems take an expert systems approach, manually constructing a system of inference rules on top of a large body of linguistic and domain knowledge (e.g., [4, 6]), data-driven approaches induce a set of rules from a corpus of labeled training documents. From a practical standpoint, data-driven systems can be put in place relatively quickly (weeks or months) and are able to adapt to changes in the data environment, while hand-crafted systems, though often extremely accurate on their intended task, may take several person-years to develop and are ultimately brittle to changes in the data environment. In this paper we present a data-driven approach to topic spotting that applies nonlinear neural networks to estimate topic probabilities.

Some inductive categorization approaches have been concerned with constructing "human readable" rules that render the operation of the system intuitively understandable, and thus heuristically adjustable. In contrast, the neural network framework we employ is essentially a nonlinear regression model for fitting high-order interactions in some feature space to binary topic assignments. These models are not necessarily interpretable, although they are hopefully high-performance. We examine in this paper to what degree high-order interactions are important in these models.

Since neural networks, like most statistical models, cannot operate with tens of thousands of input variables (corresponding to individual terms), dimensionality reduction is of prime importance. We compare two approaches: term selection, which picks a subset of the original terms to use as features, and Latent Semantic Indexing (LSI), which constructs new features from combinations of a large number of the original terms.

The paper is organized as follows. Section 2 describes related work; Section 3 describes the corpus; Section 4 discusses our representational approach; Section 5 discusses our neural network models; Section 6 presents experimental results; and Section 7 is a concluding discussion.

2 Related Work

Among the many classification methods that have been used for text categorization are Bayesian belief networks [16], decision trees [10], nearest neighbor algorithms [11], Bayesian classifiers [10], and boolean decision rules [1]. Most systems have used words or word-couples for features, but Lewis [9] and Tzeras and Hartmann [16], for example, incorporated phrasal structure into their document representations.

We describe in more detail two recent text categorization studies based on the Reuters-22173 corpus, which is also the subject of the experiments in this paper.

Lewis and Ringuette [10] compared two categorization techniques, one a Bayesian classifier based on a fairly simple conditional probability model and another using decision trees. In both cases, a separate classifier was constructed for each topic. They represented documents as binary word vectors, but found that it was important to drastically reduce the number of words in the representation for successful classification. Using the *information gain* measure[1] to pick words with high individual predictive power, a different set of words was selected for predicting each topic. Results showed the *decision tree* method to give slightly higher performance than the Bayesian classifier, and the authors noted that the decision tree method also resulted in a classifier whose rules were easier to interpret.

Apte, Damerau, and Weiss [1] used a rule induction technique called Swap-1 to produce disjunctive binary decision rules relating the presence of certain combinations of terms to the presence of topics. As in the previous approach, separate classifiers were trained for each topic, each using their own local feature set. The authors found that classification performance improved when *word frequency* values were used instead of binary presence/absence features, and that performance could also be improved by selecting for the local representation the most frequent words for a topic instead of the most informative words.

[1]The information gain measure is based on the *system mutual information* between the presence of a topic and the presence of a word over the documents in the corpus.

3 The Corpus

For our experiments, we used the Reuters-22173 corpus of Reuters newswire stories from 1987, which is now becoming a standard testbed for text categorization research [1, 9, 10].[2] Although there are 21,450 stories in the full collection, we used only those stories which had at least one topic assigned, which left 9,610 for training and 3,662 for testing. While it would have been better to use the full collection (including the stories deemed by the human assigners to be about none of the predefined topics), we believe our partitioning supported meaningful experiments and seems to have been used by at least one other group [1]. The stories have a mean length of 90.6 words with standard deviation 91.6.

There are 92 topics which appear at least once in our training set, and cover such areas as commodities, interest rates, and foreign exchange. While some documents have up to fourteen topics assigned, the mean is only 1.24 topics per document. The frequency of occurrence varies greatly from topic to topic; *earnings*, for example, appears in roughly 30% of the documents, while *platinum* is the topic of only five training documents.

There are 11,161 unique terms in the collection (after performing inflectional stemming, stop word removal, conversion to lower case, and elimination of words which appeared in fewer than three documents).

4 Representation

The starting point for our representations is a term by document matrix containing word frequency information. The entries for each document vector, called a *document profile*, are computed as follows:

$$p_{dk} = \frac{\sqrt{f_{dk}}}{\sqrt{\sum_{vector}(\sqrt{f_{di}})^2}},$$

where f is word frequency. The square root dampens the effect of high counts and the normalization removes the effect of variations in document length. Our term selection and LSI algorithms are applied to these document profile vectors.

4.1 Term Selection

Term selection aims to find the subset of the original terms which seem the most useful for the classification task. It is difficult, if not impossible, to select a set of terms that can adequately discriminate between 92 classes of documents while at the same time being small enough to serve as the feature set for a neural network. Therefore, we divide the problem into 92 independent classification tasks and search for the set of terms for each topic which can best discriminate between documents with that topic and those without.

