Text Categorization by Boosting Automatically Extracted Concepts

Lijuan Cai
Department of Computer Science
Brown University, Providence, RI, USA
ljcai@cs.brown.edu

Thomas Hofmann
Department of Computer Science
Brown University, Providence, RI, USA
th@cs.brown.edu

ABSTRACT

Term-based representations of documents have found widespread use in information retrieval. However, one of the main shortcomings of such methods is that they largely disregard lexical semantics and, as a consequence, are not sufficiently robust with respect to variations in word usage. In this paper we investigate the use of concept-based document representations to supplement word- or phrase-based features. The utilized concepts are automatically extracted from documents via probabilistic latent semantic analysis. We propose to use AdaBoost to optimally combine weak hypotheses based on both types of features. Experimental results on standard benchmarks confirm the validity of our approach, showing that AdaBoost achieves consistent improvements by including additional semantic features in the learned ensemble.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing Methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and retrieval—*Information Filtering*; I.2.6 [**Artificial Intelligence**]: Learning—*Induction*

General Terms

Algorithms, Experimentation, Theory

Keywords

boosting, classification, concept extraction, document categorization, lexical semantics, machine learning

1. INTRODUCTION

The number of digital documents available in corporate intranets, digital libraries, commercial data bases or the Web as a whole, is vastly growing in size. The sheer volume makes it often prohibitively expensive, if not impossible, to rely on librarians or other domain experts to efficiently annotate and categorize content. Consequently, automated document categorization is an important application area for machine learning and many classification methods have been applied to text categorization. Most recently excellent results have been obtained with Support Vector Machines (SVMs) [1] and AdaBoost [2].

While a broad range of methods has been utilized for text categorization, virtually all of the above approaches use the same underlying document representation: term frequencies. This is typically called the "bag-of-words" representation in the context of naive Bayes classification, while it is also referred to as the term frequency or vector space representation of documents [3]. In the latter case, different dimensions are typically weighted differentially using schemes like `tfidf` and its relatives.

In this paper, we investigate a different direction, namely the use of concept-based document representations. This aims at achieving robustness with respect to linguistic variations such as vocabulary and word choice. We are skeptical that existing general-purpose thesauri can be used to the extent required to handle linguistic variations. Instead, we propose to automatically extract *domain-specific* concepts using an unsupervised learning stage and then to use these learned concepts as additional features for supervised learning. Notice that the set of documents used to extract the concepts need not be labeled and can be much larger than the labeled set of training documents. Hence, our approach also offers an attractive venue for using unlabeled data to improve supervised learning.

We employ a three stage approach. First, an unsupervised learning technique known as probabilistic Latent Semantic Analysis (pLSA) [4] – a probabilistic generalization of LSA [5] – is utilized to automatically extract concepts and to represent documents in a semantic concept space. Second, weak classifiers or hypotheses are defined based on single terms as well as based on the extracted concepts. Third, term-based and semantic weak hypotheses are combined using AdaBoost, resulting in an ensemble of weak classifiers.

2. PROBABILISTIC LSA

Latent Semantic Analysis (LSA) is a well-known dimension reduction technique for co-occurrence and count data [5, 6], which uses a singular value decomposition (SVD) to map documents (and by the virtue of duality also terms) from their standard vector space representation to a lower dimensional latent space. The rationale behind this is that term co-occurrence statistics can at least partially capture seman-

tic relationships among terms and topical or thematic relationships between documents. Hence this lower-dimensional document representation may be preferable over the naive high-dimensional representation since, for example, dimensions corresponding to synonyms will ideally be conflated to a single dimension in the semantic space.

Probabilistic Latent Semantic Analysis (pLSA) [4, 7] is an approach that has been inspired by LSA, but instead builds upon a statistical foundation, namely a mixture model with a multinomial sampling model. pLSA has shown promise in *ad hoc* information retrieval, where it can be used as a semantic smoothing technique. However, our main interest here is less in accurately modeling term occurrences in documents, and more in the potential of pLSA for automatically identifying factors that may correspond to relevant concepts or topics.

The formal setting is as follows: Given a collection of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ and a fixed vocabulary $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$, the data are summarized in a document-term matrix, $\mathbf{N} = (n_{ij})_{i,j}$, where $n_{ij} = n(d_i, w_j)$ is the number of times term w_j occurs in document d_i. We denote concepts by $\mathcal{Z} = \{z_1, z_2, \ldots, z_K\}$. The number of concepts, K, is fixed beforehand, but the concepts themselves are derived in a data-driven fashion. In pLSA, it is assumed that document-term pairs are generated independently and that term and document identity are conditionally independent given the concept. Under this assumption, the joint probabilities of document-term pairs are given by [4]

$$P(d_i, w_j) \quad = \quad P(d_i) \sum_{z_k \in \mathcal{Z}} P(w_j|z_k) P(z_k|d_i)\,. \qquad (1)$$

We interpret the distribution $P(w_j|z_k)$ for a fixed z_k as a representation for concept z_k. Notice that one may identify the terms belonging to a concept by ordering terms according to $P(w_j|z_k)$. It has been shown that extracted concepts are often quite interpretable and can even be used for visualization purposes [8]. However, more important in the context of semantic document representations are the probabilistic "concept memberships" of documents which are encoded in the parameters $P(z_k|d_i)$. They will be used to derive weak learners in Section 4. The concepts or topics z_k in the pLSA model can be thought of as a bottleneck layer in a model that aims at predicting words in documents (cf. Figure 1).

