Improved Nearest Neighbor Methods For Text Classification With Language Modeling and Harmonic Functions

Güneş Erkan GERKAN@UMICH.EDU
Department of EECS
University of Michigan, Ann Arbor, MI 48109 USA

Ahmed Hassan HASSANAM@UMICH.EDU
Department of EECS
University of Michigan, Ann Arbor, MI 48109 USA

Qian Diao QIAN.DIAO@INTEL.COM
Intel Corporation, Santa Clara, CA

Dragomir R. Radev RADEV@UMICH.EDU
School of Information & Department of EECS
University of Michigan, Ann Arbor, MI 48109 USA

Abstract

We present new nearest neighbor methods for text classification and an evaluation of these methods against the existing nearest neighbor methods as well as other well-known text classification algorithms. Inspired by the language modeling approach to information retrieval, we show improvements in k-nearest neighbor (kNN) classification by replacing the classical cosine similarity with a KL divergence based similarity measure. We also present an extension of kNN to the semi-supervised case which turns out to be a formulation that is equivalent to semi-supervised learning with harmonic functions. In both supervised and semi-supervised experiments, our algorithms surpass the state-of-the-art methods such as Support Vector Machines (SVM) and transductive SVM on the Reuters Corpus Volume I (RCV1) and the 20 Newsgroups dataset, and produce competitive results on the Reuters-21578 dataset. To our knowledge, this paper presents the most comprehensive evaluation of different machine learning algorithms on the entire RCV1 dataset.

1. Introduction

Text classification has been one of the most popular problems in information retrieval and machine learning. The vast number of its potential applications, the availability of huge (but mostly unlabeled) data especially on the Web and the high dimensionality

of its feature space make it a particularly interesting testbed for machine learning methods in general.

Among many approaches to text classification, this paper primarily focuses on the *nearest neighbor* (NN) based approaches.[1] Even the simplest NN methods such as k-nearest neighbor algorithm (kNN) have been shown to perform surprisingly well in text classification (Joachims, 1998; Yang & Liu, 1999). Despite this success, little effort has been made to improve kNN's performance further. Instead, it has been used as a popular choice of a baseline in many studies to compare it against the proposed method.

There are two critical decisions in a NN based learning algorithm. The first is how the "nearest" neighbors of an instance (document) are determined. The second is how the category of a document is determined by looking at its nearest neighbors. In this paper, we present several alternatives to these two problems. To find the nearest neighbors of a document, we make use of the relevance measures recently popularized in language modeling based document retrieval research (Lafferty & Zhai, 2001). We show that this improves the classification results compared to the classical approach based on the cosine similarity measure.

While combining the labels of the neighbors of a document, we also consider a semi-supervised version of the kNN algorithm where we look at the unlabeled neighbors of a document together with its labeled neighbors. This algorithm turns out to be equivalent to the semi-supervised learning method based on harmonic functions (Zhu, Ghahramani, & Lafferty, 2003) and greatly improves the classification results when there are limited number of training documents. The use of semi-supervised algorithms with limited training data is essential in text classification. This is not only due to the numerous theoretical and empirical justifications of using unlabeled data and the cost of getting training data in the machine learning literature, but also due to the simple fact that some text classification problems inherently have limited training data. For example, consider the problem of classifying one's emails into personal folders (Mock, 2001; Bekkerman, McCallum, & Huang, 2004). The training dataset for this problem is the set of emails that already have been received and classified. There is no way of enlarging this set unless the specific person receives and classifies more emails. A similar problem is classifying the Web pages into one's bookmarks directories.

This paper is organized as follows. Section 2 presents a brief overview of related work. In Section 3, we present the popular Naive Bayes text classification algorithm. Naive Bayes has important connections to language models which will be used repeatedly in the subsequent chapters. Section 4 presents several versions of the k-nearest neighbor (kNN) algorithm, alternative similarity measures and a semi-supervised version of kNN. We explain our experiments and results of comparing several nearest

1. Nearest neighbor-based methods are also known as *instance-based* or *case-based* methods in the machine learning literature.

neighbor methods against each other as well as against Naive Bayes and Support Vector Machines in Section 5.

2. Related Work

A large number of statistical classification and machine learning techniques have been applied to text classification, including regression models, Bayesian classifiers, decision trees, nearest neighbor classifiers, neural networks, and support vector machines (Aas & Eikvil, 1999).

Even a short survey of methods that have been applied to text classification would be out of the scope of this paper, and it wouldn't be an exaggeration to say that nearly all known machine learning methods have been applied where applicable (Sebastiani, 2002).

In this section, we will briefly review some of the work on integrating background knowledge, and using alternate similarity functions in nearest neighbor text classification. We also review some of the work that used the Reuters Corpus Volume I (RCV1)(Stevenson & Whitehead, 2002; Lewis, Yang, Rose, & Li, 2004). RCV1 is becoming the new standard benchmark for text classification. However, it is still not widely used due to its very large size.

Nearest neighbor methods have been widely used to address the task of text classification. Simplest NN methods such as k-nearest neighbor algorithm (kNN) have been shown to perform surprisingly well in text classification (Joachims, 1998; Yang & Liu, 1999).

(Zelikovitz & Hirsh, 2002) propose a method for integrating background knowledge into nearest neighbor text classification. Their approach uses the background knowledge for computing the similarity between training and test instances rather than assessing the similairty directly. This shows that using background knowledge helps to determine which training examples are closest to which test examples. Similar approaches for using background data to improve text classification using Latent Semantic Indexing (LSI) have been proposed in (Zelikovitz & Hirsh, 2001; Zelikovitz & Marquez, 2005). They evaluate their approaches on a set of small datasets including technical papers, news documents, and web pages.

(Galavotti, Sebastiani, & Simi, 2000) propose a variant of the kNN method by exploiting negative evidences. They show that evidences provided by negative examples may help improve kNN performance under certain conditions. They find out that negative examples are not always determinal to the learning process. However, they may be useful under certain conditions, especially when the evidence brought by them is not very similar to test documents. They report the results on the standard Reuters-21578 benchmark.

Reuters Corpus Volume I (RCV1) is a large dataset of more than 800,000 manually categorized newswire stories. RCV1 is significantly larger than the older Reuters-21578 dataset, and it is becoming the new standard benchmark for text classifica-

tion (Stevenson & Whitehead, 2002; Lewis et al., 2004). However, the large size of RCV1 has made it difficult to conduct experiments on the whole datasets, hence most papers conduct experiments on subsets of RCV1. For example, (Fradkin & Kantor, 2005) use 23,149 documents as a training set and 4,060 documents for testing. (Salojärvi, Puolamäki, & Kaski, 2005) use a subset of 4000 documents from four categories and then split them into equal-sized training and test sets. (Zhang, 2005) uses a varying size of training documents ranging from 200 to 2000, and 10,000 documents for testing. (Rousu, Saunders, Szedmak, & Shawe-Taylor, 2004) use 2500 documents for training and 5000 documents for testing. To our knowledge, this paper presents the most comprehensive evaluation of different machine learning algorithms on the entire RCV1 dataset.

