Improved Nearest Neighbor Methods For Text Classification

Güneş Erkan[1], Ahmed Hassan[1], Qian Diao[2], and Dragomir Radev[1]

[1]Department of EECS, University of Michigan, Ann Arbor, MI, USA
[2]Intel Corporation, Santa Clara, CA, USA

October 26, 2011

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 traditional nearest neighbor methods and produce competitive results when compared to the state-of-the-art methods such as Support Vector Machines (SVM) and transductive SVM on the Reuters-21578 dataset, the 20 Newsgroups dataset, and the Reuters Corpus Volume I (RCV1) dataset. To our knowledge, this paper presents one of 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. Even the simplest NN methods such as k-nearest neighbor algorithm (kNN) have been shown to perform surprisingly well in text classification [7, 24]. 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 [1, 11]. 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 [29] 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 [14, 2]. 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 2.1 presents a brief overview of related work. In Section 2.2, 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 2.3 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 neighbor methods against each other as well as against Naive Bayes and Support Vector Machines in Section 3.

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

2 Materials and Methods

2.1 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.

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 [21].

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)[22, 12]. 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 [7, 24].

[26] 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 [25, 27]. They evaluate their approaches on a set of small datasets including technical papers, news documents, and web pages.

[6] 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 do experiments using kNN with different similarity functions and 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 classification [22, 12]. 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.

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

2.2 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: [28, 13]:

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

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:

$$y_{\mathrm{NB}}(d) = \operatorname*{argmax}_c \, p(c) \cdot \prod_{w \in d} p(w|c)^{tf(w,d)} \tag{6}$$

2.3 Similarity-based Approaches

2.4 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:

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

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:

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

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

2.4.1 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 [18], 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:

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

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.

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 [18]. We can turn it into a document similarity measure by replacing q in Equation 9 with a document so that $p_{\text{LM}}(d|d')$ 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 [3]. Weighted kNN in combination with the LM similarity 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 [17, 3].

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 [19] 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 three datasets we use in this paper (see Section 3.1), 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 [11]. 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:

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

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

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

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 $\exp(-\mathrm{KL}(\cdot|\cdot))$ function has been used as a document-document similarity in recent research and has helped improve the quality of document retrieval [10] and document clustering [5].

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

2.4.2 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 [4]. Harmonic functions were first introduced as a semi-supervised learning method by Zhu et. al. [29]. 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

3 Results

3.1 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) that contains $18,828$ documents after removing the duplicates in the original dataset which has $19,997$

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

documents. All the headers except for "From" and "Subject" are removed in the newsgroup articles. 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 [7, 24, 15, 20], 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[22, 12]. 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 [16]. We used cross validation to select a value for the Dirichlet smoothing parameter while computing the generation probabilities and KL divergence. We tried different values for this parameter ranging from 500 to 3000 using the Reuters-21578 dataset. We used statistical significance tests to compare the performance at different values. We found out that the differences due to varying this parameter in this range are statistically insignificant. We set this parameter to the default value suggested by the Lemur toolkit [16], 1000, through all our experiments.

3.2 Implementation Details

Note that the generation probabilities and the KL divergence based similarity measure are not symmetric. As mentioned in Section 2.4.1, 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.

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.

We do cross-validation on each method/dataset pair to compare the performance at different values of the number of neighbors k. We used the following values for k: 10, 20, 30, and 40. We did 10 fold cross-validation on each method/dataset pair for each value of k. We compared the performance at each value of k using statical significance tests. We found out the following:

1. For some method/dataset pairs, the difference between $k = 20$, 30, or 40 is statistically insignificant and all are better than $k = 10$.

2. For other method/dataset pairs, the difference between $k = 20$, or 30 is statistically insignificant and all are better than $k = 10$, or 40.

3. For the rest of method/dataset pairs, the difference between $k = 30$, or 40 is statistically insignificant and all are better than $k = 10$, or 20.

As a result we set the number of neighbors k to 30 in all of our nearest neighbor based experiments.

