Classification using random walks with binary features

I. M. N. Author

Abstract

We present a new algorithm for classification based on Markov random walks. We evaluate our method, TUMBL, on the ppattach prepositional phrase attachment data set and report top performance when the amount of training data is severely limited.

1 Introduction

A number of problems in Natural Language Processing (NLP) can be cast as classification problems in which an object has to be labeled with one of a small number of possible classes, based on some properties (features) of that object. Such problems include word sense disambiguation, extractive summarization, and text classification. In this paper we will focus on a different problem, namely prepositional phrase attachment (PP attachment).

Prepositional phrase (PP) attachment is an important component of parsing. Consider the sentence Jessica saw the child with the kite. Humans use world knowledge to disambiguate the meaning of such sentences, and in this case would have no trouble figuring that with the kite describes the child and not the way Jessica saw him. In PP attachment, each instance of an ambiguous sentence can be represented using a number of features, either binary, e.g., does the noun phrase (NP) nested in the PP describe an animate object or multi-valued, e.g., what preposition is the head of the PP.

The majority of such classification methods are based on machine learning. A large number of the existing machine learning methods are based on vector representations. Each object is represented as a vector \( \mathbf{x} \) of features. The main assumption made by this type of algorithms is that a pair of objects \( \mathbf{x} \) and \( \mathbf{y} \) will be classified the same way if the distance between them in some space \( D \) is small (Zhu and Ghahramani, 2002a). In some cases, e.g., text classification using words as features, the space is Euclidean and one can use the Euclidean distance \( D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2} \) between two objects as a proxy for similarity. When dealing with nominal features (e.g., what preposition is the head of the PP), Euclidean distance is harder to define.

Recently, a number of graph-based methods have been proposed (Szummer and Jaakkola, 2001; Zhu and Ghahramani, 2002b; Zhu and Ghahramani, 2002a; Toutanova et al., 2004). In a weighted graph, objects (nodes) that have some similar properties are connected via a link. The weight of the link is typically set as follows:

\[
    w_{i,j} = \exp(-D(i, j)/\sigma)
\]

(Szummer and Jaakkola, 2001), but it can be defined in many other ways as well. In general, graph-based methods for classification work as follows: the probability that an unlabeled node is of the same type as a labeled node is inversely related to the cost of the path between the two nodes.

The method that we introduce falls into the category of graph-based learning. If no labeled data is given, they work as unsupervised clustering algorithms which can be used for document ranking (Page et al., 1998; Kleinberg, 1999) or summarization (Erkan and Radev, 2004). If a small amount of labeled data is available relative to the amount of unlabeled data, this method will perform transductive learning.

2 Background and Related Work

Random walk machine learning methods have been tested on natural language problems such as prepositional phrase attachment (Toutanova et al., 2004), word sense disambiguation (Mihalcea and Tarau, 2004) and text summarization (Erkan and Radev, 2004). In (Erkan and Radev, 2004),
the authors developed LexRank, a random walk based ranking method for extractive summarization. A graph based on lexical similarity is induced from the input text and LexRank determines the most central sentences using a random walk on the graph.

2.1 Formal Definitions of Random Walk

A graph is a pair $G = (V, E)$ in which $V$ is a set of $n$ vertices (nodes) and $E$ is a set of $m$ edges (links). The adjacency matrix of such a graph is an $n \times n$ matrix, $A$, where an entry $A(i,j) = w_{ij}$ indicates the weight of the edge between nodes $i$ and $j$. We are interested in symmetric and non-negative adjacency matrices, that is for all $i, j$ $A(i,j) = A(j,i) \geq 0$.