More related work is mentioned in the experiments section when comparing our results with previously published results.

3. Naive Bayes

We start with Naive Bayes (NB), one of the simplest yet effective generative models for text classification. NB classification has interesting properties which are important in developing our ideas in the sections that follow. Using Bayes rule, the probability that a document d of length l belongs to the category c is determined by the following equation:

$$p(c|d,l) = \frac{p(d|c,l) \cdot p(c|l)}{\sum_{c'} p(d|c',l) \cdot p(c'|l)} \tag{1}$$

where $p(d|c,l)$ is the probability of observing d given the class c and its length l. It is often assumed that the probabilities in Equation 1 do not depend on document length, so l is dropped from the equation. Therefore, $p(d|c,l)$ becomes $p(d|c)$, and $p(c|l)$ becomes $p(c)$. $p(c)$ is the prior probability of the category c and can be computed from the training data:

$$p(c) = \frac{|c|}{\sum_{c'} |c'|} \tag{2}$$

where $|c|$ is the number of documents that belong to category c in the training data.

In the most popular version of the Naive Bayes model, the words in a document are assumed to be drawn from an underlying multinomial distribution independently of each other. Since all document lengths are assumed to be equally likely, we have

$$p(d|c) \approx \prod_{w \in d} p(w|c)^{tf(w,d)} \tag{3}$$

where $tf(w,d)$ is the number of times word w occurs in d. $p(w|c)$ can be estimated from the relative frequencies of the words in the documents that belong to category c in the training set. However, to avoid zero probability for unseen words in a category, a smoothing method needs to be used. We choose to use Bayesian smoothing with a

Dirichlet prior: (Zhai & Lafferty, 2004; Liu & Croft, 2004):

$$p(w|c) = \frac{tf(w,c) + \mu \cdot p(w|Corpus)}{\sum_{w' \in c} tf(w,c) + \mu} \tag{4}$$

where μ is the Dirichlet smoothing parameter and $p(w|Corpus)$ is the probability of the word w in the entire training set of documents in all categories:

$$p(w|Corpus) = \frac{tf(w,Corpus)}{\sum_{w' \in Corpus} tf(w',Corpus)} \tag{5}$$

The denominator in Equation 1 is the same for all categories, thus it does not affect the category assignment decision for a given document. Therefore, the final category assignment for an unlabeled document is determined by:[2]

$$y_{\text{NB}}(d) = \underset{c}{\text{argmax}}\ p(c) \cdot \prod_{w \in d} p(w|c)^{tf(w,d)} \tag{6}$$

4. Similarity-based Approaches

4.1 k-Nearest Neighbor

A different class of approaches includes nearest neighbor methods where a document is categorized by only looking at the training documents that are most similar to it. Let U be the set of unlabeled documents, and L be the set of labeled documents. Given a document $d \in U$, let $NN_k^L(d)$ be the set of the top k documents in L that are most similar to d with respect to some similarity measure. In the simplest version of the k-nearest neighbor (kNN) algorithm, d is assigned to the category that the majority of the documents in $NN_k^L(d)$ belong to:

$$y_{\text{voting\_kNN}}(d) = \underset{c}{\text{argmax}} \sum_{\substack{d' \in NN_k^L(d) \\ y(d')=c}} 1 \tag{7}$$

We call this method *voting kNN*.

Another version of kNN, which we will call *weighted kNN* takes the magnitude of the similarity values into account. The weighted kNN decision rule can be written as:

$$y_{\text{weighted\_kNN}}(d) = \underset{c}{\text{argmax}} \sum_{\substack{d' \in NN_k^L(d) \\ y(d')=c}} sim(d,d') \tag{8}$$

where $sim(d,d')$ is the similarity between d and d'.

2. We will be using $y(d)$ to denote the category of the document d throughout this paper. When d is unlabeled, $y(d)$ denotes the category assigned by the learning algorithm. When d is labeled, $y(d)$ is its actual label in the training data.

4.2 Similarity Functions

By far the single most popular similarity function used in text classification by kNN is the well-known *cosine* measure defined on the document vectors in the tf or $tf{\cdot}idf$ weighted term space. One is tempted to ask whether using any other similarity function in kNN would improve the classification results.

One inspiration for alternative similarity measures comes from recent research in information retrieval. The language modeling (LM) approach to document retrieval, first introduced by Ponte and Croft (Ponte & Croft, 1998), has proven to be more effective than the traditional cosine retrieval model. In its most basic form, the LM retrieval model assigns a probability to a given query q for each document d in the collection:

$$p_{\mathrm{LM}}(q|d) = \prod_{w \in q} p(w|d)^{tf(w,q)} \tag{9}$$

The above equation looks exactly like Equation 3. There we computed the probability of a document given the term distribution of a set of documents (category) while in Equation 9 the same probability is computed for a query given the term distribution of a single document. In LM terms, $p(q|d)$ is called the *generation probability* of q given the *language model* $p(w|d)$ of d.

The generation probability is a measure of how related (or similar) the query is to a document and has been shown to perform better in document retrieval than the $tf \cdot idf$ cosine based retrieval model (Ponte & Croft, 1998). We can turn it into a document similarity measure by replacing q in Equation 9 with a document so that $p_{\mathrm{LM}}(d|d')$ becomes the similarity of d' to d.

Using generation probabilities in weighted kNN has connections to ensemble methods in machine learning which aim to build a classifier by taking weighted votes from a "committee" of classifiers (Dietterich, 2000). Weighted kNN in combination with the sim_{LM} measure can be seen as a special case of *Bayesian voting*, one of the simplest ensemble methods, where we construct a Naive Bayes classifier from individual documents (nearest neighbors) and take their weighted votes (Equation 8) to make the final classification decision. Previous research has shown that an ensemble classifier is often more accurate than any of the classifiers that it is derived from (Opitz & Maclin, 1999; Dietterich, 2000).

There is one problem with using language model probabilities in nearest neighbor classification. Since all the word probabilities are multiplied to find the generation probability of a given document, small differences among different language models may result in huge differences in terms of the ratio of these different generation probabilities to each other. Indeed, Naive Bayes is known to produce overconfident probabilities for this reason. Rennie (Rennie, 2001) empirically showed that the (normalized) Naive Bayes probability for the top category is close to 1.0 for most of the documents in a classification dataset. In our experiments, we have tried to see if this property holds for pairwise document-document generation probabilities as well. For each document d in the two datasets we use in this paper (see Section 5.1),

we have computed $p_{\text{LM}}(d|d')$ for all other documents d' in the dataset to find the 30 nearest neighbors of d. For 17,879 documents out of a total of 18,828 documents in the 20 Newsgroups dataset, the top nearest neighbor has a similarity value larger than the sum of the similarity values of the remaining 29 neighbors (Table 1). This is not the case for any of the documents if we use cosine as the similarity measure. This essentially means that we do not gain anything by taking weighted votes from the neighbors. In other words, using weighted kNN for these 17,879 documents is effectively the same as assigning each of them to the category that its top nearest neighbor belongs to.

