Using the Mutual k-Nearest Neighbor Graphs for Semi-supervised Classification of Natural Language Data

Kohei Ozaki and **Masashi Shimbo** and **Mamoru Komachi** and **Yuji Matsumoto**
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{kohei-o,shimbo,komachi,matsu}@is.naist.jp

Abstract

The first step in graph-based semi-supervised classification is to construct a graph from input data. While the k-nearest neighbor graphs have been the de facto standard method of graph construction, this paper advocates using the less well-known *mutual* k-nearest neighbor graphs for high-dimensional natural language data. To compare the performance of these two graph construction methods, we run semi-supervised classification methods on both graphs in word sense disambiguation and document classification tasks. The experimental results show that the mutual k-nearest neighbor graphs, if combined with maximum spanning trees, consistently outperform the k-nearest neighbor graphs. We attribute better performance of the mutual k-nearest neighbor graph to its being more resistive to making hub vertices. The mutual k-nearest neighbor graphs also perform equally well or even better in comparison to the state-of-the-art b-matching graph construction, despite their lower computational complexity.

1 Introduction

Semi-supervised classification try to take advantage of a large amount of unlabeled data in addition to a small amount of labeled data, in order to achieve good classification accuracy while reducing the cost of manually annotating data. In particular, graph-based techniques for semi-supervised classification (Zhou et al., 2004; Zhu et al., 2003; Callut et al., 2008; Wang et al., 2008) are recognized as a promising approach. Some of these techniques have been successfully applied for NLP tasks: word sense disambiguation (Alexandrescu and Kirchhoff, 2007; Niu et al., 2005), sentiment analysis (Goldberg and Zhu, 2006), and statistical machine translation (Alexandrescu and Kirchhoff, 2009), to name but a few.

However, the focus of these studies is how to assign accurate labels to vertices in a given graph. By contrast, there has not been much work on how such a graph should be built, and graph construction remains "more of an art than a science" (Zhu, 2005). Yet, it is an essential step for graph-based semi-supervised classification and (unsupervised) clustering, and the input graph affects the quality of final classification/clustering results.

Both for semi-supervised classification and for clustering, the k-nearest neighbor (k-NN) graph construction has been used almost exclusively in the literature. However, k-NN graphs often produce *hubs*, or vertices with extremely high degree (i.e., the number of edges incident to a vertex). This tendency is obvious especially if the original data is high-dimensional—a characteristic typical of natural language data. In a later section, we demonstrate that such hub vertices indeed deteriorate the accuracy of semi-supervised classification.

While not in the context of graph construction, Radovanović et al. (2010) made an insightful observation into the nature of hubs in high-dimensional space; in their context, a hub is a sample close to many other samples in the (high-dimensional) sample space. They state that such hubs inherently emerge in high-dimensional data as a side effect of the "curse of dimensionality," and argue that this is a

Proceedings of the Fifteenth Conference on Computational Natural Language Learning, pages 154–162,
Portland, Oregon, USA, 23–24 June 2011. ©2011 Association for Computational Linguistics

reason nearest neighbor classification does not work well in high-dimensional space.

Their observation is insightful for graph construction as well. Most of the graph-based semi-supervised classification methods work by gradually propagating label information of a vertex towards neighboring vertices in a graph, but the neighborhood structure in the graph is basically determined by the proximity of data in the original high-dimensional sample space. Hence, it is very likely that a hub in the sample space also makes a hub in the k-NN graph, since k-NN graph construction greedily connects a pair of vertices if the sample corresponding to one vertex is among the k closest samples of the other sample in the original space. It is therefore desirable to have an efficient graph construction method for high-dimensional data that can produce a graph with reduced hub effects.

To this end, we propose to use the *mutual k-nearest neighbor graphs* (*mutual k-NN graphs*), a less well-known variant of the standard k-NN graphs. All vertices in a mutual k-NN graph have a degree upper-bounded by k, which is not usually the case with standard k-NN graphs. This property helps not to produce vertices with extremely high degree (hub vertices) in the graph. A mutual k-NN graph is easy to build, at a time complexity identical to that of the k-NN graph construction.