Since Eq. (1) is a special case of a latent class model, one can employ standard techniques for maximum likelihood estimation such as the Expectation Maximization (EM) algorithm for model fitting. We refer to [7] for a formal derivation and simply restate the main result here. Starting from randomized initial conditions, one alternates E-step and M-step updates. In the E-step one computes the posterior probabilities for the latent class variables for all observed pairs (d_i, w_j)

$$P(z_k|d_i, w_j) = \frac{P(w_j|z_k)P(z_k|d_i)}{\sum_l P(w_j|z_l)P(z_l|d_i)}\,. \qquad (2)$$

In the M-step one updates the parameters according to

$$\begin{aligned} P(w_j|z_k) \quad &\propto \quad \sum_i n_{ij} P(z_k|d_i, w_j) \qquad (3) \\ P(z_k|d_i) \quad &\propto \quad \sum_j n_{ij} P(z_k|d_i, w_j)\,. \end{aligned}$$

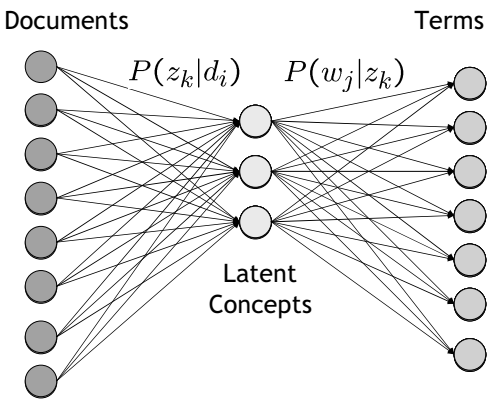


Figure 1: Diagrammatic representation of pLSA as a "bottleneck" model.

In order to determine the optimal stopping point one may perform early stopping based on held-out data.

3. ADABOOST

Boosting is one of the most powerful machine learning approaches to emerge in the past decade. It combines many weak, moderately accurate classifiers into a highly accurate ensemble. Our work is based on AdaBoost, the most popular boosting algorithm [9]. AdaBoost has been successfully applied to a variety of classification problems and has experimentally proven to be highly competitive in the context of text categorization [2].

In this paper, we focus on text categorization with overlapping binary categories. We assume that for every category a set of training examples $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$ is available. Here x_i is the representation of a document d_i, $i = 1, \ldots, M$ and $y_i \in \{-1, 1\}$ encodes whether or not the i-th document belongs to the category.

We have worked with two slightly different versions of AdaBoost called AdaBoost.MH and AdaBoost.MR [10]. The major difference between them is that AdaBoost.MH minimizes an upper bound on the Hamming loss while AdaBoost.MR minimizes an upper bound on the ranking loss.

3.1 AdaBoost.MH

The AdaBoost.MH algorithm is described in Algorithm 1. A distribution $D_t(i)$ over all instances is dynamically maintained, i.e. at every time step t, $D_t(i) \geq 0$ and $\sum_{i=1}^{M} D_t(i) = 1$. $D_0(i)$ is initialized to be uniform. We assume the weak hypotheses have range $\{-1, 1\}$. In the t-th iteration, given data $\mathcal{S}$ and the current distribution $D_t(i)$, the optimal weak hypothesis is selected as the one which minimizes the D_t-weighted empirical error,

$$\begin{aligned} h_t \quad &= \quad \arg\min_h \mathrm{err}(h, D_t), \quad \text{where} \qquad (4) \\ \mathrm{err}(h, D_t) \quad &\equiv \quad \sum_i D_t(i) \frac{1 - y_i h(x_i)}{2} \end{aligned}$$

The corresponding optimal parameter update is given by

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \mathrm{err}(h_t, D_t)}{\mathrm{err}(h_t, D_t)}\,. \qquad (5)$$

Then D is updated by putting more weights on "difficult"

Algorithm 1 AdaBoost.MH for binary classification

1: initialize $D_1(i) = 1/M$ for each x_i in training set
2: **for** $t = 1, \ldots, T$ **do**
3: select optimal h_t w.r.t. distribution D_t
4: compute optimal update step $\alpha_t \in \mathbb{R}$
5: compute new distribution

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is the normalization factor
6: **end for**
7: final discriminant function: $f(\cdot) = \sum_{t=1}^{T} \alpha_t h_t(\cdot)$

Algorithm 2 AdaBoost.MR for binary classification

1: initialize

$$D_1(x_+, x_-) = \begin{cases} 1/(n_+ n_-) & \text{if } x_+ \in \mathcal{X}_+ \wedge x_- \in \mathcal{X}_- \\ 0 & \text{else} \end{cases}$$