To select the set of terms for a topic, we score all of the terms according to how well they serve as individual predictors of the topic and pick the top scoring terms. The score, which we call the *relevancy score* because of its relation to what Salton and Buckley call the relevancy weight [15], measures how "unbalanced" the term is across documents with and without the topic:

$$r_k = \log \frac{w_{tk}/d_t + 1/6}{w_{\bar{t}k}/d_{\bar{t}} + 1/6},$$

where w_{tk} is the number of documents with the topic that contain the term, d_t is the total number of documents with the topic, etc.[3] Highly positive and highly negative

[2] The Reuters-22173 collection is available on anonymous ftp at ftp.cs.umass.edu (/pub/doc-/reuters1), thanks to Reuters, Carnegie Group, and David Lewis.

[3] We also considered other measures, such as information gain and the chi-squared test, but they produced similar enough term rankings as the *relevancy score* to not make a major difference in the set of selected features.

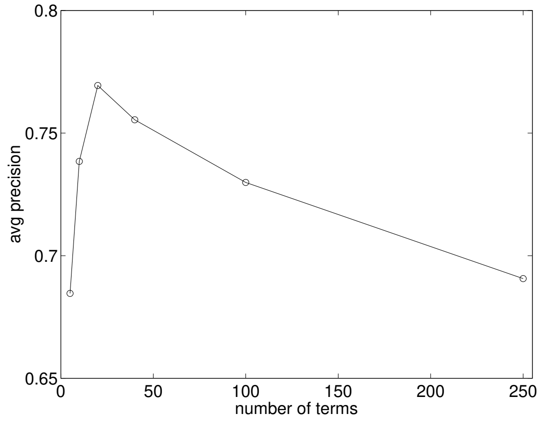


Figure 1: For networks using a selected term representation, performance peaks at around 20 terms. Performance is averaged over six networks predicting topics of varying frequency.

scores indicate useful terms for discrimination.

We found that using about 20 terms yielded, on average, the best classification performance, as measured on an independent test set (see Figure 1). This number falls in the range of feature set sizes used in [10] and is a little smaller than the number used in [1]. Note that we can always decrease the error on the training set by including more terms, but classification performance on out-of-sample data quickly falls off after about 20 terms due to overfitting. Overfitting occurs when the network starts to "memorize" the training patterns; i.e. when it starts fitting the peculiarities of the training data, thus decreasing its performance on out-of-sample data. Overfitting is typically more of a problem the more free parameters (weights) there are in the model compared to training samples. See, for example, [17].

As a sanity check, we computed how well topics could be predicted using only a single key term. We found that by using the top scoring term for each topic as that topic's sole predictor, 20 of the 92 topics could be predicted with over 90% average precision. This was quite surprising, but demonstrates that many of the topics in the Reuters corpus are simply not very hard to predict. These results represent a trivial baseline for experiments with the corpus.

Although term selection has the advantage that relatively little computation is required and that the resulting features have direct interpretability, there are several drawbacks that make it worthwhile exploring other techniques. For example, it is likely that many of the best individual predictors contain redundant information. Also, a term which may appear to be a very poor predictor on its own may turn out to have great discriminative power in combination with other terms. And even if it is possible to find a good set of terms for predicting individual topics, it is unlikely that term selection scales well to models that predict a large number of topics using a single representation.

To address these problems, we applied Latent Semantic Indexing to reduce the dimensionality of our feature set.

4.2 Latent Semantic Indexing

LSI, originally developed to address the problems created by synonymy and polysemy in the standard vector space model of information retrieval [2], seeks to transform the original document vectors to a meaningful lower-dimensional space by analyzing the correlational structure of terms in the document collection. The transformation is computed by applying a singular-value decomposition (SVD) to the the original term by document matrix. The dimensions of the new space have no direct interpretability—they are "rotations" of the original space. However, they are orthogonal. Terms are related in LSI to the extent to which they "travel together" in the corpus. Thus, documents using different terminology to talk about the same concept are positioned near each other in the new space. (See [2] for further details of the LSI technique.)

One property of LSI vectors is that higher dimensions capture increasingly less of the variance of the original data, and hence can be dropped with minimal loss. Original experiments with LSI for IR tasks found that retaining about the first 100 dimensions gave the optimal performance [2], but more recent results suggest that using more dimensions (around 300) is better for some tasks [7]. In our topic spotting experiments, we found that performance continues to improve up to at least 250 dimensions, although the improvement rapidly slows down after about 100 dimensions.

The process of constructing LSI representations from a term by document matrix is straightforward. An SVD is computed on the training set and then new document vectors, such as those from the test set, are transformed by simply projecting them onto the left singular matrix produced in the original decomposition. The cost of computing the SVD is incurred only once for each corpus.

Note, one drawback of LSI is that if we are using a nonlinear classifier, we have to hope that the nonlinear manifold of the original data is preserved in the linear subspace of the LSI representation.