A Markov random walk on the graph is defined as follows: a token starts at a random node $n_0$ and at each step $k$, it moves from node $n_{k-1}$ to node $n_k$ with probability $p_{n_{k-1}, n_k}$. The matrix of transition probabilities $K$, where $K(i,j) = p_{i,j}$, is called the stochastic kernel of the Markov random walk. If $G$ is undirected, which is the case in all graph-based learning algorithms, $K$ is symmetric since $p_{i,j} = p_{j,i}$. Note that $K^n(i,j)$ indicates the probability that a random walk ends at node $j$ given that it originated at node $i$. It can be shown that if the stochastic matrix $K$ is irreducible ($G$ is connected) and aperiodic (not bipartite), all the rows of $K^n$ converge to the same unique stationary distribution, $\pi$, as $n$ goes to infinity. In other words, an element $\pi_i$ of the stationary distribution indicates the probability of any random walk ending at node $i$ in an infinite number of steps. Equivalently, $\pi_i$ indicates the fraction of the time spent at node $i$ by any random walk in the long run. Thus, $\pi$ gives an implicit ranking of the importance of the nodes in a graph. Many algorithms such as PageRank (Page et al., 1998) and HITS (Kleinberg, 1999) are based on this principle to identify important web pages.

More about these definitions can be found in (Lovász, 1993) and in many standard texts on linear algebra and stochastic processes.

2.2 Previous Methods Based on Random Walks

Szummer and Jaakkola (Szummer and Jaakkola, 2001) perform a Markov random walk on a graph as just described, starting from the labeled examples. The class label of each unlabeled instance $k$ is “voted” by all data points (labeled and unlabeled), with the votes weighted by the node’s likelihood of being the start of the random walk that ends at $k$ in $t$ steps. A data point $i$ votes for others’ label using its own distribution over the set of all class labels $Y$, i.e. $P(y|i)$, which is not available a priori. The authors hence suggest two methods to estimate these parameters with the objective to maximize the resultant likelihood or margin of the labeled data.

Zhu et al. (Zhu and Ghahramani, 2002b) propose a similar use of random walks to overcome the labeled data scarcity in classification problems. In their framework, the labeled data serve as the sources that “push out” class information, and the manifold structure of the quantitatively abundant unlabeled data is exploited to propagate the labels in a desirable way through the graph.

Other graph-based learning methods include graph min cuts (Kernighan and Lin, 1970; Pang and Lee, 2004; Blum and Chawla, 2001; Flake et al., 2002; Blum et al., 2004) and spectral partitioning (Meila and Shi, 2000; Dhillon, 2001; Ng et al., 2001; Zha et al., 2001; Zha, 2002; Kamvar et al., 2003; Bansal et al., 2004; Drineas et al., 2004).

2.3 Weakly-Supervised Learning

One of the earliest papers on bootstrapping for NLP problems, by Yarowsky (Yarowsky, 1995), presents an unsupervised learning algorithm for word sense disambiguation that, when trained on unannotated English text, rivals the performance of supervised techniques that require time-consuming hand annotations. The algorithm is based on two powerful constraints – that words tend to have one sense per discourse and one sense per collocation – exploited in an iterative bootstrapping procedure. Tested accuracy exceeds 96%.

Blum and Mitchell (1998) introduce co-training for the problem of classifying Web pages based on two views including different types of features (one based on the words in the pages themselves and another on the words on the hyperlinks of the pages pointing to them). Their main contribution is to show how two views can iteratively train each other to label a set of data.

Collins and Singer (1999) discuss the use of unlabeled examples for the problem of named entity classification. They develop a technique that uses only 7 manually labeled “seed” examples to classify entities into three classes plus “other”. Their approach works because given a particular
instance to classify, many features correlate with any particular class and one can thus iteratively augment the set of features associated with a given class. Two algorithms are presented. The first method uses a decision list algorithm similar to that of (Yarowsky, 1995), with modifications motivated by (Blum and Mitchell, 1998). The second algorithm extends ideas from boosting algorithms, designed for supervised learning tasks, to the framework suggested by (Blum and Mitchell, 1998).

Recent theoretical results can be found in (Abney, 2002), who refines the analysis of co-training, defines and evaluates a new co-training algorithm that gives a theoretical justification for the Yarowsky algorithm (Yarowsky, 1995), and shows that co-training and the Yarowsky algorithm are based on different independence assumptions.