We first evaluated the quality of the graphs apart from specific classification algorithms using the ϕ-edge ratio of graphs. Our experimental results show that the mutual k-NN graphs have a smaller number of edges connecting vertices with different labels than the k-NN graphs, thus reducing the possibility of wrong label information to be propagated. We also compare the classification accuracy of two standard semi-supervised classification algorithms on the mutual k-NN graphs and the k-NN graphs. The results show that the mutual k-NN graphs consistently outperorm the k-NN graphs. Moreover, the mutual k-NN graphs achieve equally well or better classification accuracy than the state-of-the-art graph construction method called b-matching (Jebara et al., 2009), while taking much less time to construct.

2 Problem Statement

2.1 Semi-supervised Classification

The problem of semi-supervised classification can be stated as follows. We are given a set of n examples, $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, but only the labels of the first l examples are at hand; the remaining $u = n - l$ examples are unlabeled examples. Let $S = \{1, \ldots, c\}$ be the set of possible labels, and $y_i \in S$ the label of $\boldsymbol{x}_i$, for $i = 1, \ldots, n$. Since we only know the labels of the first l examples, we do not have access to $y_{l+1}, \ldots, y_n$. For later convenience, further let $\boldsymbol{y} = (y_1, \ldots, y_n)$.

The goal of a semi-supervised classification algorithm is to predict the hidden labels $y_{l+1}, \ldots, y_n$ of u unlabeled examples $\boldsymbol{x}_{l+1}, \ldots, \boldsymbol{x}_n$, given these unlabeled examples and l labeled data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)$. A measure of similarity between examples is also provided to the algorithm. Stated differently, the classifier has access to an all-pair similarity matrix W' of size $n \times n$, with its (i, j)-element W'_{ij} holding the similarity of examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. It is assumed that W' is a symmetric matrix, and the more similar two examples are (with respect to the similarity measure), more likely they are to have the same label. This last assumption is the premise of many semi-supervised classification algorithms and is often called the cluster assumption (Zhou et al., 2004).

2.2 Graph-based Semi-supervised Classification

Graph-based approaches to semi-supervised classification are applicable if examples X are graph vertices. Otherwise, X must first be converted into a graph. This latter case is the focus of this paper. That is, we are interested in how to construct a graph from the examples, so that the subsequent classification works well.

Let $\mathcal{G}$ denote the graph constructed from the examples. Naturally, $\mathcal{G}$ has n vertices, since vertices are identified with examples. Instead of graph $\mathcal{G}$ itself, let us consider its real-valued (weighted) adjacency matrix W, of size $n \times n$. The task of graph construction then reduces to computing W from all-pairs similarity matrix W'.

The simplest way to compute W from W' is to let $W = W'$, which boils down to using a dense,

complete graph $\mathcal{G}$ with the unmodified all-pairs similarity as its edge weights. However, it has been observed that a sparse W not only save time needed for classification, but also results in better classification accuracy[1] than the full similarity matrix W' (Zhu, 2008). Thus, we are concerned with how to sparsify W' to obtain a sparse W; i.e., the strategy of zeroing out some elements of W'.

Let the set of binary values be $\mathbb{B} = \{0, 1\}$. A sparsification strategy can be represented by a binary-valued matrix $P \in \mathbb{B}^{n \times n}$, where $P_{ij} = 1$ if W'_{ij} must be retained as W_{ij}, and $P_{ij} = 0$ if $W_{ij} = 0$. Then, the weighted adjacency matrix W of $\mathcal{G}$ is given by $W_{ij} = P_{ij}W'_{ij}$. The $n \times n$ matrices W and P are symmetric, reflecting the fact that most graph-based algorithms require the input graph to be undirected.

3 k-Nearest Neighbor Graphs and the Effect of Hubs

The standard approach to making a sparse graph $\mathcal{G}$ (or equivalently, matrix W) is to construct a k-NN graph from the data (Szummer and Jaakkola, 2002; Niu et al., 2005; Goldberg and Zhu, 2006).