To find the solution of the harmonic function in Equation 13, we use the Iterative Template Library , 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. [29], 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

$$z(d) = (1 - r) \cdot \frac{y(d)}{\sum_{d' \in U} y(d')} - r \cdot \frac{1 - y(d)}{\sum_{d' \in U} 1 - y(d')} \geq 0 \tag{14}$$

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 14. 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 cosine similarity, Weighted kNN with KL similarity, Weighted kNN with cosine similarity, Voting kNN with KL similarity,

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

Voting kNN with cosine 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.

Table 2. The F1 score 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	87.82	85.71	91.93	88.34	89.26	92.65	89.41	93.82	83.66
corn	72.06	76.47	66.67	63.64	66.67	63.49	49.21	82.54	84.54
crude	79.04	79.04	86.35	83.81	80	81.18	77.06	82.94	82.54
earn	94.69	92.53	96.65	95.93	94.56	97.1	94.37	98	97.91
grain	81.6	83.44	85.02	82.57	83.91	80.76	73.19	90.85	82.94
interest	72.79	71.32	78.23	73.8	76.98	79.25	61.13	81.51	73.85
money-fx	76.54	79.01	82.18	79.7	78.87	80.41	67.01	81.44	90.22
ship	76.83	76.83	74.36	74.36	76.36	77.58	75.15	83.64	83.02
trade	69.7	72.73	80.9	75.66	73.85	72.31	53.08	74.62	86.06
wheat	68.79	70.06	77.3	77.3	73.2	71.9	64.05	83.66	93.82
microavg. (10)	86.02	85.17	89.67	87.52	87.13	88.92	83.08	91.47	91.72
microavg. (90)	80.37	80.26	82.42	80.72	80.68	82.25	73.14	86.1	86.15

Evaluation Metrics

Achieving high accuracy for each binary classification on the Reuters and RCV1 corpora is trivial since the negative class dominates the dataset. Therefore we consider precision and recall for each binary classification defined as follows:

$$\text{Precision} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}} \tag{15}$$

$$\text{Recall} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive documents}} \tag{16}$$

To combine precision and recall, we report the F1 score; which is the harmonic mean of precision and recall.

Table 3. The F1 score of various algorithms on a random split (2/3 training and 1/3 testing) of the 10 largest categories of the 20news dataset. Microaverages of these categories and all of the 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
C2	87.53	93.12	86.63	92.87	90.97	91.15	91.06	91.34	91.21
C3	74.18	82.84	72.49	85.57	87.99	88.62	86.04	89.00	88.66
C7	84.94	87.39	84.14	90.55	89.57	89.95	90.21	90.39	90.29
C8	87.74	90.03	86.71	91.18	90.24	90.81	90.89	91.02	90.73
C9	74.13	82.56	72.83	84.61	88.82	88.90	89.10	89.63	89.45
C10	87.37	90.39	86.77	91.15	89.80	90.18	89.58	90.51	90.30
C11	87.33	91.33	86.82	91.74	90.39	90.02	90.95	90.46	90.89
C13	87.02	91.01	86.99	92.40	90.56	90.61	90.68	91.02	91.27
C14	87.36	92.60	86.74	90.76	90.90	91.33	91.19	91.50	91.14
C15	87.81	90.32	86.71	92.81	89.77	89.89	91.15	90.42	90.51
microavg. (10)	84.54	89.16	83.68	90.36	89.90	90.15	90.09	90.53	90.44
microavg. (All)	83.34	87.52	82.44	89.10	89.65	89.85	89.55	90.22	90.18

To combine the scores of different binary classifications for the Reuters dataset, we use microaveraging [23], that is, pool per-document decisions across classes, and then computes the F1 score on the pooled contingency table. We test statstical significance using the the paired t-test.