Nigam et al. (Nigam et al., 2000) show that the accuracy of text classifiers can be improved by adding large collections of unlabeled documents. The authors use an algorithm based on a combination of Expectation Maximization (EM) and Naive Bayes. The initial classifier is trained on labeled data and then used to label the unlabeled set. The next classifier uses the large pool as part of its training process. After some iterations, the algorithm converges. Some of the improvements on this basic algorithm, proposed by Nigam et al., include adding a weighting factor to determine the contribution of the unlabeled data and the use of multiple mixture components for each class. The paper reports a reduction in classification error on three real-world tasks by up to 30%. Other papers related to this are (Shahshahani and Landrege, 1994; Zhang and Oles, 2000; Nigam and Ghani, 2000; Goldman and Zhou, 2000; Seeger, 2000; Chawla and Karakoulas, 2005) as well as the collections of papers in (Kremer and Stacey, 2001) and (Ghani et al., 2003).

Collins and Singer (Collins and Singer, 1999) also investigate EM as a weakly supervised learning algorithm for the application of named entity classification. The performance was shown to be not as good as the Co-boosting algorithm proposed by the authors.

3 TUMBL: Tripartite Update Method for Biased Learning

TUMBL (Tripartite Update Method for Biased Learning) is a transductive graph-based learning method that propagates the class distribution of each node in a random walk. What makes it particularly relevant for classification problems in NLP is its use of nominal features as a separate mode in a bipartite graph.

To start with, we form a graph where all labeled data instances, unlabeled data instances and feature values are represented as nodes. Each possible value of a nominal feature is represented as one node in the graph. An edge between a data node and a feature node exists if the feature value on one end of the edge holds for the data instance on the other end. Edge weights reflect the importance of the feature value incident to the edge.

Let \( L \) be the set of labeled data points, \( U \) be the set of unlabeled data points, and \( F \) be the set of unique feature values. TUMBL propagates class distributions of graph nodes through random walk. Define \( X \) as an \( |L| \times c \) matrix, where \( c \) is the number of distinct classes in the classification problem and each row of \( X \) represents the class distribution of a labeled instance in \( L \). Therefore, for training instance \( i \) belonging to the class \( j \), we set \( X(i,j) = 1 \) and \( X(i,k) = 0 \) for all \( k \neq j \). In the same fashion, we define \( Y \) as the class distribution matrix for the feature objects \( F \), and \( Z \) for the unlabeled data set \( U \). For each feature node, the class distribution shows how indicative that feature value is for different classes. \( Y \) and \( Z \) change over time as information gets transferred from the labeled set. We initialize \( Y \) and \( Z \) by setting all entries to \( 1/c \); that is, all features and unlabeled nodes are considered neutral at the beginning. Edges between the set of nodes \( L \) and \( F \) are defined by the \( |F| \times |L| \) transition matrix \( T_1 \), where \( T_1(i,j) \) is non-zero iff feature \( i \) holds on labeled instance \( j \). Similarly, we define the \( |F| \times |U| \) matrix \( T_2 \) as the transition matrix between \( F \) and \( U \). A non-uniform distribution on edge weight can be imposed which indicates that some features can be more or less important than the others depending on the specific task. However, we leave this issue as a future work and assume that edge weights are uniform for all edges defined in \( T_1 \) and \( T_2 \). TUMBL iterates over a sequence of three distinct random walk steps.

1. Walk from \( L \) to \( F \) based on \( T_1 \). The transition model \( T_1 \) is an \( |F| \times |L| \) matrix where \( T_{1ij} \) is non-zero iff feature \( i \) holds on labeled instance \( j \). As remarked earlier, the value of \( T_{1ij} \) shows the indicativeness of feature \( i \),
which is task-specific. This step updates the polarity (positive or negative) of the features using information from the labeled set.

2. Walk from $F$ to $U$ based on $T_2$. The transition model $T_2$ is an $|F| \times |U|$ matrix where $T_{2 \alpha \beta}$ is non-zero if feature $\alpha$ holds on unlabeled instance $\beta$. This step updates the class distributions of the unlabeled points according to their relationship with the features.

3. Walk from $U$ to $F$ based on $T_2$. The goal of this backward step is to update the feature polarity using information learned from the unlabeled data.