3.1 The k-Nearest Neighbor Graphs

The k-NN graph is a weighted undirected graph connecting each vertex to its k-nearest neighbors in the original sample space. Building a k-NN graph is a two step process. First we solve the following optimization problem.

$$\max_{\hat{P} \in \mathbb{B}^{n \times n}} \sum_{i,j} \hat{P}_{ij} W'_{ij} \quad (1)$$
$$\text{s.t.} \sum_{j} \hat{P}_{ij} = k,\ \hat{P}_{ii} = 0,\ \forall i, j \in \{1, \ldots, n\}$$

Note that we are trying to find $\hat{P}$, and not P. This is an easy problem and we can solve it by greedily assigning $\hat{P}_{ij} = 1$ only if W'_{ij} is among the top k elements in the ith row of W' (in terms of the magnitude of the elements). After $\hat{P}$ is determined, we let $P_{ij} = \max(\hat{P}_{ij}, \hat{P}_{ji})$. Thus P is a symmetric matrix, i.e., $P_{ij} = P_{ji}$ for all i and j, while $\hat{P}$ may not. Finally, weighted adjacency matrix W is determined by $W_{ij} = P_{ij}W'_{ij}$. Matrix W is also symmetric since P and W' are symmetric.

This process is equivalent to retaining all edges from each vertex to its k-nearest neighbor vertices, and then making all edges undirected.

Note the above symmetrization step is necessary because the k-nearest neighbor relation is not symmetric; even if a vertex v_i is a k-nearest neighbor of another vertex v_j, v_j may or may not be a k-nearest neighbor of v_i. Thus, symmetrizing P and W as above makes the graph irregular; i.e., the degree of some vertices may be larger than k, which opens the possibility of hubs to emerge.

d	1	2	≥ 3	total
# of vertices	1610	1947	164	3721
original	65.9	65.7	**69.8**	66.0
hub-removed	**66.6**	**66.0**	**69.8**	**66.4**

Table 1: Classification accuracy of vertices around hubs in a k-NN graph, before ("original") and after ("hub-removed") hubs are removed. The value d represents the shortest distance (number of hops) from a vertex to its nearest hub vertex in the graph.

3.2 Effect of Hubs on Classification

In this section, we demonstrate that hubs in k-NN graphs are indeed harmful to semi-supervised classification as we claimed earlier. To this end, we eliminate such high degree vertices from the graph, and compare the classification accuracy of other vertices before and after the elimination. For this preliminary experiment, we used the "line" dataset of a word sense disambiguation task (Leacock et al., 1993). For details of the dataset and the task, see Section 6.

In this experiment, we randomly selected 10 percent of examples as labeled examples. The remaining 90 percent makes the set of unlabeled examples, and the goal is to predict the label (word sense) of these unlabeled examples.

We first built a k-NN graph (with $k = 3$) from the dataset, and ran Gaussian Random Fields (GRF) (Zhu et al., 2003), one of the most widely-used graph-based semi-supervised classification algorithms. Then we removed vertices with degree

[1] See also the experimental results of Section 6.3.2 in which the full similarity matrix W' is used as the baseline.

greater than or equal to 30 from the k-NN graph, and ran GRF again on this "hub-removed" graph.

Table 1 shows the classification accuracy of GRF on the two graphs. The table shows both the overall classification accuracy, and the classification accuracy on the subsets of vertices, stratified by their distance d from the nearest hub vertices (which were eliminated in the "hub-removed" graph). Obviously, overall classification accuracy has improved after hub removal. Also notice that the increase in the classification accuracy on the vertices nearest to hubs ($d = 1, 2$). These results suggest that the presence of hubs in the graph is deteriorating classification accuracy.

4 Mutual k-Nearest Neighbor Graphs for Semi-supervised Classification

As demonstrated in Section 3.2, removing hub vertices in k-NN graphs is an easy way of improving the accuracy of semi-supervised classification. However, this method adds another parameter to the graph construction method, namely, the threshold on the degree of vertices to be removed. The method also does not tell us how to assign labels to the removed (hub) vertices. Hence, it is more desirable to have a graph construction method which has only one parameter just like the k-NN graphs, but is at the same time less prone to produce hub vertices.

In this section, we propose to use mutual k-NN graphs for this purpose.