2: **for** $t = 1, \ldots, T$ **do**
3: select optimal h_t w.r.t. distribution D_t
4: compute optimal update step $\alpha_t \in \mathbb{R}$
5: compute new distribution

$$D_{t+1}(x_+, x_-) = \frac{D_t(x_+, x_-)\exp(\frac{1}{2}\alpha_t(h_t(x_-) - h_t(x_+)))}{Z_t}$$

for all pairs (x_+, x_-), Z_t is the normalization factor
6: **end for**
7: final discriminant function: $f(\cdot) = \sum_{t=1}^{T} \alpha_t h_t(\cdot)$

instances that are not correctly classified by h_t and less weights on instances that are correctly classified. The procedure repeats for a fixed number of T rounds or until a termination condition is met. The final ensemble is given by $f = \sum_{t=1}^{T} \alpha_t h_t$ and the predicted label can be computed according to $H(x) = \text{sign}(f(x))$.

3.2 AdaBoost.MR

The AdaBoost.MH algorithm aims at finding the hypothesis that predicts the actual label. AdaBoost.MR takes a different approach to the classification problem. The goal is to rank items such that the number of pairs of instances (x_+, x_-) with x_+ being a positive example and x_- being a negative example, where $f(x_+) \leq f(x_-)$ is minimized. Hence the objective functions only depend on relative scores $f(x)$ rather than their absolute values. AdaBoost.MR can also be used when only relative ordering information is available [11].

To be more formal denote the positive instances by $\mathcal{X}_+ = \{x_i | y_i = 1\}$ and the negative instances by $\mathcal{X}_- = \{x_i | y_i = -1\}$. Moreover define $n_+ = |\mathcal{X}_+|$ and $n_- = |\mathcal{X}_-|$. Assume $n_+ > 0$ and $n_- > 0$. Let [.] be 1 if the predicate inside is true and 0 otherwise. Then we can define the rank loss for binary classification as

$$\text{rloss}(f) = \sum_{x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-} \frac{1}{n_+ n_-}[f(x_-) \geq f(x_+)]. \qquad (6)$$

We adapt the original AdaBoost.MR algorithm to our rank loss definition yielding the algorithm described in Algorithm 2. In order to implement the algorithm efficiently, we used the improvements proposed in [10].

4. USING SEMANTIC FEATURES IN ADABOOST

4.1 Combining pLSA and AdaBoost

As argued before, the most commonly used document representation for text categorization uses single words or multi-word phrases as features. Unfortunately these are word forms rather than word meanings. In particular synonyms and polysemies make the word-based representation insufficient. Therefore, we propose to augment the document representation with the help of concepts extracted from pLSA. The parameters, $\{P(z_k|d_i)\}$ can be viewed as representations of documents in the semantic space formed by concepts $\{z_k\}$. Similarly, words can be represented in the semantic space by applying Bayes' rule to the parameterization used in Eq (1), $P(z_k|w_j) = P(z_k)P(w_j|z_k)/P(w_j)$. Words related to similar topics tend to be generated by the same concept, whereas a word with multiple meanings tends to receive a high generating probability from a few corresponding concepts. pLSA can hence identify synonyms and polysemies to some extent.

We have selected AdaBoost as the framework to combine semantic features with term-based features in text categorization because of its ability to efficiently combine heterogeneous weak hypotheses. As we will discuss in more detail in the next subsection, the weak hypotheses used are decision stumps that divide the domain based on the values of a particular feature, which means that every weak hypothesis corresponds to a single feature.

4.2 Weak Hypotheses

After the first stage of pLSA learning, a document d_i can be described in terms of semantic features $P(z_k|d_i)$ as well as word features $n(d_i, w_j)$. Notice that the semantic features are probabilities while word features are word counts or absolute frequencies. Based on these features, one can construct simple weak classifiers that can be used as the weak hypotheses out of which an additive ensemble is formed in AdaBoost.

We have used two different types of weak hypotheses in our experiments. Simple indicator functions are used for word features and threshold hypotheses are used for semantic features. In each boosting round, the two types of hypotheses compete and the best one is selected. In the following, let $h_t(x)$ be a weak hypothesis that predicts a binary label, i.e. $h_t(x)$ has the range $\{-1, 1\}$. When clear from the context, the subscript t is dropped.

4.3 Indicator Function Decision Stumps

An indicator function decision stump divides the instance domain based on a particular binary feature e into $\mathcal{X}_0 = \{x | x_e = 0\}$ and $\mathcal{X}_1 = \{x | x_e = 1\}$, where x_e is the value of feature e in instance x, corresponding to the presence of a certain term in a document in our case. We define the weak hypothesis to be of the form

$$h(x) = \begin{cases} -c & \text{if } x \in \mathcal{X}_0 \\ c & \text{if } x \in \mathcal{X}_1 \end{cases},$$

where $c \in \{-1, 1\}$. If $c = 1$ the presence of the feature is believed to be a positive indicator for the document belonging to the class in question, while $c = -1$ can be used to model negative correlations. In the AdaBoost.MH set-

ting, the value of c is determined for every weak classifier by minimizing the normalization constant Z. Define

$$W_b^j = \sum_{i:x_i \in \mathcal{X}_j \wedge y_i = b} D(i) = \Pr_{i \sim D}[x_i \in \mathcal{X}_j \wedge y_i = b]. \quad (7)$$

The optimal value for c is given by

$$c = \text{sign}(r), \text{ where } r = W_{+1}^1 + W_{-1}^0 - (W_{+1}^0 + W_{-1}^1) \quad (8)$$

and the optimal update step length can be computed as

$$\alpha = \frac{1}{2} \ln \frac{1+r}{1-r}, \quad (9)$$

which is equivalent to Eq. (5).