At the end of each step, we row-normalize the class distribution of the feature nodes or the unlabeled data nodes, whichever is updated. Since none of the steps walks to the labeled set, the class distribution of labeled instances are always intact. This way, they serve as information sources that “push out” class information ((Zhu and Ghahramani, 2002a)) to other parts of the graph. Figure 1 has TUMBL in pseudocode.

The number of iterations can be a parameter to the TUMBL algorithm. We also introduce a damping factor in the updating process. Initially set to 1, the damping factor gets multiplied by some $d_0$ ($0 < d_0 < 1$) after each iteration. Damping factor enables us to give more importance to the feature nodes that are closer to an unlabeled data node. As a side but rather important effect, the damping factor guarantees the convergence of the class distributions. Unlike PageRank, a random jump is not allowed – all transitions must follow a link.

After the termination of the random walk, the resultant class distribution of each unlabeled instance determines its label. Recall that $Z$ keeps track of the class distribution of the unlabeled points. For each row, we pick the largest entry to classify the corresponding instance to the class represented by that column. For example, a row of $(0.24, 0.76)$ in the final $Z$ would imply an assignment of negative class to the data point. The value of the cutoff can be determined from prior knowledge or from held out data. The default is to use 0.5, i.e. larger of the two entries.

The explicit representation of feature values as nodes is what makes TUMBL different than other graph-based classification algorithms mentioned earlier, where features are not represented as nodes in the graph, but rather contribute to the similarity metric defined among the data points. For instance, in case of the popular similarity metric based on Euclidean distance, each feature contribute the same amount to the similarity. In TUMBL, the importance of each feature value may be different from others and is induced from the labeled and unlabeled instances.

\[
\text{TUMBL}(n, d_0) \\
// n: number of iterations; d_0: damping factor \\
\text{Initialize class distribution matrices } X, Y_0, \text{and } Z_0. \\
d = 1 \\
\text{for } t = 1 : n \\
Y_t = d \times T_2^T X + Y_{t-1} \\
\text{Row-normalize } Y_t \\
Z_t = d \times T_2 Y_t + Z_{t-1} \\
\text{Row-normalize } Z_t \\
Y_1 = d \times T_2^T Z_1 + Y_t \\
\text{Row-normalize } Y_1 \\
\text{d = d * d}_0 \\
\text{endfor} \\
\text{for } i = 1 : |F| \\
\text{if } Z_n(i, 1) > Z_n(i, 2) \text{ then} \\
\text{Assign positive to instance} \\
\text{elseif } Z_n(i, 1) < Z_n(i, 2) \text{ then} \\
\text{Assign negative to instance} \\
\text{else} \\
\text{Assign more probable class to instance} \\
\text{endif} \\
\text{endif} \\
\text{endfor} \\
\]

Figure 1: TUMBL Algorithm.

To sum up, this algorithm propagates the class distribution information first from the labeled data to the features, capturing which feature values are more indicative of which classes. Such information is further transferred to the unlabeled set. The backward steps update the feature class distributions with information learned from the structure of the unlabeled data. This process is repeated with a damping factor to discount later rounds thus the effect of distant nodes on each other.

Two examples of TUMBL in action are shown in Figures 2 and 3. In each of these cases, the left subfigure shows the layout of the graph at the beginning of the random walk. Circles are used to represent the object to be classified and squares are the binary features. At $t = 0$, the labeled examples are either black or white depending on their class and the unlabeled examples and the features are set to a color half way between white and black. As the random walk progresses, information is passed from nodes to their neighbors (except that the values of the initially labeled objects never change), until the Markov process converges to a stationary distribution. When it converges, nodes are classified based on whether their intensity is above or
below a threshold. The first example shows how TUMBL can correctly label two non-connected regions. The second one is from a real Prepositional Phrase Attachment problem. The features are the tuples of preposition, verb, and two nouns associated with each ambiguous PP.