Another similarity measure used in information retrieval is based on the Kullback-Leibler (KL) divergence (Lafferty & Zhai, 2001). KL divergence is a measure of the distance between two distributions. In document retrieval, the KL divergences between the language model of a query and the language models of each document are computed to find the similarity scores. Since KL divergence is a distance measure, the negative KL divergence is used to rank the documents:

$$-\text{KL}(q \parallel d) = \sum_w p(w|q) \log \frac{p(w|d)}{p(w|q)} \tag{10}$$

KL divergence has interesting connections to generation probabilities. Consider the following equation:

$$\begin{aligned} \exp(-\text{KL}(q \parallel d)) &= e^{\sum_w p(w|q) \log \frac{p(w|d)}{p(w|q)}} \\ &= \prod_{w \in q} \left(\frac{p(w|d)}{p(w|q)} \right)^{p(w|q)} \\ &= \prod_{w \in q} p(w|d)^{\frac{tf(w,q)}{|q|}} \cdot \prod_{w \in q} \left(\frac{1}{p(w|q)} \right)^{p(w|q)} \end{aligned} \tag{11}$$

We can ignore the second factor in the product as it does not depend on the document, so it is a constant scaling factor for a given query. The first factor in the multiplication looks very similar to the p_{LM} value in Equation 9. The only difference is the $1/|q|$ factor in the exponent, where $|q|$ is the number of words in q. Thus, instead of multiplying the probabilities of all the words in q in Equation 9, we take the geometric mean of these probabilities in Equation 11. Note that for a given query, Equation 9 and 11 will produce exactly the same ranking of documents (the second term and the $1/|q|$ factor in Equation 11 depend only on the query and do not change the ranking.) Therefore, there is no practical difference between the two equations from the document retrieval perspective. This also means that if we use Equation 11 as a document similarity measure by replacing q with a document (as we have done for p_{LM}), the set of k nearest neighbors of a given document will be exactly the same as the set of k nearest neighbors that we get from p_{LM}. Thus, the voting kNN classification will also produce the same results for both similarity measures.

However, the similarity values computed by exp(−KL) will be "tighter" because of the geometric mean operation involved. This is obvious in Table 1 where we see that although the set of 30 nearest neighbors for a document is the same with respect to both p_{LM} and exp(−KL), similarity values computed using KL divergence are closer to each other. The exp(−KL(·|·)) function has been used as a document-document similarity in recent research and has helped improve the quality of document retrieval (Kurland & Lee, 2005) and document clustering (Erkan, 2006).

Table 1: The number of documents for which the similarity of their top nearest neighbor with respect to different similarity measures is larger than the sum of the similarities of the next 29 nearest neighbors.

Dataset	*cosine*	p_{LM}	exp(−KL)
20 News (18828)	0	17879	8
Reuters (10789)	0	8985	0

4.3 Semi-supervised kNN and Harmonic Functions

Consider a binary classification problem where each document in the training set is labeled as 0 or 1. For this special case, we can write the weighted kNN equation as follows:

$$y(d) = \frac{\sum_{d' \in N_k^L(d)} sim(d, d') y(d')}{\sum_{d' \in N_k^L(d)} sim(d, d')} \tag{12}$$

In other words $y(d)$ is set to the weighted average of its neighbors' labels $y(d') \in \{0, 1\}$. Note that $y(d)$ can take any real value in the $[0, 1]$ interval. If we classify each unlabeled document d that has $y(d) < 0.5$ as negative (class 0), and $y(d) > 0.5$ as positive (class 1), Equation 12 functions exactly the same as the weighted kNN classification of Equation 8 for binary classification.

Equation 12 suggests a generalized semi-supervised version of the same algorithm by incorporating unlabeled instances as neighbors as well:

$$y(d) = \frac{\sum_{d' \in N_k^{L \cup U}(d)} sim(d, d') y(d')}{\sum_{d' \in N_k^{L \cup U}(d)} sim(d, d')} \tag{13}$$

Unlike Equation 12, the unlabeled documents are also considered in Equation 13 when finding the nearest neighbors. We can visualize this as a graph, where each data instance (labeled or unlabeled) is a node that is connected to its k nearest neighbor nodes. The value of $y(\cdot)$ is set to 0 or 1 for labeled nodes depending on their class. For each unlabeled node x, $y(x)$ is equal to the average of the $y(\cdot)$ values of its neighbors. Since the labels of the unlabeled documents appear on both sides of Equation 13, it is not obvious whether a solution for $y(d)$ exists. Such a function that

is set to fixed values ($\{0, 1\}$ in this case) for labeled nodes, and satisfies the averaging property on all unlabeled nodes is called a *harmonic* function. Fortunately, harmonic functions on a graph are known to have a unique solution (Doyle & Snell, 1984). Harmonic functions were first introduced as a semi-supervised learning method by Zhu et. al. (Zhu et al., 2003). They also showed interesting connections between harmonic functions and several physical and mathematical models. For example, consider a random walk that starts on an unlabeled node d on the similarity graph induced by the similarity function as described by Equation 13. Then $y(d)$ is equal to the probability that this random walk will hit a node labeled as 1 before it hits a node labeled as 0. In electrical engineering, $y(d)$ is equivalent to the electric potential of a node.

We can write Equation 13 for all documents in matrix form. Let n be the total number of documents and $\mathbf{W}$ be the (normalized) weight matrix of the similarity graph such that

$$\mathbf{W}_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{sim(d_i, d_j)}{\sum_{d' \in N_k^{L \cup U}(d_i)} sim(d_i, d')} & \text{otherwise} \end{cases} \tag{14}$$

Then Equation 13 can be generalized to the matrix form as:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{y} \tag{15}$$

with the constraint that the solution vector $\mathbf{y}$ takes the value 0 or 1 for labeled documents depending on their label. Keeping this constraint in mind, we can rewrite this equation as follows:

$$(\mathbf{I} - \mathbf{W_{uu}})\mathbf{y_u} = \mathbf{W_{ul}}\mathbf{y_l} \tag{16}$$

where $\mathbf{y_u}$ and $\mathbf{y_l}$ denote only the values of the $\mathbf{y}$ vector that correspond to the unlabeled and the labeled documents, respectively. Similarly, $\mathbf{W_{uu}}$ is a submatrix of $\mathbf{W}$ that only includes the edges from unlabeled to unlabeled documents, and $\mathbf{W_{ul}}$ is a submatrix of $\mathbf{W}$ that only includes the edges from unlabeled to labeled documents. The only unknown in Equation 16 is the $\mathbf{y_u}$ vector and it can be solved efficiently with an iterative linear system solver (Zhu et al., 2003).

5. Experiments

5.1 Datasets and Preprocessing

To test the similarity functions and the learning methods introduced in previous chapters, we use three standard text classification datasets. The first one is a version of the 20 Newsgroups dataset (20news-18828)[3] that contains 18,828 documents after removing the duplicates in the original dataset which has 19,997 documents. All the headers except for "From" and "Subject" are removed in the newsgroup articles.