4.4 Threshold Hypotheses

A threshold hypothesis divides the instance domain based on a particular continuous-valued feature e and a learned threshold, β. $\mathcal{X}_0(\beta) = \{x | x_e < \beta\}$ and $\mathcal{X}_1(\beta) = \{x | x_e \geq \beta\}$. The hypothesis has the form

$$h(x) = \begin{cases} -c & \text{if } x \in \mathcal{X}_0(\beta) \\ c & \text{if } x \in \mathcal{X}_1(\beta) \end{cases}$$

where $c \in \{-1, 1\}$.

In order to learn the parameters, one maximizes (without loss of generality we can assume $r > 0$)

$$r = \sum_i D(i) y_i h(x_i) = c \left(\sum_{x_i \in \mathcal{X}_1(\beta)} D(i) y_i - \sum_{x_i \in \mathcal{X}_0(\beta)} D(i) y_i \right). \quad (10)$$

Therefore the optimal parameters can be computed as

$$\beta = \text{argmax}_{\beta'} \left| \sum_{i:x_i \in \mathcal{X}_1(\beta')} D(i) y_i - \sum_{i:x_i \in \mathcal{X}_0(\beta')} D(i) y_i \right|, \quad (11)$$

$$c = \text{sign} \left(\sum_{i:x_i \in \mathcal{X}_1(\beta)} D(i) y_i - \sum_{i:x_i \in \mathcal{X}_0(\beta)} D(i) y_i \right), \quad (12)$$

$$\alpha = \frac{1}{2} \ln \frac{1+r}{1-r}. \quad (13)$$

In Eq. (11), our implementation picks the optimal threshold, β, to be the mid-point between two adjacent feature values of e in the training data.

5. RELATED WORK

The use of automatically extracted features and dimension reduction techniques for text categorization has been investigated before. Most notably is the use of SVD-based truncation to suppress noisy features as suggested in [12, 13]. A similar idea has also been investigated more recently under the title of *semantic kernels* and utilized in conjunction with SVMs for text categorization [14]. Yet another approach of deriving document representations that takes semantic similarities of terms into account has been proposed in [15]. In [16], a method to systematically derive semantic representation from pLSA models using the method of Fisher kernels [17] has been presented. The resulting semantic kernels are combined with a standard vector space representation using a heuristic weighting scheme. Finally, there are related methods that perform *word clustering* in order to extract more reliable features. This has been investigated in the context of naive Bayes classification [18] and more recently in the context of SVMs [19].

The main advantage of the proposed method compared to previous work is that it allows us to retain the term-based representation and augment - rather than to replace - it with a semantic representation. For certain categories, individual terms might be highly discriminative and carelessly disregarding these features may prove detrimental. Moreover, boosting with decision stump classifiers alleviates the problem of how to combine term-based and semantic features, whereas SVM-based approaches require to adjust relative combination weights, a problem that is non-trivial when dealing with a large number of semantic models of different granularity. Lastly, a large portion of the mentioned work pursues the goal of generating more parsimonious document representations which may not necessarily lead to improved classification accuracy (cf. [18]). In our framework, due to the virtues of AdaBoost, we augment the dimensionality of the representation for learning, but we achieve considerable sparseness in the *learned* classifier, which may only depend on a significantly reduced feature set. Hence the feature reduction is more intimately coupled with the discriminative learning process.

6. EVALUATION MEASURES

There are two interpretations of the output of a classifier, dependent on whether one considers the thresholded binary output of a classifier (as in *information filtering*) or the real-valued output of a discriminant function with the associated linear order on documents (as in *information routing*). In the case of classifier ensembles, a real valued score for each document x is obtained from the ensemble output $f(x) = \sum_t \alpha_t h_t(x)$. The magnitude of $f(x)$ reflects the confidence of the classifier in its decision. Without loss of generality one can assume that the weights are normalized such that $\sum_t |\alpha_t| = 1$ in which case $-1 \leq f(x) \leq 1$. The extreme values of -1 and 1 then correspond to perfect agreement among the weak hypotheses. It is straightforward to convert the ensemble output into a classification output, i.e. to predict a binary label by simply taking $H(x) = \text{sign } f(x)$.

Evaluation measures have been developed for both scenarios and which ones to favor depends entirely on the specific requirements of an application. For example, an ordered list will be more helpful if domain experts refine the output of an automatic annotation system. On the other hand, for a fully automated classification task such as email spam filtering a binary classification view might be more useful. Without assuming a particular application we thus investigate metrics derived from both views.