4.3 Improving LSI for Topic Spotting

SVD is an optimal data reduction algorithm if the criterion is least-squared error between the reconstructed vectors and the originals. This says very little, however, about its utility for classification tasks. Indeed, we find that our models which use a basic, global LSI representation perform increasingly worse as topic frequency decreases. This is due to the fact that infrequent topics are usually indicated by infrequent terms, and infrequent terms may be projected out of LSI representations which utilize only a fraction of the original number of dimensions. Thus, even though a particular low frequency term may be important for discriminating between two classes of documents, it may appear to the SVD as mere noise compared to the rest of the data.

We propose two *task-directed* methods for addressing this problem that make use of prior knowledge of the classification task to improve the representations.

4.3.1 Local LSI

The first method biases the LSI representation toward a particular topic or set of topics by modeling only the local portion of the corpus related to those topics.[4] The local portion includes documents which are close to the topics (i.e. they use terminology related to the topics) but don't necessarily have any of the topics assigned. By performing an SVD over only the local set of documents, we expect that the representation will be more sensitive to small, localized effects (such as the correlated use of several infrequent terms) than a global SVD would be. Consequently, we expect that the representation should be more effective for classification of the topics related to that local structure.

This method entails that the global LSI representation be replaced by several specialized LSI representations for the prediction of particular topics, so the benefit of having a single, global representation is sacrificed.

We experiment with two approaches to constructing local LSI representations, embodying two methods of defining local structure. In one approach, we define five broad meta-topics—agriculture,

[4] This method is similar in spirit to a method proposed by Hull [5] for improving classification performance on the routing problem.

energy, foreign exchange, government, and metals—and break the corpus into five clusters, each containing all the documents on a particular meta-topic. These clusters form the local structures for five task-directed LSI representations, which we can call *cluster-directed representations*. To predict any of the topics within a particular meta-topic, the cluster-directed representation for that meta-topic is used.

In the other, more fine-grained approach to local LSI, a separate representation is constructed for classifying each topic. In this *topic-directed* approach, the local region of the corpus relevant for a particular topic is found by using the 100 most predictive terms for the topic as a query and picking the n most similar documents. We pick n to be five times the number of documents containing the topic, but no less than 350 and and no more than 1100. We also add 150 randomly selected documents to the local set before computing the SVD.

Note that there is a trade-off in defining the regions of local structure. The narrower we make the regions, the less flexible the representations become for modeling the classification of multiple topics. We do not expect the topic-directed LSI approach to actually be used in practice because of its high computational overhead (one SVD per topic and 92 200-dimensional representations for each document), but we experiment with it to understand the performance of local LSI representations in the limit of fine-grained structure.

4.3.2 Relevancy Weighting

The other method for creating task-directed LSI representations uses term weights to emphasize the importance of particular terms before applying the SVD. Dumais [3] showed that using an inverse document frequency (IDF) weighting on the term by document matrix before applying the SVD led to 40% average improvement on a set of standard IR test sets. We also found IDF weighting to lead to large improvements in topic spotting performance and use it in all of the experiments reported here. IDF weighting inflates the importance of low frequency terms and diminishes the importance of high frequency terms so that the SVD is forced to distribute its resources more evenly over all terms. It is based on the assumption that, in general, low frequency terms are better discriminators than high frequency terms. In topic spotting, however, we can tune this general assumption.

We use *relevancy weighting* to emphasize terms in proportion to their estimated topic discrimination power, following a suggestion in [15]. The global relevancy weight for each term is computed by summing the absolute value of the relevancy score it has for each topic and cutting the total value off after a certain threshold. This is an admittedly crude way of ensuring that if a term is a good predictor for five topics it is not globally weighted five times as heavily as a term which predicts only a single topic. The relevancy weight is then multiplied by the IDF weight for the term to get the final weighting.[5] The effect is that all low frequency terms are pulled up by the IDF and those which are estimated to be poor predictors are pushed down again, leaving only the relevant low frequency terms with high weights. The SVD can then concentrate on modeling terms that are actually important for the classification task.

Note that this task-directed approach retains a single, global LSI representation for the prediction of all topics.

[5] Actually, we first square the IDF weight before multiplying by the relevancy weight. In using the regular IDF, we found that high frequency terms which were also good predictors received an unnecessary boost which low frequency terms were not quite able to compensate for.

5 Modeling

In this section, we describe our approach to modeling topic assignment, given the representations discussed above. Our basic framework is a regression model relating the input variables (features) to the output variables (binary topic assignments) which can be fit using training data. For linear analysis, we use logistic regression [12], which is appropriate for modeling binary output variables. In this case, the functional form is

$$p = \frac{1}{1+e^{-\eta}},$$

where $\eta = \beta^T x$ is a linear combination of the input features. The logistic function guarantees that $p \in (0,1)$.

Logistic regression can be converted into a binary classification method by thresholding the output probabilities. While this technique has been used quite successfully for a variety of classification problems, we are interested in comparing it to a nonlinear regression method that models interactions between features.