3. `http://people.csail.mit.edu/jrennie/20Newsgroups/`

Each document belongs to exactly one newsgroup category. There are a total of 20 categories that are almost evenly distributed over the documents.

The second dataset is the Reuters-21578 dataset that consists of documents collected from the Reuters newswire in 1987. Following previous research (Joachims, 1998; Yang & Liu, 1999; Nigam, Mccallum, Thrun, & Mitchell, 2000; Rennie, Shih, Teevan, & Karger, 2003), we use the "ModApte" split of this dataset and then consider only 90 categories which have at least one training and one test document in the split. This leaves us with a total of 10789 documents (7770 training and 3019 test documents in the ModApte split). Unlike the 20 Newsgroups dataset, some of the documents in Reuters-21578 belong to more than one category. The categories are also not evenly distributed: some categories have as few as two documents while others may have few thousands of documents.

The third dataset is the Reuters Corpus Volume I (RCV1). RCV1 consists of over 800,000 manually categorized newswire stories made available by Reuters,Ltd. for research purposes(Stevenson & Whitehead, 2002; Lewis et al., 2004). RCV1, is significantly larger than the older Reuters-21578 dataset. It includes all English news stories produced by Reuters in the period between August 1996, and August 1997. Like Reuters-21578, RCV1 documents may belong to more than one category, and the categories are not evenly distributed.

We removed the stopwords and stemmed all the words in all datasets using the Porter stemmer. While computing the generation probabilities and KL divergence, the Dirichlet smoothing parameter μ is set to 1000. All this preprocessing has been performed using the Lemur toolkit (Ogilvie & Callan, 2001).

5.2 Implementation Details

Note that the generation probabilities and the KL divergence based similarity measure are not symmetric. As mentioned in Section 4.2, for a given document d, we use $p_{\mathrm{LM}}(d|\cdot)$ and $\exp(-\mathrm{KL}(d|\cdot))$ (not $p_{\mathrm{LM}}(\cdot|d)$ and $\exp(-\mathrm{KL}(\cdot|d))$) to find the nearest neighbors of d with respect to the generation probability and the KL divergence based similarity measures, respectively. However, for harmonic functions, we consider both directions and we make two documents neighbors of each other if *either* of them is in the k-nearest neighbor set of the other. The similarity value is set to the maximum of the two values for different directions. This makes the underlying graph of the harmonic function undirected.[4] In all of our nearest neighbor based experiments, k (the number of neighbors) was chosen as 30. We tried different values for this parameter and found out that it does not have a significant impact on the performance and we kept it at a reasonable number to be able to make a fair comparison across different algorithms and datasets. To find the solution of the harmonic function in

4. Although previous research focused on undirected graphs, it can be shown that the solution of the harmonic function still exists on a directed graph. However, undirected graphs have consistently performed better in our experiments.

Equation 15, we use the Iterative Template Library [5], which is an efficient C++ library to solve linear equations.

After computing $y(\cdot)$ in Equation 13 for each document, using 0.5 as the threshold value to classify a document as positive or negative does not always yield the best result. This is especially the case for datasets such as Reuters-21578 where the class sizes vary a lot. Therefore, following Zhu et. al. (Zhu et al., 2003), we use *class mass normalization* to adjust the threshold for a given binary classification problem. Suppose the ratio of the training documents in the positive class to all the training documents is r. Class mass normalization classifies a document d as positive if and only if

$$z(d) = r \cdot \frac{y(d)}{\sum_{d' \in U} y(d')} - (1 - r) \cdot \frac{1 - y(d)}{\sum_{d' \in U} 1 - y(d')} \geq 0 \tag{17}$$

Since we have defined the semi-supervised kNN (harmonic functions) method as a binary classification algorithm, it needs to be extended to the multi-class case. For each class c in the training data, we construct a one-vs-all classifier where the training documents in c are labeled as positive and all other documents are labeled as negative. After running all these classifiers using class mass normalization, we classify each document d to the class whose classifier yields the maximum $z(d)$ value given by Equation 17. This extension to the multi-class case is done only for the 20 Newsgroups dataset. Since some of the documents in the Reuters corpus belong to more than one class, we construct only binary one-vs-all classifiers for each class and then evaluate them separately.

RCV1 is a very large dataset with more than 800,000 documents. We applied 8 different algorithms variants on RCV1: Harmonic Functions with kl similarity, Harmonic Functions with cos similarity, Weighted kNN with kl similarity, Weighted kNN with cos similarity, Voting kNN with kl similarity, Voting kNN with cos similarity, NaiveBayes classifier, and SVM classifier. For each algorithm, we used 13 different training data sizes. For each training data size, we chose a random subset of the documents as the training set, we repeated this for ten times and reported the average of ten random runs. This results in 130 runs for each algorithm and a total of 1040 runs. This large number of runs on such a huge dataset requires a prohibitively long running time. Hence, we used 30 machines from a large parallel cluster to deploy the different runs. To our knowledge, the evaluation this paper presents on the RCV1 dataset, is the most comprehensive evaluation of different text classification algorithms on the entire dataset.

5.3 Results

5.3.1 Evaluation Metrics

The first evaluation metric we use is the classification accuracy. However, achieving high accuracy for each binary classification on the Reuters corpus is trivial since the

5. `http://www.osl.iu.edu/research/itl/`

Table 2: The precision-recall breakeven points of various algorithms on the ModApte split of the 10 largest categories of the Reuters-21578 dataset. Microaverages of these categories and all of the 90 categories are also shown. v-kNN: voting kNN, w-kNN: weighted kNN, Harm.: Harmonic functions, NB: Naive Bayes, SVM: Support Vector Machines TSVM: Transductive Support Vector Machines with normalized $tf{\cdot}idf$ document representation and linear kernel.

Class	v-kNN (cos)	w-kNN (cos)	v-kNN (KL)	w-kNN (KL)	Harm. (cos)	Harm. (KL)	Naive Bayes	SVM	TSVM
acq	88.87	86.09	94.30	93.46	89.57	94.99	93.05	97.50	97.08
corn	71.43	67.86	62.50	62.50	67.86	66.07	46.43	87.50	78.57
crude	83.07	83.60	85.71	85.19	87.30	88.36	84.13	89.42	88.89
earn	93.65	95.31	97.52	97.33	95.03	97.88	94.85	98.80	98.53
grain	83.22	83.22	81.21	80.54	85.23	83.22	73.15	92.62	87.25
interest	71.76	73.28	76.34	75.57	77.10	79.39	61.07	81.68	82.44
money-fx	70.39	69.27	79.33	77.65	77.09	78.77	65.92	79.89	81.01
ship	79.78	78.65	80.90	79.78	78.65	80.90	79.78	88.76	85.39
trade	74.36	71.79	74.36	75.21	77.78	74.36	57.27	76.92	70.09
wheat	67.61	67.61	64.79	64.79	74.65	69.01	63.38	85.92	77.46
microavg. (10)	86.26	86.01	89.81	89.31	88.27	90.71	85.22	93.68	92.47
microavg. (90)	80.07	79.89	82.61	82.18	82.05	83.89	74.63	88.62	85.90

negative class dominates the dataset. Therefore we consider precision and recall for each binary classification defined as follows:

$$\text{Precision} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}} \tag{18}$$

$$\text{Recall} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive documents}} \tag{19}$$

After ranking the documents by their score in the binary classification, we have a trade-off between precision and recall by adjusting the positive/negative cutoff in the ranked list. We report the precision-recall breakeven point (Joachims, 1998) which is defined as the precision and the recall value where the two are equal. Note that the scoring transformation for harmonic functions in Equation 17 does not change the ranking of the documents but only the classification threshold value. Therefore, it will have an effect on the accuracy of the classification but no effect on the precision-recall breakeven point.