6.1 Metrics for Classification Functions

Commonly used measures are *precision*, *recall*, *F measure*, and *classification error*. Formally, let the training set be $\mathcal{S} = \{(x_1, y_1), \ldots, (x_M, y_M)\}$, and denote the test set by $\mathcal{S}' = \{(x'_1, y'_1), \ldots, (x'_{M'}, y'_{M'})\}$, where $y_i, y'_i \in \mathcal{Y} = \{-1, 1\}$ is a binary class label. The number of relevant documents in $\mathcal{S}$ is n_+ and the number in $\mathcal{S}'$ is n'_+. Learning on $\mathcal{S}$ results in a classifier $H(x) = \text{sign } f(x)$.

1. Precision

$$p(H, \mathcal{S}') = \frac{|\{x'_i \in \mathcal{S}' | H(x'_i) = 1 \wedge y'_i = 1\}|}{|\{x'_i \in \mathcal{S}' | H(x'_i) = 1\}|}. \quad (14)$$

2. Recall

$$r(H, \mathcal{S}') = \frac{|\{x'_i \in \mathcal{S}' | H(x'_i) = 1 \wedge y'_i = 1\}|}{n'_+}. \quad (15)$$

3. $F1$ score

$$F1(H, \mathcal{S}') = \frac{2p(H, \mathcal{S}')r(H, \mathcal{S}')}{p(H, \mathcal{S}') + r(H, \mathcal{S}')}. \quad (16)$$

4. Classification error

$$\mathrm{Err}(H, \mathcal{S}') = \frac{|\{x'_i \in \mathcal{S}' | H(x'_i) \neq y'_i\}|}{|\mathcal{S}'|}. \quad (17)$$

6.2 Metrics for Ranking Functions

In contrast to the above-mentioned measures, only relative differences of scores $f(x)$ count in metrics based on document ranking. These metrics evaluate how good the ranked list defined by f is compared to an ideal ordering that would rank every relevant document higher than every irrelevant one.

1. Maximal $F1$

   Order documents in $\mathcal{S}'$ based on their predicted scores and denote the obtained order by π. Precision (recall) at k is the precision (recall) obtained by classifying the first k documents in the list as relevant.

$$p_k(f, \mathcal{S}') = \frac{|\{j \leq k | y'_{\pi(j)} = 1\}|}{k} \quad (18)$$

$$r_k(f, \mathcal{S}') = \frac{|\{j \leq k | y'_{\pi(j)} = 1\}|}{n'_+}. \quad (19)$$

   Now the maximal $F1$ value is defined as

$$\mathrm{MaxF1}(f, \mathcal{S}') = \max_k F1_k(f, \mathcal{S}'), \quad \text{where} \quad (20)$$

$$F1_k(f, \mathcal{S}') = \frac{2p_k(f, \mathcal{S}')r_k(f, \mathcal{S}')}{p_k(f, \mathcal{S}') + r_k(f, \mathcal{S}')}$$

2. Precision/recall break-even point

   Note when $k = n'_+$, $p_k(f, \mathcal{S}') = r_k(f, \mathcal{S}')$. This value is called the precision/recall break-even point (BEP).

3. Average precision

$$\mathrm{AvgP}(f, \mathcal{S}') = \frac{\sum_{k: y'_{\pi(k)}=1} p_k(f, \mathcal{S}')}{n'_+}. \quad (21)$$

4. Adjusted $F1$, adjusted precision, and adjusted recall

   The computation of maximal $F1$ actually examines the test data to effectively find an optimal threshold. An alternative would be to optimize the threshold on training data. Hence we devise the adjusted $F1$, adjusted precision and adjusted recall:

$$\theta(f, \mathcal{S}) = \mathrm{argmax}_{\theta'} F1(\mathrm{sign}(f(x) + \theta'), \mathcal{S}) \quad (22)$$

$$\mathrm{AdF1}(f, \mathcal{S}, \mathcal{S}') = F1(f', \mathcal{S}'),\ f' = f + \theta(f, \mathcal{S}). \quad (23)$$

   θ is determined as the midpoint between two π-adjacent training instances. Precision and recall corresponding to the adjusted $F1$ are called adjusted precision and adjusted recall, respectively.

5. Adjusted error

   The idea is similar to Adjusted $F1$. A bias θ is computed from training data to minimize classification error on the training data. We call the classification error of $f' = f + \theta$ on test data the adjusted error.

6.3 Averaging

Macro-averaging and micro-averaging are two conventional methods to average performance across categories. Macro-averaged scores are averaged values over the number of categories. Micro-averaged scores are averaged values over the number of all counted documents. Therefore macro-averaging is generally viewed as a per-category average while micro-averaging corresponds to a per-document average.

Let $q(l_i) = a_i/b_i$ be the score of a metric q on the i-th category l_i. b_i is the number of related documents. For example, $b_i = n'_+$ for the precision/recall break-even point. Then the macro-averaged score on L categories is $\sum_{i=1}^{L} q(l_i)/L$, while the micro-averaged score is $(\sum_{i=1}^{L} a_i)/(\sum_{i=1}^{L} b_i)$. The micro-averaged $F1$ measure is computed from the micro-averaged precision and recall. The micro-averaged adjusted $F1$ is computed in a similar way. The micro-averaged maximal $F1$ is undefined in our experiments.