4.1 Mutual k-Nearest Neighbor Graphs

The mutual k-NN graph is not a new concept and it has been used sometimes in clustering. Even in clustering, however, they are not at all as popular as the ordinary k-NN graphs. A mutual k-NN graph is defined as a graph in which there is an edge between vertices v_i and v_j if each of them belongs to the k-nearest neighbors (in terms of the original similarity metric W) of the other vertex. By contrast, a k-NN graph has an edge between vertices v_i and v_j if one of them belongs to the k-nearest neighbors of the other. Hence, the mutual k-NN graph is a subgraph of the k-NN graph computed from the same data with the same value of k. The mutual k-NN graph first optimizes the same formula as (1), but in mutual k-NN graphs, the binary-valued symmetric matrix P is defined as $P_{ij} = \min(\hat{P}_{ij}, \hat{P}_{ji})$. Since mutual k-NN graph construction guarantees that all vertices in the resulting graph have degree at most k, it is less likely to produce extremely high degree vertices in comparison with k-NN graphs, provided that the value of k is kept adequately small.

4.2 Fixing Weak Connectivity

Because the mutual k-NN graph construction is more selective of edges than the standard k-NN graphs, the resulting graphs often contain many small disconnected components. Disconnected components are not much of a problem for clustering (since its objective is to divide a graph into discrete components eventually), but can be a problem for semi-supervised classification algorithms; if a connected component does not contain a labeled node, the algorithms cannot reliably predict the labels of the vertices in the component; recall that these algorithms infer labels by propagating label information along edges in the graph.

As a simple method for overcoming this problem, we combine the mutual k-NN graph and the maximum spanning tree. To be precise, the minimum number of edges from the maximum spanning tree are added to the mutual k-NN graph to ensure that only one connected component exists in a graph.

4.3 Computational Efficiency

Using a Fibonacci heap-based implementation (Fredman and Tarjan, 1987), one can construct the standard k-NN graph in (amortized) $O(n^2 + kn \log n)$ time. A mutual k-NN graph can also be constructed in the same time complexity as the k-NN graphs. The procedure below transforms a standard k-NN graph into a mutual k-NN graph. It uses Fibonacci heaps once again and assumes that the input k-NN graph is represented as an adjacency matrix in sparse matrix representation.

1. Each vertex is associated with its own heap. For each edge e connecting vertices u and v, insert e to the heaps associated with u and v.

2. Fetch maximum weighted edges from each heap k times, keeping globally the record of the number of times each edge is fetched. Notice that an edge can be fetched at most twice,

once at an end vertex of the edge and once at the other end.

3. A mutual k-NN graph can be constructed by only keeping edges fetched twice in the previous step.

The complexity of this procedure is $O(kn)$. Hence the overall complexity of building a mutual k-NN graph is dominated by the time needed to build the standard k-NN graph input to the system; i.e., $O(n^2 + kn \log n)$.

If we call the above procedure on an *approximate* k-NN graph which can be computed more efficiently (Beygelzimer et al., 2006; Chen et al., 2009; Ram et al., 2010; Tabei et al., 2010), it yields an approximate mutual k-NN graphs. In this case, the overall complexity is identical to that of the approximate k-NN graph construction algorithm, since these approximate algorithms have a complexity at least $O(kn)$.

5 Related Work

5.1 b-Matching Graphs

Recently, Jebara et al. (2009) proposed a new graph construction method called *b-matching*. A b-matching graph is a b-regular graph, meaning that every vertex has the degree b uniformly. It can be obtained by solving the following optimization problem.

$$\max_{P \in \mathbb{B}^{n \times n}} \sum_{ij} P_{ij} W'_{ij}$$

$$\text{s.t.} \sum_{j} P_{ij} = b, \qquad \forall i \in \{1, \ldots, n\} \quad (2)$$

$$P_{ii} = 0, \qquad \forall i \in \{1, \ldots, n\} \quad (3)$$

$$P_{ij} = P_{ji}, \qquad \forall i, j \in \{1, \ldots, n\} \quad (4)$$

After P is computed, the weighted adjacency matrix W is determined by $W_{ij} = P_{ij} W'_{ij}$ The constraint (4) makes the binary matrix P symmetric, and (3) is to ignore self-similarity (loops). Also, the constraint (2) ensures that the graph is regular. Note that k-NN graphs are in general not regular. The regularity requirement of the b-matching graphs can be regarded as an effort to avoid the hubness phenomenon discussed by Radovanović et al. (2010).