We use nonlinear neural network classifiers to extend logistic regression by modeling higher-order term interaction, and hence non-linear decision boundaries in the input space. Logistic regression is used as the baseline for network performance. We will briefly describe the nature of neural network classifiers, and then discuss how we apply them to the topic spotting task.

5.1 Neural Network Classifiers

There are essentially three major components of a neural network model: *architecture*, *cost function*, and *search algorithm*. The architecture defines the functional form relating the inputs to the outputs (in terms of network topology, unit connectivity, and activation functions). The search in weight space for a set of weights which minimizes the cost function is the training process. In the case of binary targets, cross-entropy is theoretically the most appropriate cost function because it presumes a binomial error model for the outputs [14]. We use the standard backpropagation method of gradient descent as a search technique.

The simplest linear classifier network (also called a logistic network) has an output unit with a logistic activation and no hidden layer, resulting in a functional form equivalent to the logistic regression model. It is useful to separate the model expressed by a network from the search algorithm used to fit the model to the training data. For example, we can learn the weights in a logistic network using either gradient descent in weight space or by using the iterated-reweighted least squares algorithm traditionally used for logistic regression. When applying logistic regression in our experiments, we typically use a logistic network because it gives us greater control over the fitting process, such as allowing us to add priors to the cost function and to use early stopping in training to minimize overfitting [17].

The use of one or more hidden layers of nonlinear activation functions allows the network to model nonlinear relationships between the input and output variables. Each hidden layer learns to re-represent the input data by discovering higher-level features formed out of combinations of the features from the previous level. As in logistic networks, nonlinear networks with logistic activations in the outputs estimate the Bayesian a posteriori probability of the output given the input features, assuming certain conditions are met in training. [13].

5.2 Neural Networks for Topic Spotting

In the context of topic spotting, nonlinear neural networks extract "hidden" features out of nonlinear combinations of terms.

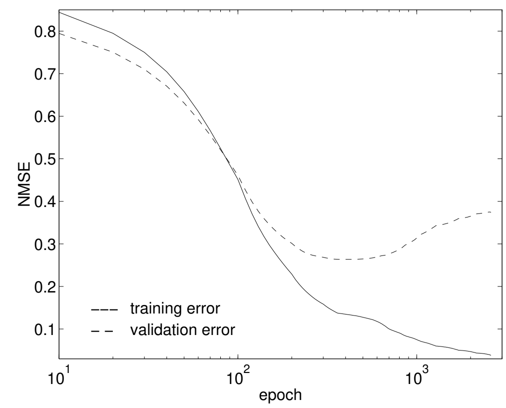


Figure 2: The learning curve for an LSI network predicting the topic GNP (gross national product) shows the typical shape: validation error reaches a minimum and begins to increase long before the network converges.

Network outputs are estimates of the probability of topic presence given the feature vector for a document.

An interesting advantage nonlinear neural networks have over other techniques is that multiple topics can be predicted simultaneously by a single model. In ths case, the hidden units must form a shared representation for predicting all of the topics. Although we believe this has potential benefits, we do not explore multiple, simultaneous prediction in this paper.

We experiment with two different network architectures, one flat and one modular.

5.2.1 Flat Architecture

In the first case, we use the entire training set to train a separate network for each topic. For global LSI representations, each network uses the same representation, while for selected word representations and local LSI, different representations are used for each network. We use simple networks with a single hidden layer of six logistic sigmoid units.

Overfitting the training data is a major problem for all topics, though worse for low frequency topics. While the training set can often be learned with extremely low error, the error on a separate validation set typically reaches its minimum and begins rising after a few hundred epochs (see Figure 2). To help alleviate this problem, we employ a simple regularization scheme based on weight-elimination in which we add a term penalizing network complexity to the cross-entropy cost function.[6] We also use the technique of early stopping based on cross-validation [17].

5.2.2 Modular Architecture

In our second approach, we use a modular architecture to decompose the learning problem into a set of smaller problems. The architecture is shown in Figure 3. The first component is a network trained on the full training set to estimate the probability that each of the five meta-topics—agriculture, energy, foreign exchange, government, metals—is present in the document. A meta-topic is said to be present if any one of its subtopics are present. The second component is a set of five network groups corresponding to the meta-topics. Each group consists of a separate network for each topic in the group, trained only on the region of the corpus corresponding to the meta-topic for that group. For example, the wheat network is trained only on documents which contain one or more topics in the agriculture meta-topic. It can thus focus on the finer distinctions between, for example, wheat and grain, rather than wasting its efforts on easier distinctions, such as between wheat and gold. The meta-topic network uses fifteen hidden units and the

[6] The penalty term, which encourages the elimination of small magnitude weights, is given by $\lambda \sum_{i,j} w_{ij}^2/(1+w_{ij}^2)$.