Discussion

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 F1 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

Table 4. The F1 score of various algorithms on a random split (2/3 training and 1/3 testing) of the 10 largest categories of the RCV1 dataset. Microaverages of these categories and all of the 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
CCAT	65.8	66.28	65.65	66.01	93.13	94.39	90.51	92.50
M13	92.99	93.13	93.01	93.08	94.11	94.82	91.49	93.50
GPOL	91.69	91.84	91.68	91.59	88.53	88.97	87.36	89.42
GCAT	77.04	77.84	77.76	78.33	92.87	94.21	91.93	92.80
C152	88.96	89.62	88.81	89.61	81.29	84.57	82.17	83.62
MCAT	78.2	78.52	78.74	78.91	93.37	94.42	90.83	91.68
ECAT	84.91	85.17	85.03	85.25	89.16	90.68	90.38	89.89
C15	81.51	82.21	81.44	82.23	86.61	90.27	88.39	88.76
M14	89.77	89.81	89.84	89.76	96.52	96.96	96.36	96.92
C151	89.19	89.51	89.21	89.46	86.25	89.55	90.61	90.92
microavg. (10)	86.26	86.01	89.81	89.31	90.37	92.02	90.07	93.68
microavg. (All)	80.07	79.89	82.61	82.18	92.02	92.45	94.54	93.62

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 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 microaveraged F1 score 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 similarity performs better than the cosine similarity. Again the differences in the performance tend to shrink as we introduce more training documents.

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 2.2. We also perform experiments with the transductive SVM (TSVM) algorithm since it is one of the leading semi-supervised learning algorithms for text classification [9]. For SVM classification, we choose the linear SVM and use the SVM^{light} package as implemented by Joachims [8]. 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.Table 3 shows the same results for the 20news dataset.

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. We also notice that TSVM performs quite similar to SVM.

Figure 5 shows that harmonic functions 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. Figure 6 compares the performance of harmonic functions and weighted kNN with KL divergence, Naive Bayes, and SVM on the RCV1 dataset. It shows that harmonic functions are very competitive to SVM. Harmonic functions do better than SVM for some points where the number of training document is limited. SVM is better when the number of training documents increases. 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.

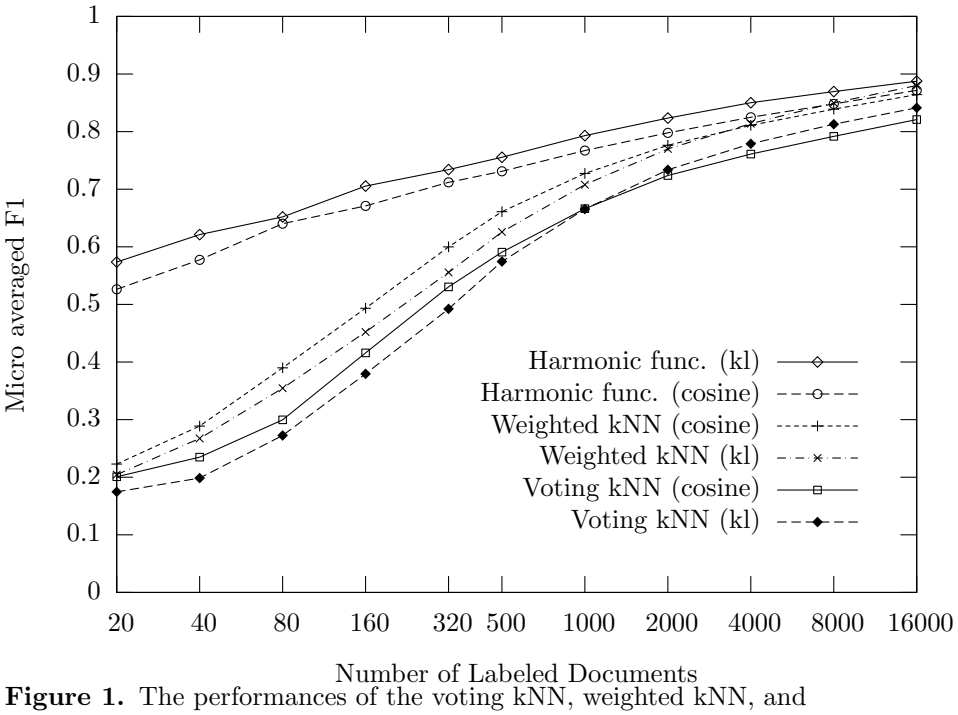

Figure 1. The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the 20 Newsgroups dataset.

4 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 an extensive evaluation of this algorithm on text categorization. The results show that the proposed algorithm performs competitively to SVM and TSVM on all the datasets. It also surpasses them when the the training data is limited.