To combine the scores of different binary classifications for the Reuters dataset, we use microaveraging (Yang, 1999), that is, take the weighted average of the scores of all binary classifications. In other words, instead of taking the simple average of

all binary classifications, the size of each class is also taken into account. This gives a better overall score since the sizes of the classes in the dataset may vary a lot.

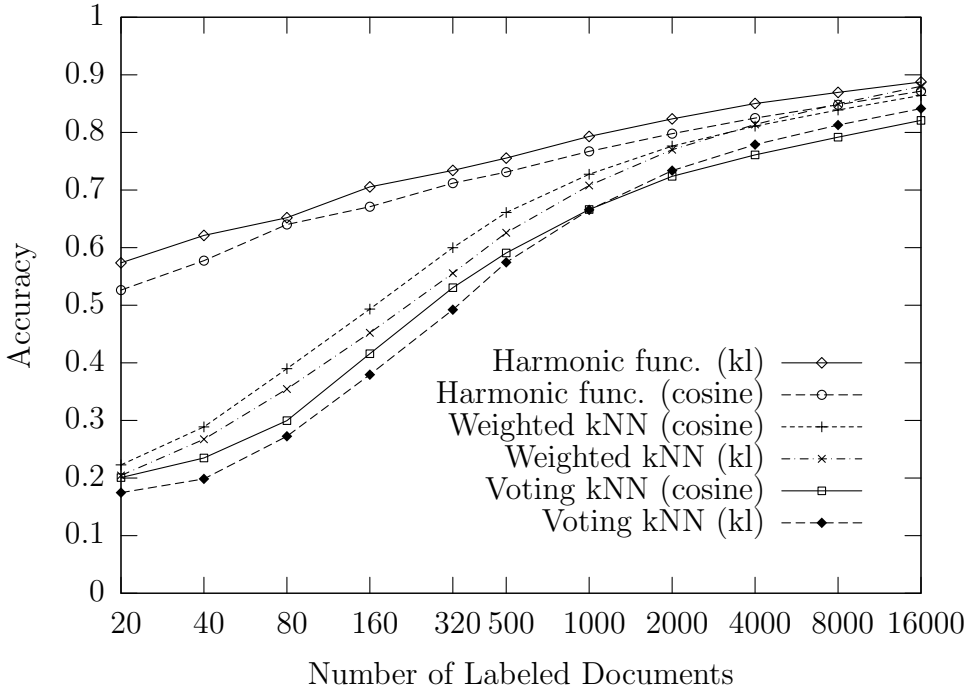

Figure 1: The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the 20 Newsgroups dataset.

5.3.2 Comparison of Nearest Neighbor Methods and Similarity Functions

As we have mentioned above, using accuracy for the Reuters-21578 dataset produces very high scores for all the algorithms because the negative class in each binary classification is too large. For this reason, we report the precision-recall breakeven scores on the 10 largest classes in the Reuters-21578 dataset as well as the microaverages for these 10 binary classifications and for all of the 90 binary classifications in Table 2. We see that for a given similarity measure, weighted kNN performs slightly (but not significantly) worse than voting kNN, and harmonic functions perform better than both weighted and voting kNN overall and on most of the individual classes. KL divergence based similarity performs better than cosine for a given learning method.

To see the effects of semi-supervised learning, we also run several experiments by varying the size of the training data and using the rest of the dataset as the test data. For each training data size, we choose a random subset of the documents as the training set, and then we report the average of 10 such random runs. We make

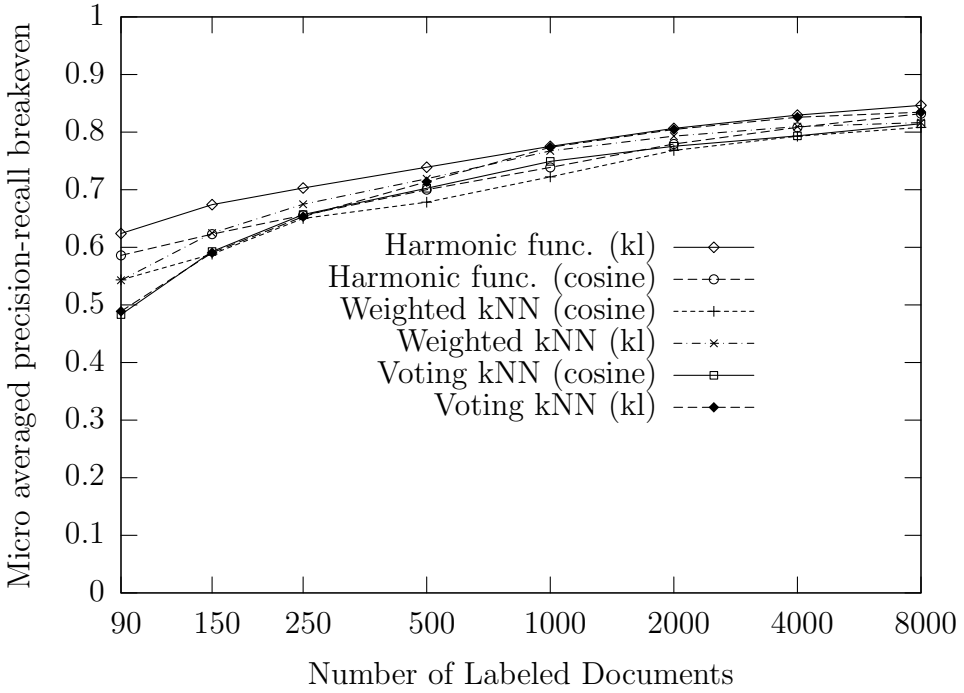

Figure 2: The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the Reuters-21578 dataset.

