On Supervised Mining of Dynamic Content-Based Networks[†]

Charu C. Aggarwal[1]* and Nan Li[2]

[1]*IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA*

[2]*Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA 93106, USA*

Received 16 June 2011; revised 9 September 2011; accepted 5 October 2011
DOI:10.1002/sam.10140
Published online 31 January 2012 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: In recent years, a large amount of information has become available online in the form of web documents, social networks, or blogs. Such networks are large, heterogeneous, and often contain a huge number of links. This linkage structure encodes rich structural information about the topical behavior of the network. Such networks are often *dynamic* and evolve rapidly over time. Much of the work in the literature has focused on classification either with purely text behavior or with purely linkage behavior. Furthermore, the work in the literature is mostly designed for static networks. However, a given network may be quite diverse, and the use of either content or structure could be more or less effective in different parts of the network. In this paper, we examine the problem of node classification in *dynamic* information networks with both text content and links. Our techniques use a random walk approach in conjunction with the content of the network to facilitate an effective classification process. Our approach is dynamic, and can be applied to networks which are updated incrementally. Our results suggest that an approach based on both content and links is extremely robust and effective. We also present methods to perform supervised keyword-based clustering of nodes using this approach. We present experimental results illustrating the effectiveness and efficiency of our classification approach. We also show that the approach is able to find effective and coherent clusters. © 2012 Wiley Periodicals, Inc. Statistical Analysis and Data Mining 5: 16–34, 2012

Keywords: graph classification; structural classification; label propagation

1. INTRODUCTION

In recent years, there has been an explosion of text content on the web in a variety of forms. In addition to the standard forms of content such as web pages, an enormous amount of content may be found in the form of blogs, wikis, and other forms of social media. Such networks are often a challenge for mining algorithms because they often contain both structure and content. Some examples of such networks include author citation networks, coauthorship networks, product databases with large amounts of text content, and so on. Such networks are *highly dynamic* and may be frequently updated over time. For example, new nodes may constantly be created in the network, as new postings are created in a blog network; similarly old nodes may be deleted, as old postings are deleted. As a result the structure of the network may be quite dynamic, and may vary over time. In the most general case, we model our problem as a graph of nodes, each of which may contain text content.

A key problem which often arises in these domains is that of node classification [2,3]. The classification problem arises in the context of many network scenarios in which the underlying nodes are associated with content. In the node classification problem, it is assumed that a subset of the nodes in the network may be labeled. It is desirable to use these labeled nodes in conjunction with the structure and content for the classification of nodes which are not currently labeled. For example, many blogs or other network documents may naturally belong to specific topics on the basis of their content and linkage patterns. However, most such documents may not be formally associated with labels in social networking scenarios because of a lack of resources available for a human-centered labeling process. In this paper, we will address the classification problem, in which it is desirable to determine the categories of the unlabeled nodes in an automated way with the use of both

Correspondence to: Charu C. Aggarwal (charu@us.ibm.com)
[†] This paper is an extended version of Ref. [1].

structure and content of the network. The presence of labels on a subset of the nodes provides the implicit training data which can be leveraged for learning purposes.

The node classification problem is particularly challenging in the context of very large, dynamic, and evolving social and information networks. In particular, a number of natural desiderata are applicable in the design of classification algorithms in this scenario. These desiderata are as follows:

- Social and information networks are very large, as a result of which link classification algorithms need to be *efficient*. This can be particularly challenging, if we intend to use both text and links during the classification process. The addition of text content to the linkage information is responsible for a considerable increase in the size of the underlying network representation.

- Many such networks are dynamic, and are frequently updated over time. As the structure of the network may constantly change over time, the underlying classification model may also change. Therefore, the model needs to be efficiently updatable in *real time* in order to account for such changes. Such a dynamic model also needs to be easy to use, so that the end-process of classification can be achieved without too much overhead.

- Such networks are often noisy, as many of the links and content features may not be relevant to the classification process. In addition, different portions of the network may be better suited to different kinds of classification models. For example, some portions of the network may be better classified with structure, whereas other portions may be better classified with content. We need to design a classifier, which can make such decisions in a seamless way, so that the appropriate parts of the network may be used most effectively for the classification process.

The problem of classification is widely studied in the data mining community [4]. The problem has been studied in the context of both structural [5–7] and content-based [8–11] analysis. Two natural choices can be used for classification of content-rich networks:

- The most straightforward approach is to directly use text classifiers in order to perform the classification. A variety of text classifiers are available for this purpose. A detailed evaluation of techniques for text categorization may be found in Refs [10,11]. However, such an approach ignores the rich structural information which is often available in the context of a network.