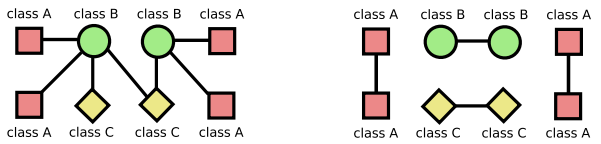

Figure 1: Two extreme cases of ϕ-edge ratio. Vertex shapes (and colors) denote the class labels. The ϕ-edge ratio of the graph on the left is 1, meaning that all edges connect vertices with different labels. The ϕ-edge ratio of the one on the right is 0, because all edges connect vertices of the same class.

Jebara et al. (2009) reported that b-matching graphs achieve semi-supervised classification accuracy higher than k-NN graphs. However, without approximation, building a b-matching graph is prohibitive in terms of computational complexity. Huang and Jebara (2007) developed a fast implementation based on belief propagation, but the guaranteed running time of the implementation is $O(bn^3)$, which is still not practical for large scale graphs. Notice that the k-NN graphs and mutual k-NN graphs can be constructed with much smaller time complexity, as we mentioned in Section 4.3. In Section exp, we empirically compare the performance of mutual k-NN graphs with that of b-matching graphs.

5.2 Mutual Nearest Neighbor in Clustering

In the clustering context, mutual k-NN graphs have been theoretically analyzed by Maier et al. (2009) with Random Geometric Graph Theory. Their study suggests that if one is interested in identifying the most significant clusters only, the mutual k-NN graphs give a better clustering result. However, it is not clear what their results imply in semi-supervised classification settings.

6 Experiments

We compare the k-NN, mutual k-NN, and b-matching graphs in word sense disambiguation and document classification tasks. All of these tasks are multi-class classification problems.

6.1 Datasets

We used two word sense disambiguation datasets in our experiment: "interest" and "line." The "interest" data is originally taken from the POS-tagged

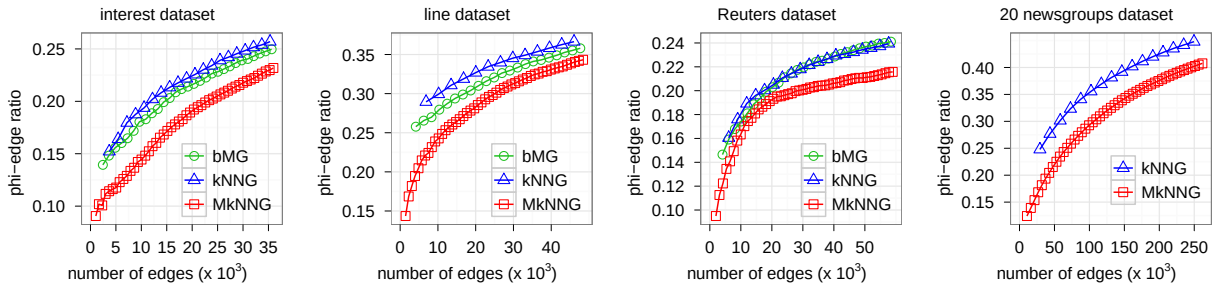

Figure 2: ϕ-edge ratios of the k-NN graph, mutual k-NN graph, and b-matching graphs. The ϕ-edge ratio of a graph is a measure of how much the cluster assumption is violated; hence, smaller the ϕ-edge ratio, the better. The plot for b-matching graph is missing for the 20 newsgroups dataset, because its construction did not finish after one week for this dataset.

dataset	examples	features	labels
interest	2,368	3,689	6
line	4,146	8,009	6
Reuters	4,028	17,143	4
20 newsgroups	19,928	62,061	20

Table 2: Datasets used in experiments.

portion of the Wall Street Journal Corpus. Each instance of the polysemous word "interest" has been tagged with one of the six senses in Longman Dictionary of Contemporary English. The details of the dataset are described in Bruce and Wiebe (1994). The "line" data is originally used in numerous comparative studies of word sense disambiguation. Each instance of the word "line" has been tagged with one of the six senses on the WordNet thesaurus. Further details can be found in the Leacock et al. (1993). Following Niu et al. (2005), we used the following context features in the word sense disambiguation tasks: part-of-speech of neighboring words, single words in the surrounding context, and local collocation. Details of these context features can be found in Lee and Ng (2002).