sure that each class is represented by at least one document in each training set. Therefore, the minimum number of training documents can be 20, 90, and 103 for the 20 Newsgroups, Reuters-21578, and RCV1 datasets, respectively, since there are that many classes in each case. In Figure 1, it is clear that weighted kNN performs better than voting kNN, and harmonic functions perform better than weighted kNN on the 20 Newsgroups dataset. Not surprisingly, the differences in the accuracies of these methods tend to shrink as we introduce more training documents. One interesting observation is that KL divergence based similarity performs somewhat worse than the cosine similarity using the supervised kNN methods especially with limited training data. However, it performs the best in conjunction with harmonic functions consistently at all training data sizes. Figure 2 shows the results of the same experiment on the Reuters-21578 dataset using the precision-recall breakeven point microaverage of 90 classes. Similar observations can be made here except that the absolute differences among the performances are smaller. However, harmonic functions using KL divergence based similarity still perform the best especially where there is little training data. Figure 3 shows the results of the same experiments on the RCV1 dataset. Like the the 20 Newsgroups dataset, harmonic functions perform better than both weighted and voting kNN. We also notice that KL divergence based

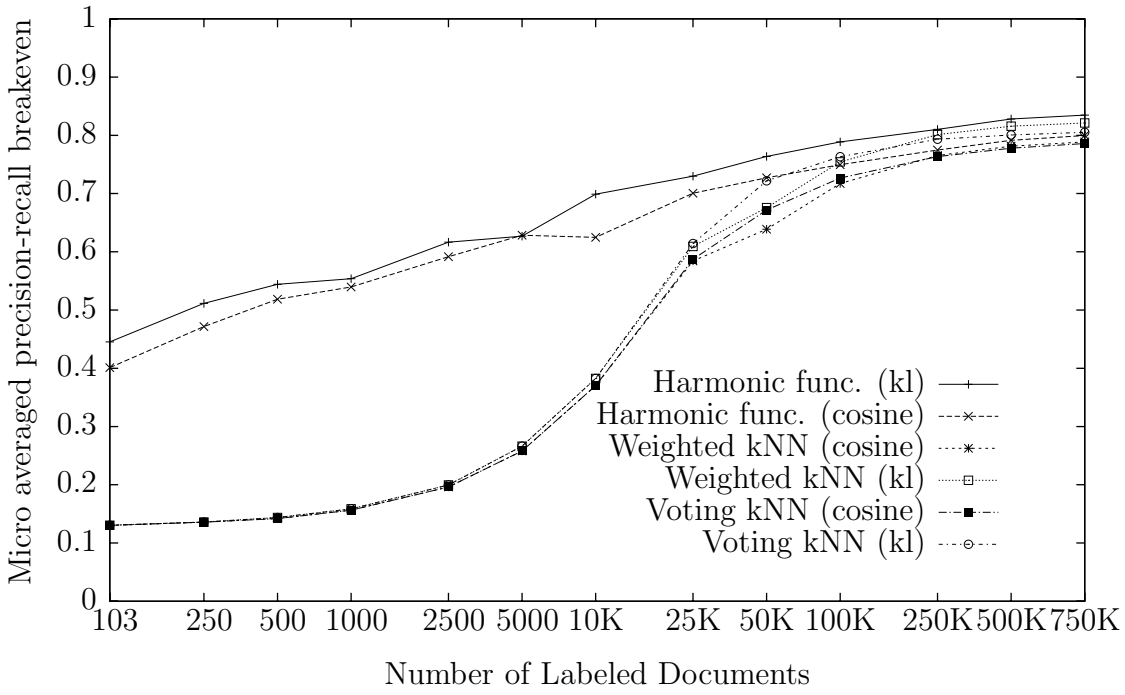

Figure 3: The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the RCV1 dataset.

similarity performs better than the cosine similarity. Again the differences in the performance tend to shrink as we introduce more training documents.

5.3.3 Comparison against Other Methods

To compare the nearest neighbor methods against other standard text classification methods, we chose two of the most popular learning algorithms: Naive Bayes and Support Vector Machines (SVM). We use the multinomial naive Bayes as explained in Section 3. We also perform experiments with the transductive SVM (TSVM) algorithm since it is one of the leading semi-supervised learning algorithms for text classification (Joachims, 1999b). For SVM classification, we choose the linear SVM and use the SVM^{light} package as implemented by Joachims (Joachims, 1999a). We compute the $tf{\cdot}idf$ vector representation of each document normalized to unit length before running the SVM classifier.

Table 2 shows that the nearest neighbor methods perform better than Naive Bayes on the Reuters-21578 ModApte split. SVM performs the best.[6]

6. The microaveraged value we have found for SVM is slightly higher than Joachims's (Joachims, 1998) (88.62 versus 86.5) probably due to the difference in preprocessing the dataset and also the fact that we use $tf{\cdot}idf$ vectors but he uses only tf vectors.

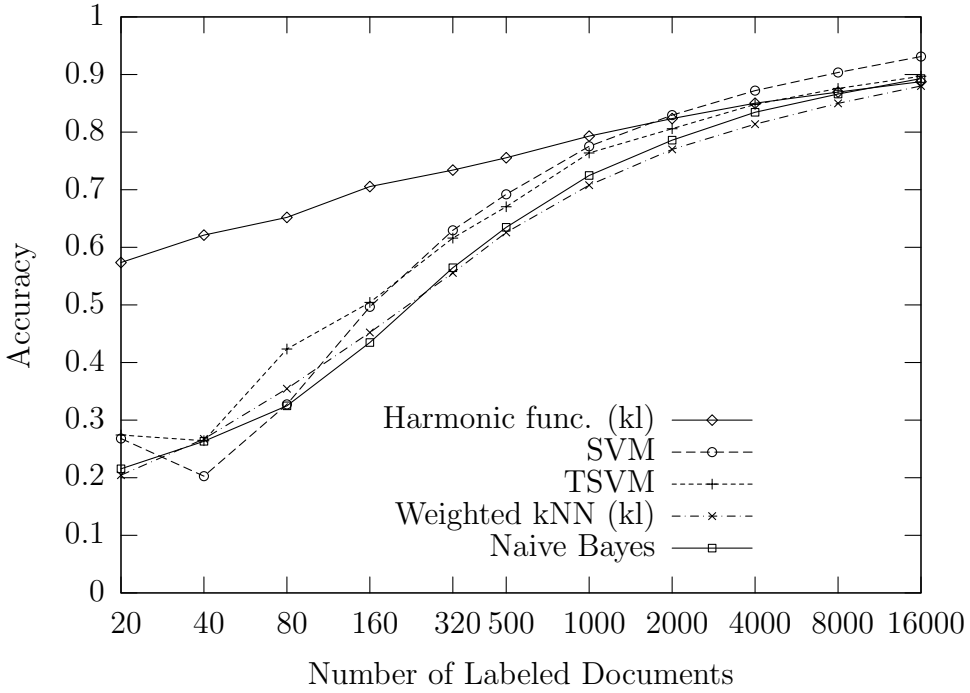

Figure 4: The performances of different algorithms on the 20 Newsgroups dataset with varying training data sizes.