- A second way to perform the classification is by using the information which is latent in the underlying link structure. For example, effective methods have been proposed in Ref. [7] in order to perform link-based classification. Similar techniques have been used in Ref. [5] in order to label blogs for classification. However, these methods fail to leverage the information which is available in the underlying content for classification purposes.

It is clear that both text and links encode important information about the underlying network. Furthermore, these provide different views of the underlying information. For example, the text provides ideas about content, whereas the linkage behavior provides information about interconnectedness between different kinds of nodes, some of which may be used for classification. The latter is especially the case, when the content in a given node is limited and the linkage information provides an idea of the relationships of the test node with other labeled nodes. On the other hand, the content can be used to glean better classification insights when the linkage structure is either sparse or not informative enough to provide information about the classification behavior of the underlying node. Therefore, it makes sense to examine whether it is possible to *combine* text and linkage behavior in order to perform robust classification, which works more effectively in a generic scenario. Furthermore, such an integration must be seamless, in that it should be able to automatically use the most effective strategy in a given scenario. This paper will propose a random walk approach, which combines text and linkage behavior, and show that it can be used in a seamless way in order to perform more robust classification. We will refer to this algorithm as DyCOS, which corresponds to the fact that it is a *DY*namic *C*lassification algorithm with c*O*ntent and *S*tructure. Furthermore, the approach is *dynamic*, and *scalable to large networks*, as it can be applied to large, and rapidly updatable networks.

We also show that the technique can be used for supervised keyword-based clustering of nodes in a social network. A common application which often arises in social networks is to create clusters (or communities) of nodes which are related to specific keywords. As the text in such social networks is quite sparse, it may often be the case that such clusters cannot be effectively determined with the use of pure keyword-based analysis, because many relevant nodes may not directly contain the keyword itself. As our approach uses the link structure in addition to the content, it is able to create much more robust communities even when the text on the nodes is noisy and sparse. Clusters are created effectively from the different nodes with the use of the propagation paradigm introduced in DyCOS. While a number of unsupervised techniques for combining structure and content have been proposed in past works, they cannot be used for supervised keyword-based clustering. In many applications, such supervised clusters may be more useful, especially when we have a rough idea of the types of communities that need to be found in terms of keywords.

This paper is organized as follows. We discuss related works in the remainder of this section. In Section 2, we introduce a dynamic random-walk model for classification with text and links. Section 3 shows how DyCOS algorithm leverages this model for the classification. We discuss an application of this technique to the problem of supervised clustering in Section 4. The experimental results are presented in Section 5. Section 6 presents the conclusions.

1.1. Related Work

The problem of text classification [8–11] has been studied widely in the information retrieval literature. Detailed surveys may be found in Refs [10,11]. In the context of the web and social networks, text classification poses a significant challenge, because the text is often drawn from heterogeneous and noisy sources which are often hard to model with a standardized lexicon. Some of the earliest work on the use of linkage techniques to enhance classification may be found in Ref. [12]. This work uses the *text content* in adjacent web pages in order to model the classification behavior of a web page. However, it is not focussed on the problem of node-classification in a partially labeled graph of documents.

The problem of node classification has also been studied in the graph mining literature, and especially relational data in the context of *label or belief propagation* [13–15]. Such propagation techniques are also used as a tool for semisupervised learning with both labeled and unlabeled examples [16]. A technique has been proposed in Ref. [7], which uses link-based similarity for node-classification. Recently, this technique has also been used in the context of blogs [5]. However, all of these techniques use link-based methods only. Some recent work has been done on the clustering problem with content and links [17]. Another work [6] discusses the problem of label acquisition in the context of collective classification. Label acquisition is an important problem, because it is required in order to provide the base data necessary for classification purposes. A method to perform *collective classification* of email speech acts has been proposed in [18]. It has been shown that the analysis of relational aspects of emails (such as emails in a particular thread) significantly improves the classification accuracy. It has also been shown in Refs [12,19] that the use of graph structures during categorization improves the classification accuracy of web pages. While these methods provide a limited application of structural information, they are not designed to work for dynamic networks which may constantly evolve over time. This paper provides a first approach to the problem of *efficient and dynamic* node classification in a *massive labeled* network, where both text and node labels are available for classification purposes. Much of the work proposed recently is not applicable to the case of *massive information networks in a dynamic scenario* because of scalability issues. We will use carefully designed summary structures which can efficiently perform such a classification. We will show that the use of both sources in the classification process provides an effective classification technique in such massive networks. Furthermore, we design our technique to work effective for *massive and dynamic networks* which may constantly evolve over time. We will show that our method will show considerable improvements over other existing methods. For the case of the clustering problem, a number of techniques have been designed for unsupervised for combining content with structure [17], though these methods are not designed for text data, and they do not use supervision for the mining process.