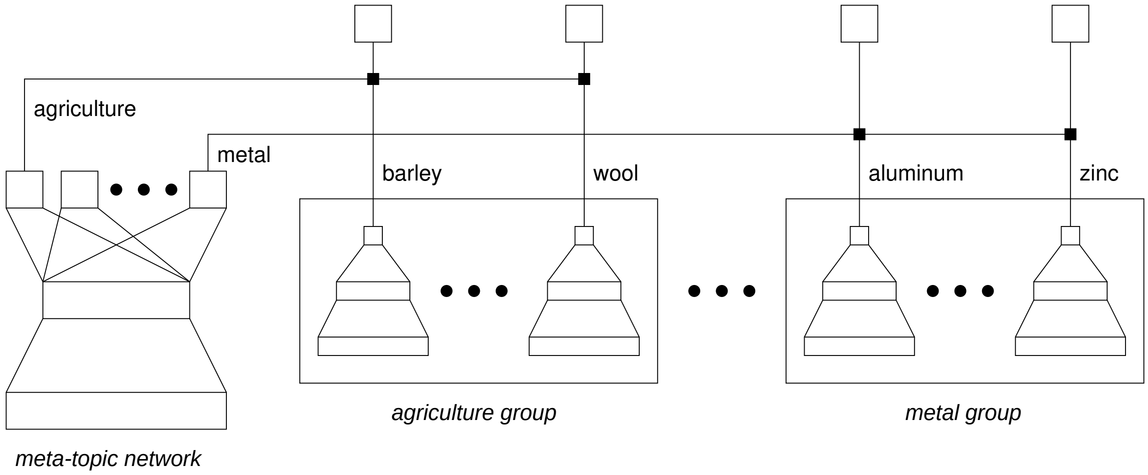


Figure 3: The modular architecture consists of a meta-topic network and five groups of topic networks. Outputs of the meta-topic network are multiplied by outputs of individual topic networks to get final topic predictions.

local topic networks use six hidden units.

To compute topic predictions for a given document using this modular network, we present the document to the meta-topic network and to each of the topic networks (perhaps using a different representation for each network). The outputs of the meta-topic network are then multiplied by the estimates of the topic networks to produce final topic estimates. For example, the output of each network in the agriculture group is multiplied by the agriculture output of the meta-topic network, and so on for each network.

The rationale behind this architecture is that we wish to have a set of networks which are experts in local regions of the corpus, but we need to prescreen documents to know how to weight the opinion of these networks. If, for example, we get a document about cotton prices, we should give no weight to the estimates of the networks in the metal group because they haven't been trained to give judgements on agricultural topics. The meta-topic network performs the job of determining on a high-level what incoming documents are about so the local networks can be deployed appropriately.

6 Experimental Results

In this section we describe our evaluation measures and then present the results of several combinations of the representational and modeling techniques we have discussed.

6.1 Evaluating Performance

The typical evaluation measure in the neural network literature, mean-squared error between actual and predicted values, would tell us very little about how well the network has performed the task of separating the positive from negative examples. A relatively high error, for example, can arise if the examples are perfectly separated but the probability estimates are mid-range rather than near 0 and 1. Instead, we force binary decisions by thresholding the probabilities and compute precision and recall figures based on contingency tables constructed over a range of decision thresholds. Recall is the percentage of documents with the topic that are predicted as having the topic, and precision is the percentage of documents predicted as having the topic that actually have the topic.

We pick decision thresholds in two ways. In proportional assignment [10], we pick a different set of thresholds for each topic by varying an integer parameter, k, over a set of values and at each level setting the decision threshold just below the probability value for the kp'th highest ranked document, where p is the fraction of positive examples of the topic in the training set. In the fixed recall level approach, we predetermine the set of recall levels at which we want to compute precision, and analyze the ranked documents in the test set to determine what the decision thresholds are for each topic that would lead to the desired set of recall levels. Neither is a very satisfying method for picking an operation threshold, but both suffice for characterizing the effectiveness of a classifier over a full range of potential thresholds.

Given a set of contingency tables computed over a range of decision thresholds for each topic, we can summarize performance either by *microaveraging*, which means adding all the contingency tables together across topics at a certain threshold and then computing precision and recall, or we can *macroaverage*, meaning we compute precision and recall individually for each topic and then take an average across topics [8]. For microaveraging, we use proportional assignment for picking decision thresholds, and for macroaveraging we use a fixed set of recall levels. Microaveraging does not weight the topics evenly, so we prefer to use it only for comparisons to previously reported results, in which case the breakeven point is used as a summary value. In all other cases, summary values are obtained for particular topics by averaging precision over the 19 evenly spaced recall levels between 0.05 and 0.95.