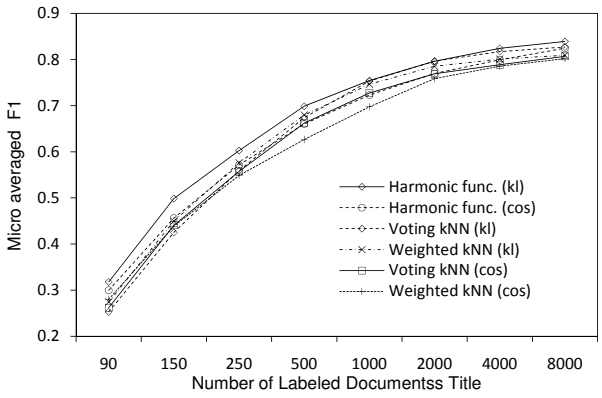

Figure 2. The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the Reuters-21578 dataset.

References

[1] Doug Baker and Andrew McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[2] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Technical Report IR-418, Center of Intelligent Information Retrieval, UMass Amherst, 2004.

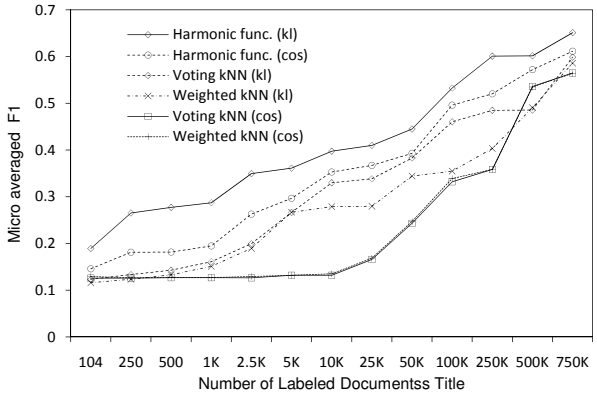

Figure 3. The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the RCV1 dataset.

[3] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[4] Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.

[5] Güneş Erkan. Language model-based document clustering using random walks. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics, 2006.

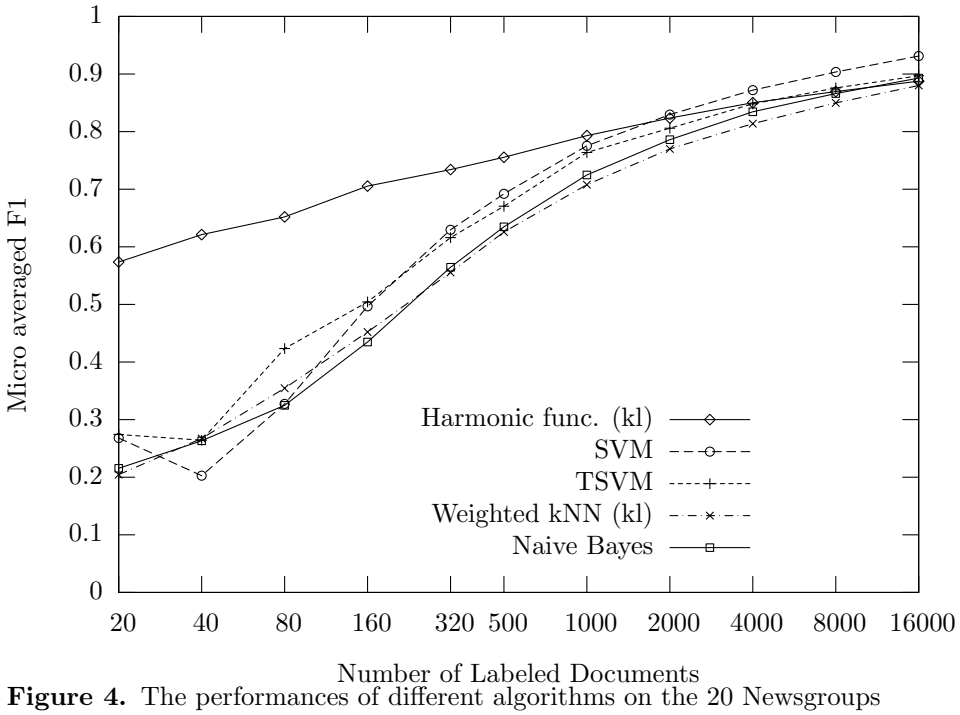

Figure 4. The performances of different algorithms on the 20 Newsgroups dataset with varying training data sizes.