The Reuters dataset is extracted from RCV1-v2/LYRL2004, a text categorization test collection (Lewis et al., 2004). In the same manner as Crammer et al. (2009), we produced the classification dataset by selecting approximately 4,000 documents from 4 general topics (corporate, economic, government and markets) at random. The features described in Lewis et al. (2004) are used with this dataset.

The 20 newsgroups dataset is a popular dataset frequently used for document classification and clustering. The dataset consists of approximately 20,000 messages on newsgroups and is originally distributed by Lang (1995). Each message is assigned one of the 20 possible labels indicating which newsgroup it has been posted to, and represented as binary bag-of-words features as described in Rennie (2001).

Table 2 summarizes the characteristics of the datasets used in our experiments.

6.2 Experimental Setup

Our focus in this paper is a semi-supervised classification setting in which the dataset contains a small amount of labeled examples and a large amount of unlabeled examples. To simulate such settings, we create 10 sets of labeled examples, with each set consisting of randomly selected l examples from the original dataset, where l is 10 percent of the total number of examples. For each set, the remaining 90 percent constitute the unlabeled examples whose labels must be inferred.

After we build a graph from the data using one of the graph construction methods discussed earlier, a graph-based semi-supervised classification algorithm must be run on the resulting graph to infer labels to the unlabeled examples (vertices). We use two most frequently used classification algorithms: Gaussian Random Fields (GRF) (Zhu et al., 2003) and the Local/Global Consistency algorithm (LGC) (Zhou et al., 2004). Averaged classification accuracy is used as the evaluation metric. For all datasets, co-

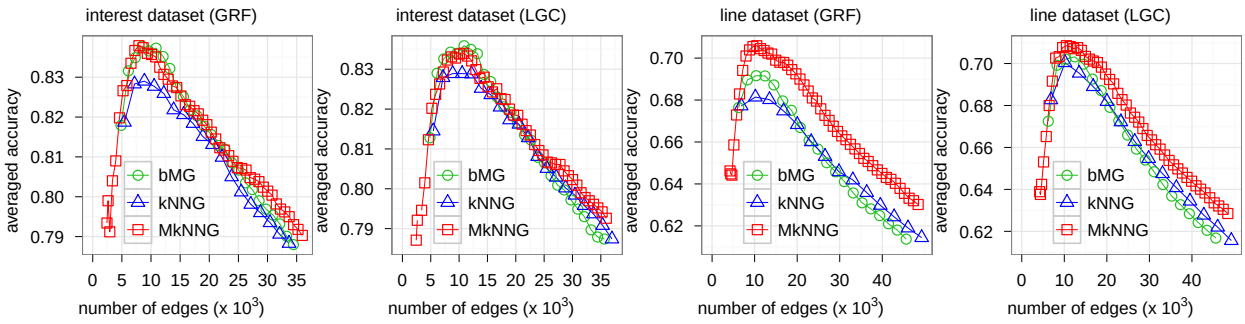

Figure 3: Averaged classification accuracies for k-NN graphs, b-matching graphs and mutual k-NN graphs (+ maximum spanning trees) in the interest and line datasets.

sine similarity is used as the similarity measure between examples.

In "interest" and "line" datasets, we compare the performance of the graph construction methods over the broad range of their parameters; i.e., b in b-matching graphs and k in (mutual) k-NN graphs.

In Reuters and the 20 newsgroups datasets, 2-fold cross validation is used to determine the hyperparameters (k and b) of the graph construction methods; i.e., we split the labeled data into two folds, and used one fold for training and the other for development, and then switch the folds in order to find the optimal hyperparameter among $k, b \in \{2, \ldots, 50\}$. The smoothing parameter μ of LGC is fixed at $\mu = 0.9$.

6.3 Results

6.3.1 Comparison of ϕ-Edge Ratio

We first compared the ϕ-edge ratios of k-NN graphs, mutual k-NN graphs, and b-matching graphs to evaluate the quality of the graphs apart from specific classification algorithms.