Figures 4 and 5 show the comparison of these algorithms at varying training data sizes on the 20 Newsgroups and Reuters-21578 datasets, respectively. For simplicity, we only include harmonic functions and weighted kNN with KL divergence based similarity from Figures 1 and 2. Semi-supervised harmonic functions perform much better on the 20 Newsgroups dataset than all other algorithms at smaller training data sizes. This is a remarkable performance especially considering the performance of the other state-of-the-art semi-supervised algorithm, TSVM. SVM and TSVM seem to catch up only when a lot more training documents are introduced. Weighted kNN's performance is competitive and parallel to Naive Bayes's.

On the Reuters-21578 dataset, however, SVM is the best algorithm at all training sizes. An interesting observation is that TSVM performs worse than SVM even with few training documents. This is contrary to the previous experiments on a subset of this dataset (Joachims, 1999b). Our experiments on all 90 Reuters categories show that TSVM performs worse than SVM. Aside from this oddity of the Reuters dataset, Figure 5 shows that harmonic functions again perform better than TSVM when there is limited training data. TSVM surpasses the harmonic functions when the size of the training data is increased. As in the 20 Newsgroups dataset, all the supervised and semi-supervised nearest neighbor methods are still better than Naive Bayes this time by a wide margin. Figure 6 compares the performance of harmonic functions and

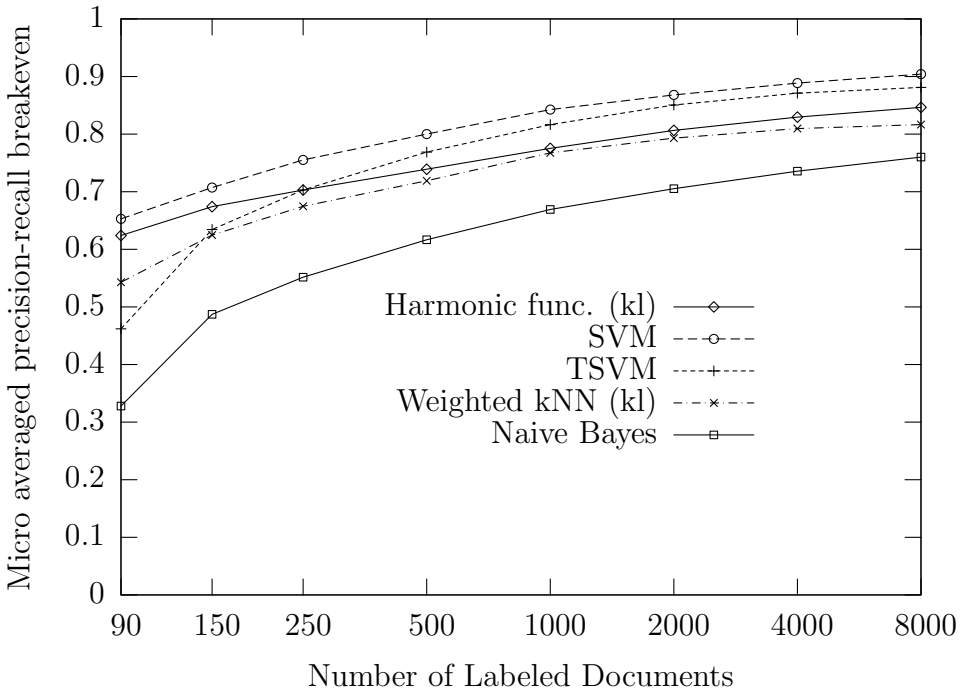

Figure 5: The performances of different algorithms on the Reuters-21578 dataset with varying training data sizes.

weighted kNN with KL divergence, Naive Bayes, and SVM on the RCV1 dataset. It shows that harmonic functions still surpasses all other algorithms for limited sizes of training data, and kNN performs the worst. As the size of the training data increases, all methods converge to a similar performance except Naive Bayes which has the worst performance. We also tried to run TSVM on the RCV1 dataset but we were not able to see the algorithm terminate after running it for several days. It seems that TSVM, or at least the SVM^{light} package we use, does not scale well for this large dataset.

5.3.4 Statistical Significance Tests

To further test the statistical significance of the results, we apply the paired t-test to the results obtained on the RCV1 dataset similar to (Yang & Liu, 1999).

Given two paired sets X and Y of n measured values, the paired t-test determines whether they differ from each other in a significant way. The paired t-test involves carrying out a one sample t-test on the differences between corresponding elements of the paired sets X and Y. Let's define the following notation:

- $x_i \in [0, 1]$ is the score of system X on category i.
- $y_i \in [0, 1]$ is the score of system Y on category i.
- $d_i = x_i - y_i$ is the difference between x_i, and y_i.

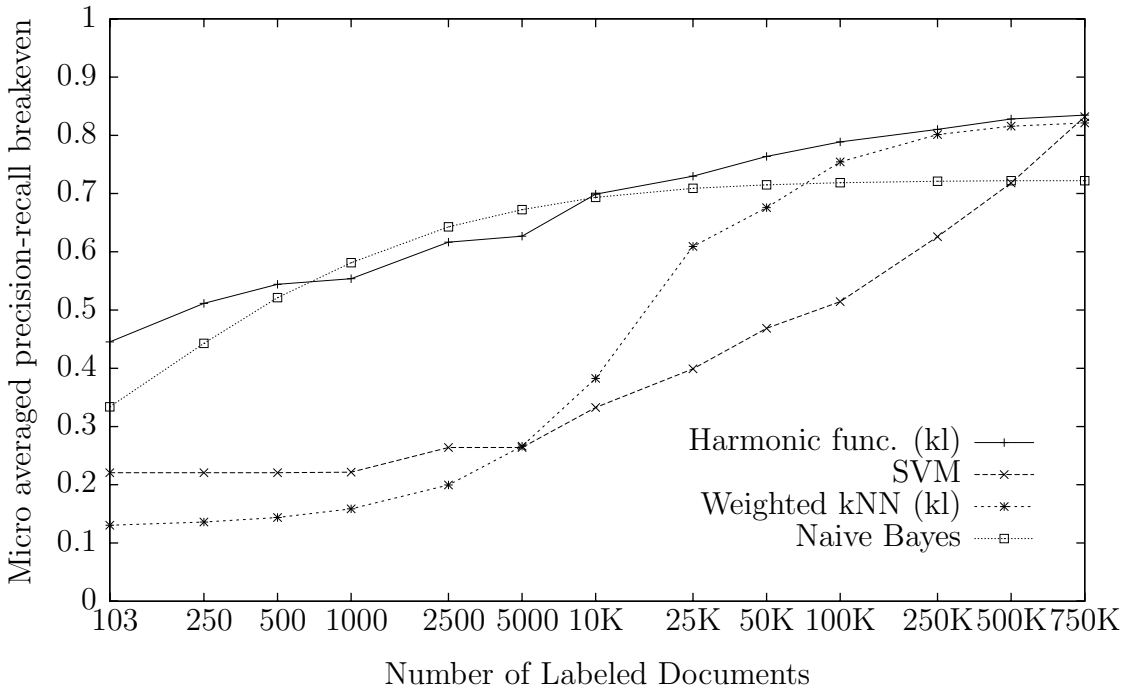

Figure 6: The performances of different algorithms on the RCV1 dataset with varying training data sizes.