6.2 Results

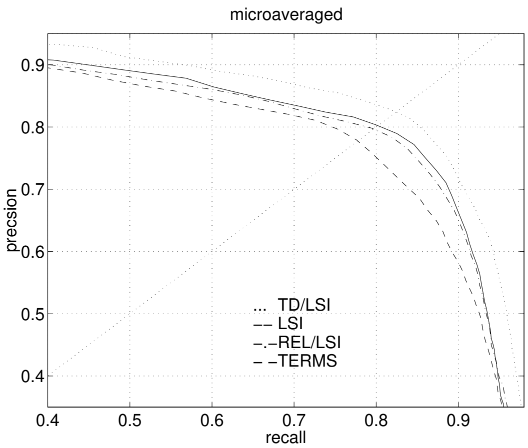
In the experiments reported on here, we used 200 features for LSI representations, both generic and task-directed, and 20 features for selected term representations.

Figure 4a shows the microaveraged precision-recall curves for networks using a generic LSI representation, a topic-directed LSI representation (TD/LSI), a relevancy weighted LSI representation (REL/LSI), and a selected term representation. The breakeven points are 0.801, 0.820, 0.795, and 0.775, respectively. Since we used the same corpus, we can roughly compare our results to the best algorithm from [1] that didn't give special weight to words in the headlines of the stories, which had a reported breakeven point of 0.789. While our results seem quite good, microaveraged performance is somewhat misleading on this task because more frequent topics are weighted heavier in the average. Since our classifiers generally perform the best on high frequency topics, the microaverage is in this case an optimistic measure.

Figure 4b shows macroaveraged performance for the same four classifiers. The selected term network now appears to be much closer in performance to the other three. It is interesting to note that the relative effectiveness of the the representations at low recall levels is reversed at high recall levels.

Figure 5 is a detailed plot showing average precision for six techniques on the 54 most frequent topics (skipping three topics which occurred too infrequently in the test set to give reliable results). Topics are listed in order of frequency, and precision is averaged over 19 recall levels. We present this detailed view to give a feel for the kind of variation that is often hidden behind averages.

Note that although there is considerable variation of performance across topics, the relative ups and downs are mir-



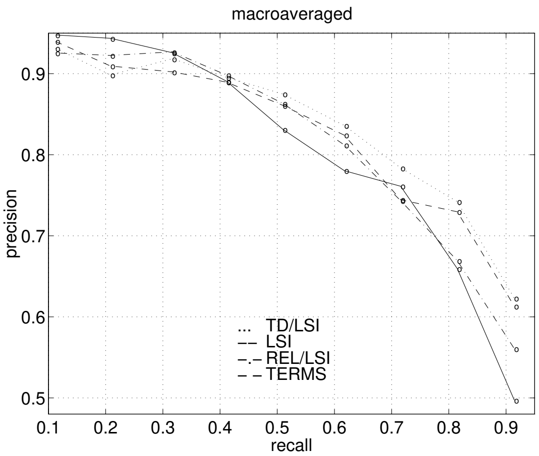


Figure 4: The two plots show microaveraged and macroaveraged precision-recall curves for networks using a topic-directed LSI representation, a relevancy weighted LSI representation, a generic LSI representation, and a selected term representation.

rored to a large degree in both plots. It is also clear that LSI performance degrades quite markedly compared to selected term performance as topic frequency decreases, but that both task-directed representations suffer less from this effect. We note that the topics on which regular LSI performs better than relevancy weighted LSI are in many cases also the topics on which regular LSI does better than term selection. This suggests that relevancy weighting hinders the LSI representation in cases where a large number of terms are important for classification. Finally, we can see that the nonlinear term network offers only a sight improvement over the linear model.

The following tables present a concise summary of the performance of several combinations of techniques, as well as the relative improvement over relevant baseline techniques. The set of 54 most frequent topics is split into thirds (with the highest frequency topics in the first group, etc.) and performance is summarized over each group.

Table 1 displays macroaveraged precision for networks using term selection and three LSI variations.

We experimented with the modular architecture using three representations: the cluster-directed LSI (CD/LSI) representation, a selected term representation, and a hybrid representation in which each document was represented using 200 cluster-directed LSI terms plus 20 selected terms. The meta-topic network component of the modular architecture was trained in all cases using the generic LSI representation. We used only the agriculture, energy, foreign exchange, and metal clusters in this experiments, so some of the 54 most frequent topics not in those clusters were ignored. Therefore, in order to be able to compare the performance of the modular networks with the flat models, we recompute average precision for the flat networks over the topics predicted by the modular network. Table 2 displays the modular network performance and flat networks for comparison.

Table 3 displays for several models their relative improvement over a relevant baseline.