For this purpose, we define the *ϕ-edge ratio* as the yardstick to measure the quality of a graph. Here, a *ϕ-edge* of a labeled graph $(\mathcal{G}, \boldsymbol{y})$ is any edge (v_i, v_j) for which $y_i \neq y_j$ (Cesa-Bianchi et al., 2010), and we define the *ϕ-edge ratio* of a graph as the number of ϕ-edges divided by the total number of edges in the graph. Since most graph-based semi-supervised classification methods propagate label information along edges, edges connecting vertices with different labels may lead to misclassification. Hence, a graph with a smaller ϕ-edge ratio is more desirable. Figure 1 illustrates two toy graphs with extreme values of ϕ-edge ratio.

Figure 2 shows the plots of ϕ-edge ratios of the compared graph construction methods when the values of parameters k (for k-NN and mutual k-NN graphs) and b (for b-matching graphs) are varied. In these plots, the y-axes denote the ϕ-edge ratio of the constructed graphs. The x-axes denote the number of edges in the constructed graphs, and not the values of parameters k or b, because setting parameters b and k to an equal value does not achieve the same level of sparsity (number of edges) in the resulting graphs.

As mentioned earlier, the smaller the ϕ-edge ratio, the more desirable. As the figure shows, mutual k-NN graphs achieve smaller ϕ-edge ratio than other graphs if they are compared at the same level of graph sparsity.

The plot for b-matching graph is missing for the 20 newsgroups data, because we were unable to complete its construction in one week[2]. Meanwhile, a k-NN graph and a mutual k-NN graph for the same dataset can be constructed in less than 15 minutes on the same computer.

6.3.2 Classification Results

Figure 3 shows the classification accuracy of GRF and LGC on the different types of graphs constructed for the interest and line datasets. As in Figure 2, the x-axes represent the sparsity of the constructed graphs measured by the number of edges in the graph, which can change as the hyperparameter (b or k) of the compared graph construction methods

[2] All experiments were run on a machine with 2.3 GHz AMD Opteron 8356 processors and 256 GB RAM.

				kNN graph		b-matching graph		mutual kNN graph	
dataset	algorithm	Dense	MST	original	+MST	original	+MST	original	+MST
Reuters	GRF	43.65	72.74	81.70	80.89	84.04	84.04	**85.01**	84.72
Reuters	LGC	43.66	71.78	82.60	82.60	84.42	84.42	84.81	**84.85**
20 newsgroups	GRF	10.18	66.96	75.47	75.47	—	—	76.31	**76.46**
20 newsgroups	LGC	14.51	65.82	75.19	75.19	—	—	75.27	**75.41**

Table 3: Document classification accuracies for k-NN graphs, b-matching graphs, and mutual k-NN graphs. The column for 'Dense' is the result for the graph with the original similarity matrix W' as the adjacency matrix; i.e., without using any graph construction (sparsification) methods. The column for 'MST' is the result the for the maximum spanning tree. b-matching graph construction did not complete after one week on the 20 newsgroups data, and hence no results are shown.

	vs. kNNG		vs. bMG	
dataset (algo)	orig	+MST	orig	+MST
Reuters (GRF)	$\gg$	$\gg$	$>$	$\sim$
Reuters (LGC)	$\gg$	$\gg$	$\sim$	$\sim$
20 newsgroups (GRF)	$\gg$	$\gg$	—	—
20 newsgroups (LGC)	$\sim$	$>$	—	—

Table 4: One-sided paired t-test results of averaged accuracies between using mutual k-NN graphs and other graphs. "$\gg$", "$>$", and "$\sim$" correspond to p-value $<$ 0.01, $(0.01, 0.05]$, and > 0.05 respectively.

are varied.

As shown in the figure, the combination of mutual k-NN graphs and the maximum spanning trees achieves better accuracy than other graph construction methods in most cases, when they are compared at the same levels of graph sparsity (number of edges).

Table 3 summarizes the classification accuracy on the document classification datasets. As a baseline, the table also shows the results ('Dense') on the dense complete graph with the original all-pairs similarity matrix W' as the adjacency matrix (i.e., no graph sparsification), as well as the results for using the maximum spanning tree alone as the graph construction method.