7. EXPERIMENTS

In the experimental evaluation, we focus on a comparison between AdaBoost using only term features and AdaBoost using both semantic features and term features. As described above pLSA has been utilized to extract semantic features. We ran experiments for both AdaBoost.MH and AdaBoost.MR on two data sets: the Reuters-21578 collection and the 1987 part of the OHSUMED collection.

7.1 Reuters: News Stories Collection

The Reuters-21578 dataset consists of Reuters newswire stories from 1987 [20]. After our pre-processing stage that includes tokenization and stop word removal, 37,926 word types remained. We used the modified Apte ("ModApte") split, which divides the collection into 9,603 training documents; 3,299 test documents; and 8,676 unused documents.

In the first stage, all documents in the collection were used for pLSA learning without making use of the class labels. A held-out set with 10% of the data was created randomly. The other 90% were used to learn the pLSA model while the held-out set was used to prevent overfitting, namely using the strategy of early stopping.

As described earlier, indicator function decision stumps were used for word features, and adaptive threshold weak hypotheses were used for semantic features. Binary classification was performed on the top 50 topics with the largest number of positive documents. The performance was then averaged across the 50 topics.

We ran experiments with the number of concepts, K, equal to $100, 200, 300, \ldots, 1000$. Using semantic features generally led to better averaged performance. The best performance however was achieved by merging semantic features from models with different K. The weighting of semantic features of different granularity was again performed by virtue of AdaBoost. No prior weighting of different features is necessary. Table 1 summarizes the results on the Reuters-21578 data set. In all experiments boosting was run for a fixed number of 200 rounds on each topic. The reported relative improvement of b over a is computed as $(b-a)/a$, except

			p	r	F1	Err	MaxF1	BEP	AvgP	AdF1	AdErr
MH	Macro	term	80.25	61.61	69.71	0.68	79.26	74.85	77.23	71.58	0.66
		mixA	84.19	66.47	74.29	0.59	81.41	76.90	81.20	74.90	0.58
		rel impv	4.91	7.87	6.57	14.03	2.71	2.73	5.15	4.64	12.29
	Micro	term	89.83	77.67	83.31	N/A	N/A	85.79	88.49	84.84	N/A
		mixA	90.18	82.20	86.00	N/A	N/A	86.98	90.49	86.40	N/A
		rel impv	0.38	5.84	3.23	N/A	N/A	1.39	2.25	1.84	N/A
MR	Macro	term	N/A	N/A	N/A	4.54	78.72	73.30	77.23	72.00	0.70
		mixA	N/A	N/A	N/A	1.62	81.40	76.48	81.03	74.00	0.61
		rel impv	N/A	N/A	N/A	64.20	3.40	4.33	4.91	2.78	12.81
	Micro	term	N/A	N/A	N/A	N/A	N/A	84.33	88.14	83.32	N/A
		mixA	N/A	N/A	N/A	N/A	N/A	86.07	90.23	85.82	N/A
		rel impv	N/A	N/A	N/A	N/A	N/A	2.06	2.38	3.00	N/A

Table 1: Results for AdaBoost.MH and AdaBoost.MR on Reuters-21578. All numbers are percentages. The evaluation metrics are p=precision, r=recall, Err=classification error, MaxF1=maximal $F1$, BEP=precision/recall break-even point, AvgP=average precision, AdF1=adjusted $F1$, and AdErr=adjusted classification error. "Term" indicates only using term features. "MixA" means using term features and semantic features drawn from $K = 100, 200, \ldots, 1000$ mixed all together. "Macro" stands for macro-averaged performance while "micro" stands for micro-averaged performance.

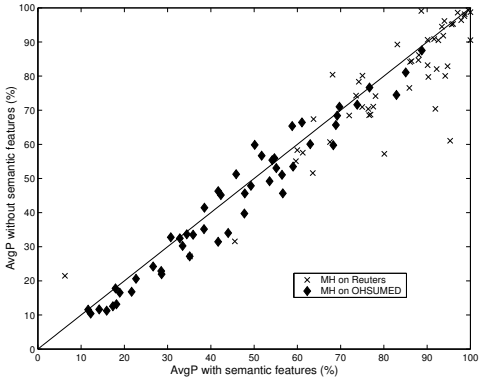


Figure 2: Scatter plot of the average accuracies obtained on individual categories (top 50) for the Reuters and the OHSUMED document collections.

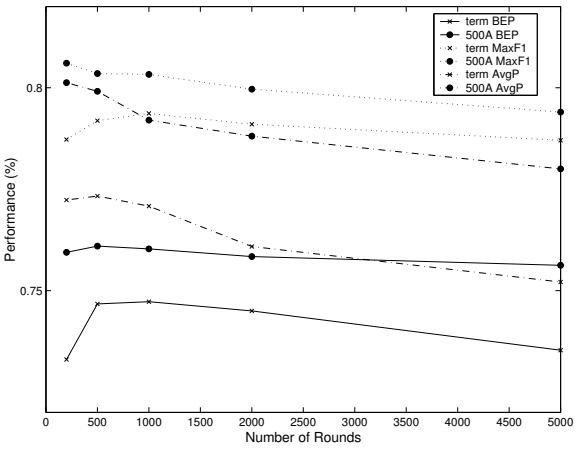
for the classification error and adjusted classification error where the formula is $(a - b)/a$. Note that macro-averaged and micro-averaged classification errors are identical. So are macro-averaged and micro-averaged adjusted error. Besides, precision, recall or $F1$ was not reported for AdaBoost.MR since MR only focuses on the relative ordering of documents.