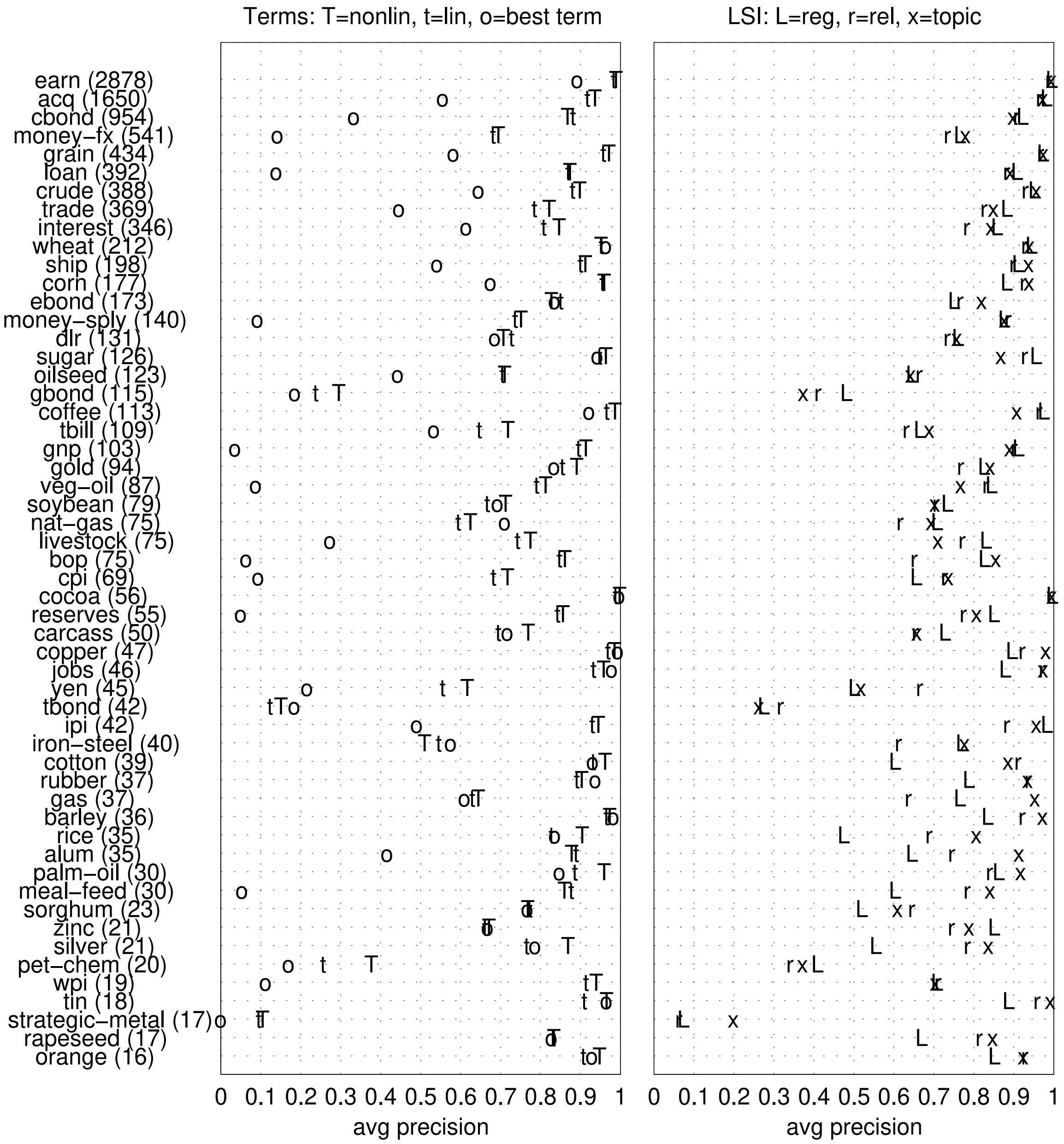


Figure 5: This figure shows average precision on the 54 most frequent topics for three selected term models (nonlinear, linear, linear using single best term) and three LSI models (regular LSI nonlinear, topic-directed LSI linear, relevancy weighted LSI linear). Topics frequencies in the training set are listed beside the names.

Model	All	High	Medium	Low
Terms nonlinear	.804	.834	.796	.784
LSI nonlinear	.771	.862	.786	.664
REL/LSI linear	.785	.850	.773	.735
TD/LSI linear	.811	.856	.780	.798

Table 1: The table displays precision for four flat networks macroaveraged over four topic frequency ranges: topics 1-54 (all), topics 1-18 (high frequency), topics 19-36 (medium frequency), topics 37-54 (low frequency).

Model	All	High	Medium	Low
Modular terms nonlinear	.836	.882	.838	.813
Modular LSI nonlinear	.823	.875	.791	.818
Modular hybrid nonlinear	.835	.887	.807	.829
Terms nonlinear	.808	.860	.823	.774
LSI nonlinear	.752	.864	.812	.661
REL/LSI linear	.782	.860	.797	.736
TD/LSI linear	.811	.862	.785	.803
CD/LSI linear	.796	.857	.800	.764

Table 2: The table displays average precision for three modular networks, as well as several flat networks for comparison. Precision is macroaveraged over the same frequency ranges as in Table 1.