4 Experimental Data Set

We evaluate the performance of TUMBL on the prepositional phrase attachment (PP-attachment) data set as explained in (Collins and Brooks, 1995). PP-attachment is a well-studied binary classification task in the literature. The corpus is provided by IBM which contains 4-tuples (v n1 p n2) drawn from the Wall Street Journal Treebank ((Marcus et al., 1993)). The task is to determine whether the preposition in each quadruple is syntactically attached to the head verb or to the first head noun. For example, in the tuple (join, board, as, director), as attaches to the verb join. The prior class distribution is 59.0% for noun attachment (positive) and 41.0% for verb attachment (negative).

There are 20801, 4039, and 3097 tuples in the training, devtest and test sets, respectively. Collins and Brooks (Collins and Brooks, 1995) reported high accuracy on classifying the prepositional phrase attachment data set using a simple backed-off model. The model decides the label (noun attachment versus verb attachment) for each quadruple verb phrase (V N1 P N2) by using the occurrence of the quadruple in the training set. In case the quadruple under consideration has zero count in training (which is known to be most likely), the model suggests looking at the attachment distribution of low-dimensional subtuples that include the preposition p. 84.5% accuracy is achieved on the standard Wall Street Journal data set for this problem, which is fairly close to expert human judgment.

As a further improvement of the Collins and Brooks backed-off model, Toutanova et al. (Toutanova et al., 2004) use a Markov chain representation of the problem, and utilizes various sources of external knowledge to create more links besides those induced by the training data. These augmentations are proven to be effective in capturing higher order inter-word relationship otherwise not observed due to data sparseness. Random walk is carried out on the augmented graph to calculate the final attachment probability, where the parameters involved in the random walks are automatically learned with the objective function to optimize conditional log-likelihood of the correct attachment including quadratic regularization. The paper claims a boost in the classification accuracy on the same data set used by (Collins and Brooks, 1995).

5 Experiments and Results

We believe graph-based algorithms are particularly suitable for this task since shared words among tuples would allow us to propagate information from labeled to unlabeled tuples. Our aim is to identify words or phrases that trigger a certain attachment (to the verb or the noun) with high probability, and then use them to classify unlabeled tuples.

The construction of features for TUMBL is done as a preprocessing step. Each quadruple (V NP1 P NP2) is associated with exactly 15 feature values. They are, the quadruple itself, the 4 triples
composed by the elements ((V NP1 P), (V NP1 NP2), (V P NP2), (NP1 P NP2)), the 6 pairs ((V NP1), (V P), ⋮, (P NP2)), and the 4 individual tokens. Therefore, each tuple instance is connected to a maximum of 15 feature nodes in the graph. To account for morphological variations, we stem all words and convert year numbers (e.g. 1996), other numbers (e.g. 50), and entity names (e.g. IBM) into general tokens (e.g. YEARNUM, OTHERNUM, and ENTITYNAME) before creating the features.

For an initial comparison of TUMBL against other supervised or semi-supervised learning algorithms, we use the devtest data. (Collins and Brooks, 1995) shows the upper bound of the performance is roughly 88.2% by human annotation while the lower bound is around 72.2% by assigning the more likely attachment for each preposition using the entire training set of size 20801. The performance of the state-of-the-art (supervised) algorithms for this task is close to the upper bound when the entire training set is used. Also, the advantage of using unlabeled data is not obvious when there is plenty of training data. For these reasons, we are more interested in the performance of our graph-based algorithms when there is very limited training data and no external knowledge sources.

Table 1 shows the performance comparison of various algorithms using the unlabeled devtest data. For popular supervised learning algorithms, we use implementations provided by the Waikato Environment for Knowledge Analysis (WEKA) Package (Witten and Frank, 2000). We also implemented the algorithm presented by (Collins and Brooks, 1995) that is based on their Backed Off Model and their baseline method that assigns the most probable class to each preposition. We abbreviate the two as Backed Off and Majority Class, respectively. Interestingly, we observe that Majority Class algorithm reaches an accuracy of close to 72% rather quickly, a phenomenon that proves the entire data set is nicely distributed such that class distribution for each preposition is consistent in different parts of the data set. The Backed Off algorithm, on the other hand, suffers from the scarcity of training data and its performance under the described conditions is way below the 85% cited by (Collins and Brooks, 1995) based on the entire training set. We also implemented the Label Propagation (LP) algorithm in (Zhu et al., 2003) to compare our algorithms this recent transductive algorithm.