- n is the number of categories.
- $\bar{d}$ is the average of all values of d_i for all i.
- s_d is the standard deviation of all values of d_i for all i.

We also define two hypotheses, the null hypothesis H_0, and an alternative hypothesis H_1 as follows:

- H_0 : the mean of the differences is not significantly different from 0
- $H_.$ the mean of the differences is significantly different from 0

Now, we define T as:

$$T = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{20}$$

We compare this value to the critical value of the t statistics from the t distribution tables to determine whether we can reject the null hypothesis or not. We applied the test to different pairs of algorithms on different data sizes. The results of applying the paired $t-test$ on the results obtained on the RCV1 dataset for 5000 labeled documents

is shown in Table 3. As the number of labeled data increase, the difference between different algorithms becomes less significant. With very large number of labeled data, all the differences are insignificant or barely significant.

Table 3: The Paired T-Test Results for different algorithms on the RCV1 dataset. " << ", " >> ", and " $\approx$ " in cell at row A and column B mean that the performance of algorithm A is significantly less than, greater than, or indistinguishable from the performance of algorithm B, respectively.

	w-kNN (cos)	w-kNN (KL)	v-kNN (cos)	v-kNN (KL)	Harm. (cos)	Harm. (KL)	Naive Bayes	SVM
w-kNN (cos)	$\approx$	<<	$\approx$	<<	<<	<<	<<	<<
w-kNN (KL)	>>	$\approx$	>>	$\approx$	<<	<<	<<	$\approx$
v-kNN (cos)	$\approx$	<<	$\approx$	<<	<<	<<	<<	<<
v-kNN (KL)	>>	$\approx$	>>	$\approx$	<<	<<	<<	$\approx$
Harm. (cos)	>>	>>	>>	>>	$\approx$	<<	<<	>>
Harm. (KL)	>>	>>	>>	>>	>>	$\approx$	$\approx$	>>
Naive Bayes	>>	>>	>>	>>	>>	$\approx$	$\approx$	>>
SVM	>>	$\approx$	>>	$\approx$	<<	<<	<<	$\approx$

6. Conclusion

We have presented an extensive evaluation of several nearest neighbor methods for text classification. Despite their simplicity, nearest neighbor based approaches perform quite well on text classification. We have shown that this performance can be improved even more by making use of new similarity metrics motivated by the language modeling approach in information retrieval. The results are much better than Naive Bayes and very competitive with the state-of-the-art SVM and TSVM. This paper has shown that nearest neighbor methods are not simple baselines but actually strong competitive learning algorithms for text classification.

The nearest neighbor framework can be extended to the semi-supervised case. We have shown that this results in an algorithm that is equivalent to the semi-supervised learning harmonic functions and presented the first extensive evaluation of this algorithm on text categorization. The results, especially with limited training data, are remarkable surpassing SVM and TSVM by a wide margin on the 20 Newsgroups dataset and the RCV1 dataset. It also performs competitively on the Reuters-21578 dataset.

References

Aas, K., & Eikvil, L. (1999). Text categorisation: A survey..

Bekkerman, R., McCallum, A., & Huang, G. (2004). Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Tech. rep. IR-418, Center of Intelligent Information Retrieval, UMass Amherst.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J., & Roli, F. (Eds.), *Multiple Classifier Systems*, Vol. 1857 of *Lecture Notes in Computer Science*, pp. 1–15. Springer.

Doyle, P. G., & Snell, J. L. (1984). *Random Walks and Electric Networks*. Mathematical Association of America.

Erkan, G. (2006). Language model-based document clustering using random walks. In Moore, R. C., Bilmes, J. A., Chu-Carroll, J., & Sanderson, M. (Eds.), *HLT-NAACL*. The Association for Computational Linguistics.

Fradkin, D., & Kantor, P. (2005). Methods for learning classifier combinations: no clear winner. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pp. 1038–1043, New York, NY, USA. ACM.

Galavotti, L., Sebastiani, F., & Simi, M. (2000). Experiments on the use of feature selection and negative evidence in automated text categorization. In *ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 59–68, London, UK. Springer-Verlag.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*. Springer.

Joachims, T. (1999a). Making large–scale SVM learning practical. In Schölkopf, B., Burges, C. J. C., & Smola, A. J. (Eds.), *Advances in Kernel Methods — Support Vector Learning*, pp. 169–184, Cambridge, MA. MIT Press.

Joachims, T. (1999b). Transductive inference for text classification using support vector machines. In Bratko, I., & Dzeroski, S. (Eds.), *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pp. 200–209, Bled, SL. Morgan Kaufmann Publishers, San Francisco, US.

Kurland, O., & Lee, L. (2005). Pagerank without hyperlinks: structural re-ranking using links induced by language models. In Baeza-Yates, R. A., Ziviani, N., Marchionini, G., Moffat, A., & Tait, J. (Eds.), *SIGIR*, pp. 306–313. ACM.

Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 111–119.

Lewis, D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*(1), 361–397.

Liu, X., & Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pp. 186–193.

Mock, K. (2001). An experimental framework for email categorization and management. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 392–393.

Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*(2/3), 103.

Ogilvie, P., & Callan, J. P. (2001). Experiments using the lemur toolkit. In *TREC*.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *SIGIR*, pp. 275–281. ACM.

Rennie, J. D. M. (2001). Improving multi-class text classification with naive bayes. In *MIT AI-TR*.

Rennie, J., Shih, L., Teevan, J., & Karger, D. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML-03, 20th International Conference on Machine Learning*, Washington, DC. Morgan Kaufmann Publishers, San Francisco, US.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2004). On maximum margin hierarchical multilabel classification. In *NIPS Workshop on Learning with Structured Outputs*.

Salojärvi, J., Puolamäki, K., & Kaski, S. (2005). On discriminative joint density modeling. In *16th European Conference on Machine Learning (ECML)*.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, *34*(1), 1–47.

Stevenson, R., & Whitehead, M. (2002). The reuters corpus volume 1 - from yesterday's news to tomorrows's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1-2), 69–90.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Speech IR & Text Categorization, pp. 42–49.

Zelikovitz, S., & Hirsh, H. (2001). Improving text classification with lsi using background knowledge. In *IJCAI01 Workshop Notes on Text Learning: Beyond Supervision*.

Zelikovitz, S., & Hirsh, H. (2002). Integrating background knowledge into nearest neighbor text classification. In *Proceedings of the Sixth European Conference on Case Based Reasoning (ECCBR)*.

Zelikovitz, S., & Marquez, F. (2005). Evaluation of background knowledge for latent semantic indexing classification. In *The FLorida Artificial Intelligence Research Society (FLAIRS)*.

Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst. (TOIS)*, *22*(2), 179–214.

Zhang, J. (2005). Sparsity models for multi-task learning. In *NIPS Workshop on Inductive Transfer*.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In Fawcett, T., & Mishra, N. (Eds.), *ICML*, pp. 912–919. AAAI Press.