As can be seen from the results, AdaBoost.MH and AdaBoost.MR perform almost equally well, with AdaBoost.MH having a small, but hardly significant advantage. The use of additional semantic features however yields significant and consistent performance gains for all evaluated metrics and for both AdaBoost.MH and AdaBoost.MR. The relative gains for the macro-averaged metrics are higher, which seems to indicate that semantic features are especially useful for categories with a small number of positive examples. This makes sense intuitively since the term-based features are typically noisier and hence are not expected to work that well in a regime where only few positive example documents are available. Concept-based features on the other hand are likely to have advantages here, because they average over occurrences of many individual terms and hence achieve a higher degree of robustness. Differentiating across different metrics, one sees from Table 1 that the classification error benefits most significantly in relative terms, but, for example, the F1 measure and the average precision are also increased by more than 5% for AdaBoost.MH. We have also included a scatter plot of the average precision performance achieved by AdaBoost.MH with and without semantic features on individual categories. It is depicted in Figure 2.

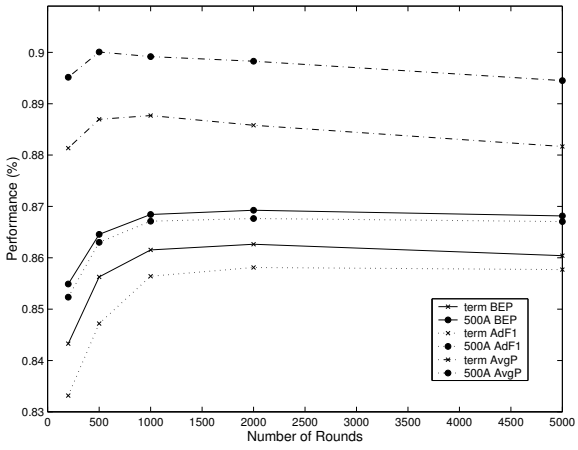
The number of boosting rounds is also a factor that affects the performance. We therefore ran boosting with 200, 500, 1000, 2000 and 5000 rounds. AdaBoost.MR was run on the Reuters-21578 set with only term features against with the data augmented with semantic features extracted from pLSA learning with 500 concepts. The advantage gained by using additional semantic features persisted, though performance measurements varied with the number of iterations performed. Figure 3 summarizes the performance numbers in chart form. Notice that the micro-averaged performance seems to peak around 500-1000 rounds, while the macro-averaged performance is optimal after a smaller number of 200-500 rounds. This seems to indicate that one should dynamically adjust the number of boosting rounds, for instance by performing more rounds on categories with more positive examples. We have however not investigated this issue further.

7.2 Medline: Medical Document Collection

The second corpus is the 1987 portion of the OHSUMED collection [21], which has also been used for the TREC9 filtering track. The collection consists of short MEDLINE documents with titles and abstracts. Each document has been manually indexed by MeSH (Medical Subject Headings) terms. The total number of active MeSH terms in the corpus is $4,904$ and each term is regarded as a category for classification. As above we evaluated the AdaBoost classifier on the top 50 classes and then averaged the results. After removing stop words and words that occurred in less than 10 documents, $19,066$ word types remained. The total number of documents with non-empty title or abstract was $54,708$.



Macro-averaging



Micro-averaging

Figure 3: Comparison of AdaBoost.MR on Reuters-21578 using term features only and using additional semantic features, with varying iteration numbers.

			p	r	F1	Err	MaxF1	BEP	AvgP	AdF1	AdErr
MH	Macro	term	52.48	32.39	40.06	0.55	50.76	45.77	42.95	47.42	0.53
		1000A	57.46	34.25	42.92	0.52	52.68	48.10	45.04	48.68	0.52
		rel impv	9.49	5.74	7.14	5.68	3.77	5.10	4.88	2.66	2.75
	Micro	term	57.47	33.66	42.45	N/A	N/A	47.41	44.82	45.72	N/A
		1000A	62.12	35.71	45.35	N/A	N/A	49.70	47.03	48.27	N/A
		rel impv	8.10	6.09	6.82	N/A	N/A	4.83	4.93	5.59	N/A
MR	Macro	term	N/A	N/A	N/A	2.37	50.53	45.70	42.15	47.04	0.528
		1000A	N/A	N/A	N/A	1.04	52.67	48.01	44.10	49.98	0.520
		rel impv	N/A	N/A	N/A	56.21	4.23	5.06	4.63	6.25	1.59
	Micro	term	N/A	N/A	N/A	N/A	N/A	47.41	43.98	45.55	N/A
		1000A	N/A	N/A	N/A	N/A	N/A	49.52	45.97	49.55	N/A
		rel impv	N/A	N/A	N/A	N/A	N/A	4.45	4.52	8.80	N/A

Table 2: Performance of AdaBoost.MH and AdaBoost.MR on the OHSUMED 1987 data collection. All numbers are percentages. Abbreviations are the same as the ones used in Table 1.