Model vs. baseline	All	High	Medium	Low
LSI nonlinear vs. linear	0.3%	1.0%	-1.6%	1.7%
CD/LSI nonlinear vs. linear	**4.8%**	2.2%	**6.1%**	**5.3%**
Modular CD/LSI nonlinear vs. linear	**2.53%**	1.5%	0.5%	**4.4%**
Terms nonlinear vs. linear	**1.8%**	0.2%	**2.8%**	2.4%
Hybrid nonlinear vs. linear	**1.8%**	**0.8%**	2.3%	**2.1%**
REL/LSI linear vs. LSI linear	2.3%	-0.5%	-3.2%	**12.7%**
TD/LSI linear vs. LSI linear	**5.5%**	0.2%	-2.4%	**22.3%**
CD/LSI nonlinear vs. LSI nonlinear	5.8%	-0.8%	-1.4%	**15.7%**
Modular CD/LSI nonlin vs. Flat CD/LSI nonlin	3.4%	2.1%	-1.1%	7.1%
Modular CD/LSI nonlin vs. Flat LSI nonlin	**9.4%**	1.3%	-2.5%	**23.8%**
LSI nonlin vs. Terms nonlin	**-7.0%**	0.5%	-1.3%	**-14.7%**
TD/LSI linear vs. Terms linear	0.3%	0.3%	**-4.6%**	3.7%
Modular terms nonlin vs. Flat terms nonlin	3.5%	2.7%	1.9%	5.0%

Table 3: The table displays the relative improvement of several networks over relevant baseline models. Improvement is based on the average precision reported in Tables 1 and 2 for the same four frequency ranges. Statistically significant differences (based on a paired t-test at level .025) are shown in bold face.

7 Discussion

While the nonlinear networks seem to perform consistently better than the linear models, the difference is very slight. One possibility is that there are too few positive examples to support nonlinear fits which generalize well. The nonlinear networks are apparently unable to extract features which generalize to out-of-sample data before training is halted with early stopping. We have found that in some cases, performance can be slightly improved by using many more than six hidden units, but the improvement over linear models is still not substantial.

Our experiments show that an LSI representation is able to equal or exceed the performance of selected term representations for high frequency topics, but performs relatively poorly for low frequency topics. However, task-directed LSI representations improve performance in the low frequency domain by 12% in the case of relevancy weighting and up to 22% in the case of local LSI. Using a modular approach in conjunction with local LSI improves performance even further, by up to 23% for low frequency topics. One reason for this huge gain is that in the cluster-directed modular network, individual networks are trained only in the domain in which the local LSI was performed.

Achieving improved performance on low frequency topics from task-directed representations is a trade-off. In the case of topic-directed representations, we need to construct a separate representation for each topic, which is costly. In the case of relevancy weighting, the trade-off is slightly lower performance on many of the medium to high frequency topics. The cluster-directed representation combined with the modular network appears to strike a good balance, since only a handful of separate representations are required and the performance on most topics is relatively high. To minimize the cost, it is worth exploring further variations on the particular relevancy weighting approach used here, as well as combinations of relevancy weighted and local LSI.

It is interesting to note that selected term representations prove to be quite competitive to the more sophisticated LSI technique. This seems to indicate that, at least for the Reuters corpus, most topics are predictable by a small set of terms, often by only a single term.

While it is encouraging that such a simple representation can do so well, sheer performance is not the only measure of a representation's utility. In some circumstances, the benefits of having a single global representation may outweigh the costs of slightly decreased performance on certain tasks. A prime example of such a circumstance is the front end to our modular network, which must draw on a large variety of information in order to determine the high level abstract class of a document. A major challenge seems to be finding ways in which global representations, such as LSI, can be biased toward particular tasks without losing too much of their flexibility as global representations. We believe some of our task-directed approaches represent a meaningful step in that direction.

Acknowledgment

We thank Wray Buntine, David Hull, Tom Landauer, Eric Saund, Hinrich Schütze and John Tukey for many helpful discussions.

References

[1] C. Apte, F. Damerau, and S. Weiss. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th Annual ACM/SIGIR Conference*, 1994.

[2] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[3] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236, 1991.

[4] P. Hayes and S. Weinstein. CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.

[5] D. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th Annual ACM/SIGIR Conference*, pages 282–291, 1994.

[6] P. Jacobs and L. Rau. SCISOR: Extracting information from on-line news. *Communications of the ACM*, 33(11):88–97, 1990.

[7] T. Landauer. Personal communication. 1994.

[8] D. D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318, 1991.

[9] D. D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Computer Science Dept., Univ. of Massachussetts at Amherst, February 1992. Technical Report 91-93.

[10] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Symposium on Document Analysis and Information Retrieval*, 1994.

[11] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the 15th Annual ACM/SIGIR Conference*, pages 59–65, 1992.

[12] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman & Hall, London, 2nd edition, 1989.

[13] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, (3):461–483, 1991.

[14] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. Backpropagation: the basic theory. In P. Smolensky, M. C. Mozer, and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*, pages 1–39. Erlbaum Associates, Hillsdale, NJ, 1995.

[15] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[16] K. Tzeras and S. Hartmann. Automatic indexing based on bayesian inference networks. In *Proceedings of the 16th Annual ACM/SIGIR Conference*, pages 22–34, 1993.

[17] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.