[6] Luigi Galavotti, Fabrizio Sebastiani, and Maria Simi. 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*, pages 59–68, London, UK, 2000. Springer-Verlag.

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

[8] T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

[9] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

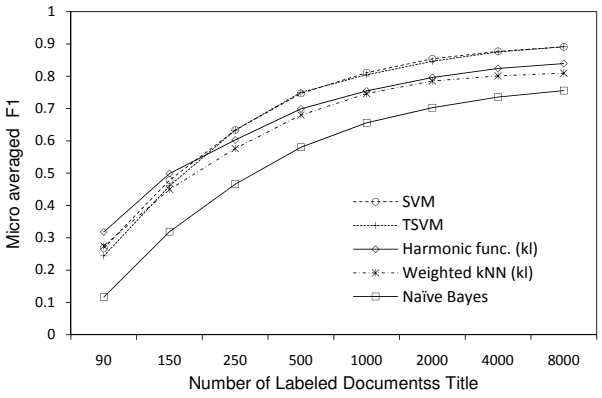

Figure 5. The performances of different algorithms on the Reuters-21578 dataset with varying training data sizes.

[10] Oren Kurland and Lillian Lee. Pagerank without hyperlinks: structural re-ranking using links induced by language models. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR*, pages 306–313. ACM, 2005.

[11] John Lafferty and Chengxiang Zhai. 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*, pages 111–119, 2001.

[12] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*,

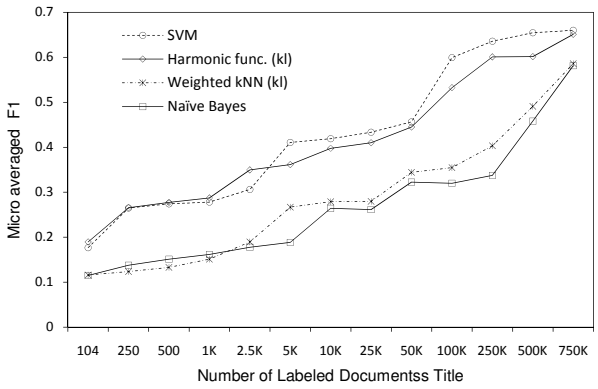

Figure 6. The performances of different algorithms on the RCV1 dataset with varying training data sizes.

5(1):361–397, 2004.

[13] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pages 186–193, 2004.

[14] Kenrick Mock. 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*, pages 392–393, 2001.

[15] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103, 2000.

[16] Paul Ogilvie and James P. Callan. Experiments using the lemur toolkit. In *TREC*, 2001.

[17] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[18] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281. ACM, 1998.

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

[20] Jason Rennie, Lawrence Shih, Jaime Teevan, and David Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML-03, 20th International Conference on Machine Learning*, Washington, DC, 2003. Morgan Kaufmann Publishers, San Francisco, US.

[21] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[22] R. Stevenson and M. Whitehead. 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*, 2002.

[23] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[24] Yiming Yang and Xin Liu. 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, pages 42–49, 1999.

[25] Sarah Zelikovitz and Haym Hirsh. Improving text classification with lsi using background knowledge. In *IJCAI01 Workshop Notes on Text Learning: Beyond Supervision*, 2001.

[26] Sarah Zelikovitz and Haym Hirsh. Integrating background knowledge into nearest neighbor text classification. In *Proceedings of the Sixth European Conference on Case Based Reasoning (ECCBR)*, 2002.

[27] Sarah Zelikovitz and Finella Marquez. Evaluation of background knowledge for latent semantic indexing classification. In *The FLorida Artificial Intelligence Research Society (FLAIRS)*, 2005.

[28] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst. (TOIS)*, 22(2):179–214, 2004.

[29] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 912–919. AAAI Press, 2003.