The pLSA model was trained with all the data. For text categorization, 90% of the data were randomly selected as the training set while the other 10% were used for testing.

The OHSUMED 1987 data set is very different from the Reuters-21578 data set with regard to the domain and the document type. However, the idea of deriving semantic features in a data-driven and hence domain-specific manner and then using these concepts to enhance the term-based representation is general. We ran experiments with pLSA models of size $K = 500, 600, \ldots, 1000$ and observed improvements with respect to almost all of the evaluation measures. Table 2 summarizes all relevant experimental results with $K = 1000$. The number of boosting rounds has been fixed to 1000 in all experiments.

By and large the achieved improvements are comparable to the ones obtained on the Reuters data set. While the absolute performance numbers are all lower than for the Reuters data, the relative improvements by using a semantically augmented document representation are again on average in the 5% range. Results on individual categories are included in the scatter plot in Figure 2.

Finally, we have investigated the relative weight given by AdaBoost to semantic features as a function of the number of boosting rounds. The results are summarized in Figure 4 and show a similar qualitative behavior on both data sets: The initial influence of semantic features is small, 45% and 20%, respectively, but increases monotonically until it flattens out, at 73% and 50%, respectively. This means that in the initial rounds of boosting term-based features are chosen more often, while semantic features dominate in later rounds. We attribute this to the fact that in many cases highly discriminative individual terms exist that are selected first, while the more noise-tolerant, semantic features have relative advantages in fine-tuning the classifier.

8. CONCLUSIONS

We have presented a novel approach to text categorization based on semantically augmented document representations. As we have argued this can address some of the shortcomings of pure term-based representations. The overall approach can be decomposed into three stages: In the unsupervised learning stage, we use pLSA to derive domain-specific con-

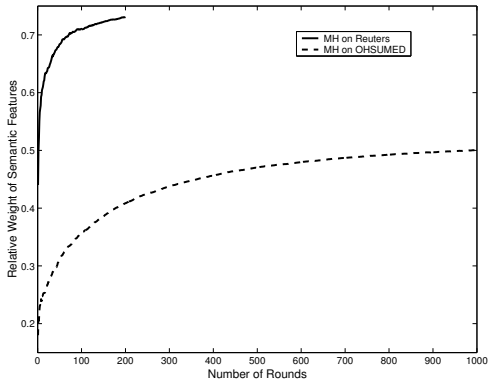


Figure 4: Relative weighting of semantic features as a function of the number of boosting rounds. The weight is averaged over top 50 categories.

cepts and to create semantic document representations over these concepts. In the second step, weak hypotheses are constructed based on both term features and concept features. The concept features can be derived from different pLSA models with different concept granularities and used together. The third stage is the combination stage, which uses AdaBoost to efficiently combine weak hypotheses and to integrate term-based and concept-based information.

The experiments on two standard document collections from different domains support the validity of our approach. The use of concept-based features in addition to term-based features leads to consistent and quite substantial accuracy gains with respect to a variety of standard evaluation metrics. The relative improvement gains achieved are in the 5% range. Future work will investigate the utilization of additional unlabeled data to improve the concept extraction stage as well as the use of linguistic resources such as WordNet to exploit prior knowledge about general concepts.

9. ACKNOWLEDGMENTS

This research was sponsored by an NSF-ITR grant, award number 5710001200.

10. REFERENCES

[1] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings 10th European Conference on Machine Learning*, pages 137–142, 1998.

[2] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[3] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Co., New York, 1983.

[4] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

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

[6] M. W. Berry, S. T. Dumais, and G. W O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):177–196, 1995.

[7] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196, 2001.

[8] T. Hofmann. Probmap - a probabilistic approach for mapping large document collections. *Journal for Intelligent Data Analysis*, 4:149–164, 2000.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[10] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[11] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings 15th International Conference on Machine Learning*, pages 170–178, 1998.

[12] S. T. Dumais. Using LSI for information filtering: TREC-3 experiments. In D. Harman, editor, *The Third Text REtrieval Conference (TREC3) NIST Special Publication*, 1995.

[13] Y. Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 256–263, 1995.

[14] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. In *Proceedings 18th International Conference on Machine Learning*, pages 66–73. Morgan Kaufmann Publishers, 2001.

[15] J. Kandola, N. Cristianini, and J. Shawe-Taylor. Learning semantic similarity. In *Advances in Neural Information Processing Systems (to appear)*, volume 15, 2003.

[16] T. Hofmann. Learning the similarity of documents. In MIT Press, editor, *Advances in Neural Information Processing Systems*, volume 12, 2000.

[17] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems*, volume 11, pages 487–493, 1999.

[18] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 96–103, 1998.

[19] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. On feature distributional clustering for text categorization. In *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval*, pages 146–153, 2001.

[20] David Lewis. Reuters-21578 dataset.

[21] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.