In all cases, mutual k-NN graphs achieve better classification accuracy than other graphs.

Table 4 reports the one-sided paired t-test results of averaged accuracies with k-NN graphs and b-matching graphs against our proposed approach, the combination of mutual k-NN graphs and maximum spanning trees. From Table 4, we see that mutual k-NN graphs perform significantly better than k-NN graphs. On the other hand, theere is no significant difference in the accuracy of the mutual k-NN graphs and b-matching graphs. However, mutual k-NN graphs achieves the same level of accuracy with b-matching graphs, at much less computation time and are applicable to large datasets. As mentioned earlier, mutual k-NN graphs can be computed with less than 15 minutes in the 20 newsgroups data, while b-matching graphs cannot be computed in one week.

7 Conclusion

In this paper, we have proposed to use mutual k-NN graphs instead of the standard k-NN graphs for graph-based semi-supervised learning. In mutual k-NN graphs, all vertices have degree upper bounded by k. We have demonstrated that this type of graph construction alleviates the hub effects stated in Radovanović et al. (2010), which also makes the graph more consistent with the cluster assumption. In addition, we have shown that the weak connectivity of mutual k-NN graphs is not a serious problem if we augment the graph with maximum spanning trees. Experimental results on various natural language processing datasets show that mutual k-NN graphs lead to higher classification accuracy than the standard k-NN graphs, when two popular label inference methods are run on these graphs.

References

Andrei Alexandrescu and Katrin Kirchhoff. 2007. Data-driven graph construction for semi-supervised graph-based learning in NLP. In *Proc. of HLT-NAACL.*

Andrei Alexandrescu and Katrin Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *Proc. of NAACL-HLT*.

Alina Beygelzimer, Sham Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *Proc. of ICML*.

Rebecca Bruce and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. In *Proc. of ACL*.

Jérôme Callut, Kevin Françoisse, Marco Saerens, and Pierre Dupont. 2008. Semi-supervised classification from discriminative random walks. In *Proc. of ECML-PKDD*.

Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. 2010. Random spanning trees and the prediction of weighted graphs. In *Proc. of ICML*.

Jie Chen, Haw-ren Fang, and Yousef Saad. 2009. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10.

Koby Crammer, Mark Dredze, and Alex Kulesza. 2009. Multi-class confidence weighted algorithms. In *Proc. of EMNLP*.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, July.

Andrew B. Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *Proc. of TextGraphs Workshop on HLT-NAACL*.

Bert Huang and Tony Jebara. 2007. Loopy belief propagation for bipartite maximum weight b-matching. In *Proc. of AISTATS*.

Tony Jebara, Jun Wang, and Shih-Fu Chang. 2009. Graph construction and b-matching for semi-supervised learning. In *Proc. of ICML*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proc. of ICML*.

Claudia Leacock, Geoffrey Towell, and Ellen Voorhees. 1993. Corpus-based statistical sense resolution. In *Proc. of ARPA Workshop on HLT*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proc. of EMNLP*.

David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, G. Dietterich, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5.

Markus Maier, Matthias Hein, and Ulrike von Luxburg. 2009. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Journal of Theoretical Computer Science*, 410.

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *Proc. of ACL*.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hub in space: popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11.

Parikshit Ram, Dongryeol Lee, William March, and Alexander Gray. 2010. Linear-time algorithms for pairwise statistical problems. In *Proc. of NIPS*.

Jason D. M. Rennie. 2001. Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology. AITR-2001-004.

Martin Szummer and Tommi Jaakkola. 2002. Partially labeled classification with markov random walks. In *Proc. of NIPS*.

Yasuo Tabei, Takeaki Uno, Masashi Sugiyama, and Koji Tsuda. 2010. Single versus multiple sorting in all pairs similarity search. In *Proc. of ACML*.

Jun Wang, Tony Jebara, and Shih-Fu. Chang. 2008. Graph transduction via alternating minimization. In *Proc. of ICML*.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Proc. of NIPS*.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML*.

Xiaojin Zhu. 2005. *Semi-Supervised Learning with Graphs*. Ph.D. thesis, Carnegie Mellon University. CMU-LTI-05-192.

Xiaojin Zhu. 2008. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.