TUMBL and kNN (with k = 5) achieve the best results in the table without being statistically significant from each other. TUMBL performs significantly better than other methods (except for kNN) especially when there are fewer labeled data available, which is the case we are more interested in. It is also worth mentioning that the low performance of Decision List and Decision Tree algorithms shows that the PPAttach problem can not be solved by simple rule systems.

<table>
<thead>
<tr>
<th>Tr. Size</th>
<th>KNN</th>
<th>DT</th>
<th>J48</th>
<th>SVM</th>
<th>MajClass</th>
<th>BackedOff</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.696</td>
<td>0.582</td>
<td>0.637</td>
<td>0.679</td>
<td>0.644</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.701</td>
<td>0.640</td>
<td>0.680</td>
<td>0.660</td>
<td>0.677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.716</td>
<td>0.661</td>
<td>0.672</td>
<td>0.693</td>
<td>0.711</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>0.717</td>
<td>0.681</td>
<td>0.680</td>
<td>0.704</td>
<td>0.721</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.720</td>
<td>0.697</td>
<td>0.695</td>
<td>0.710</td>
<td>0.724</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>0.729</td>
<td>0.704</td>
<td>0.700</td>
<td>0.716</td>
<td>0.734</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>0.734</td>
<td>0.714</td>
<td>0.709</td>
<td>0.725</td>
<td>0.740</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>0.740</td>
<td>0.725</td>
<td>0.709</td>
<td>0.733</td>
<td>0.745</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>0.743</td>
<td>0.733</td>
<td>0.715</td>
<td>0.736</td>
<td>0.746</td>
<td></td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>0.749</td>
<td>0.737</td>
<td>0.714</td>
<td>0.742</td>
<td>0.746</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>0.749</td>
<td>0.747</td>
<td>0.719</td>
<td>0.742</td>
<td>0.747</td>
<td></td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>0.751</td>
<td>0.749</td>
<td>0.719</td>
<td>0.750</td>
<td>0.752</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>0.755</td>
<td>0.752</td>
<td>0.718</td>
<td>0.749</td>
<td>0.751</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Algorithm performance on PPAttach devtest data. Numbers are averages over 30 runs. Tr. Size: Training data size (that is, size of the labeled component of the training data). TUMBL: Tripartite Update Method for Biased Learning; BackedOff: Backed Off Model; MajClass: Majority Class; NB: Naive Bayes; SVM: Support Vector Machine; KNN: K-Nearest Neighbors; DT: Decision Table; J48: Decision List; DL: Decision Tree; LP: Label Propagation of (Zhu et al., 2003)

Since TUMBL has the best performance in the previous comparison and because it is nicely scalable to larger data, we tried it with more training data and measured its performance on the entire data set of PPAttach, comparing against Backed Off and Majority Class to see if it is competitive even when there is plenty of labeled examples. Table 2 and Table 3 have the results on the devtest and test data, respectively. The tables show that TUMBL is still comparable to the Backed Off algorithm when provided with abundant training data, and rivals the Majority Class method by a large margin.

6 Conclusion and Future Work

We have presented a promising random walk based machine learning algorithm. The advantages of TUMBL are the following: (1) it allows
for data to be represented in a natural way, in terms of similarities between instances, (2) it is designed to capture lower dimensionality structure in large dimensional problems, (3) it is based on solid mathematical foundations, (4) it is language independent, (5) it relates strongly to other known methods such as singular value decomposition, spectral partitioning, heat propagation, voltage change in electric circuits, and etc., (6) it can be combined with other methods in machine learning such as active learning, boosting and bagging, (7) it has applications to a very large number of problems in Natural Language Processing, (8) it can be extended to other types of problems outside NLP.

In future work, we will address a number of important research questions and practical problems. An interesting problem for graph-based algorithms is the assessment of how informative a node is. We will investigate active learning methods that determine the most important data nodes to label based on the topology of the graph: e.g. unlabeled nodes with high geodesic or betweenness centrality (Freeman, 1977).

Not all potential features available to a learning method are potentially useful. We will investigate novel methods for identifying correlations between features based on random walks, the intuition being that two features may not connect to the same objects directly but can be connected instead via longer paths, that is, nodes that are connected to a number of other unlabeled nodes via a short distance are more likely to be informative. We will investigate active learning algorithms that utilize the topology of the graph.

It is known from the literature that small-world graphs (graphs with high clustering coefficient and short characteristic path (Watts and Strogatz, 1998)) may have some important search properties (Adamic et al., 2001; Watts et al., 2002; Adamic et al., 2002; Sarshar et al., 2004) which make them locally navigable. We will investigate how the clustering coefficient, characteristic path, degree distribution, and overall topology of the graph affects the performance of TUMBL.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>TUMBL</th>
<th>BackedOff</th>
<th>MajClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.770</td>
<td>0.777</td>
<td>0.730</td>
</tr>
<tr>
<td>2000</td>
<td>0.789</td>
<td>0.796</td>
<td>0.734</td>
</tr>
<tr>
<td>3000</td>
<td>0.797</td>
<td>0.805</td>
<td>0.739</td>
</tr>
<tr>
<td>5000</td>
<td>0.808</td>
<td>0.817</td>
<td>0.742</td>
</tr>
<tr>
<td>7500</td>
<td>0.815</td>
<td>0.824</td>
<td>0.742</td>
</tr>
<tr>
<td>10000</td>
<td>0.821</td>
<td>0.830</td>
<td>0.743</td>
</tr>
<tr>
<td>12500</td>
<td>0.827</td>
<td>0.836</td>
<td>0.743</td>
</tr>
<tr>
<td>15000</td>
<td>0.830</td>
<td>0.835</td>
<td>0.743</td>
</tr>
<tr>
<td>17500</td>
<td>0.833</td>
<td>0.840</td>
<td>0.742</td>
</tr>
<tr>
<td>20801</td>
<td>0.837</td>
<td>0.843</td>
<td>0.742</td>
</tr>
</tbody>
</table>

Table 2: Algorithm performance on PPAttach devtest data. TUMBL: Tripartite Update Method for Biased Learning; BackedOff: Backed Off Model; MajClass: Majority Class.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>TUMBL</th>
<th>BackedOff</th>
<th>MajClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.774</td>
<td>0.776</td>
<td>0.717</td>
</tr>
<tr>
<td>2000</td>
<td>0.790</td>
<td>0.798</td>
<td>0.720</td>
</tr>
<tr>
<td>3000</td>
<td>0.801</td>
<td>0.806</td>
<td>0.720</td>
</tr>
<tr>
<td>5000</td>
<td>0.814</td>
<td>0.820</td>
<td>0.721</td>
</tr>
<tr>
<td>7500</td>
<td>0.824</td>
<td>0.828</td>
<td>0.721</td>
</tr>
<tr>
<td>10000</td>
<td>0.829</td>
<td>0.833</td>
<td>0.721</td>
</tr>
<tr>
<td>12500</td>
<td>0.834</td>
<td>0.836</td>
<td>0.721</td>
</tr>
<tr>
<td>15000</td>
<td>0.838</td>
<td>0.838</td>
<td>0.722</td>
</tr>
<tr>
<td>17500</td>
<td>0.839</td>
<td>0.841</td>
<td>0.722</td>
</tr>
<tr>
<td>20801</td>
<td>0.842</td>
<td>0.842</td>
<td>0.722</td>
</tr>
</tbody>
</table>

Table 3: Algorithm performance on PPAttach test data. TUMBL: Tripartite Update Method for Biased Learning; BackedOff: Backed-Off Model; MajClass: Majority Class.

References


Tong Zhang and Frank J. Oles. 2000. A probability analysis on the value of unlabeled data for classification problems. In ICML ’00, pages 1191–1198, Stanford, California, USA.