2. NODE CLASSIFICATION MODEL WITH TEXT AND LINKS

We will first introduce some notations and definitions which are relevant to the node classification problem. We assume that we have a large network containing a set of nodes $\mathcal{N}_t$ at time t. As, the approach is dynamic, we use a time-subscripted notation $\mathcal{N}_t$ in order to denote the changing nodes in the network. A node in $\mathcal{N}_t$ corresponds to a structural node in the network along with the associated text content. We also assume that a subset $\mathcal{T}_t$ of these nodes $\mathcal{N}_t$ may be labeled. These nodes form the training nodes, and they contribute both linkage and text information for classification purposes. We assume that the nodes in $\mathcal{T}_t$ are labeled from a total of k classes, which are drawn from the set $\{1 \dots k\}$. As in the case of the node set $\mathcal{N}_t$, the set $\mathcal{T}_t$ is not static, but may dynamically change over time, as new labeled nodes may be added to the network. For example, either a new labeled node may be added to both $\mathcal{N}_t$ and $\mathcal{T}_t$ or an existing node in $\mathcal{N}_t$ may be initially unlabeled (and therefore not a part of the training data), but may eventually be labeled, when new training information is received. In the latter case, we add that node to $\mathcal{T}_t$. Similarly, the set of edges at time t is denoted by $\mathcal{A}_t$. Furthermore, new labels may be acquired for different nodes over time, as a result of which the set $\mathcal{T}_t$ may change as well. Clearly, this dynamic setting is extremely challenging, because it implies that the training model may change rapidly. The entire network is denoted by $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{A}_t, \mathcal{T}_t)$ at a given time t.

In order to achieve our goal, the DyCOS approach will construct a summary representation which is based on both text and link structure. In order to perform the classification, we will create a text-augmented representation of the network, which is leveraged for classification purposes. We will show how to implement this summary representation efficiently, so that it is possible to use it effectively in a network. Our broad approach is to construct an intuitive random walk based approach on the network, in which both text and links are used during the walk process for classification. The level of importance of text and links can be either controlled by a user or inferred in an automated way, as discussed below. As we intend to design classification techniques which use the underlying content, it is useful to first determine the words which are most discriminative for classification purposes. The ability to select out a compact classification vocabulary is also useful in reducing the complexity and size of the model at a later stage. The aim of picking only a small set of discriminative keywords is to guide the classification process with a small number of keywords for each node. For example, for a larger document such as a web page or blog, the number of possible words in the document may be as many as a few hundred or a thousand. However, the aim of using the feature selection process is to reduce it to a very small number.

The discriminative quantification of a given word from the corpus is performed with the use of a well known measure known as the *gini-index*. We dynamically maintain a *sample reservoir S_t* of *labeled documents* in the collection, and use them for the purposes of computing the gini-index. For this purpose, we can use the reservoir sampling algorithm discussed in Ref. [20]. From time to time, we compute the gini-indices in order to compute the discriminative power of the different words. The frequency of updating the gini-indices can be either equivalent to or less than the frequency the network is dynamically updated. For a given word w, let $p_1(w) \ldots p_k(w)$, be the relative *fractional presence* of the word w in the k classes. Therefore we have:

$$\sum_{i=1}^{k} p_i(w) = 1. \tag{1}$$

Let $n_1(w) \ldots n_k(w)$ be the number of pages in the k classes, in the sample S_t which contain the word w, we estimate $p_i(w)$ as follows:

$$p_i(w) = n_i(w) / \sum_{j=1}^{k} n_j(w). \tag{2}$$

Then, the gini-index $G(w)$ for the word w is computed as follows:

$$G(w) = \sum_{j=1}^{k} p_j(w)^2. \tag{3}$$

The value of $G(w)$ always lies in the range (0, 1). If the word is evenly distributed across the different classes, then the value of $G(w)$ is closer to 0. On the other hand, if the word w has a preponderance in one of the classes, then the value of $G(w)$ is closer to 1. Thus, words which have a higher value of $G(w)$ are more discriminative for classification purposes. As a first step, we pick a set $\mathcal{M}_t$ of the top m words which have the highest value of $G(w)$ and use them in order to construct our structural node classification model. The set $\mathcal{M}_t$ represents the *active vocabulary* which is useful for classification purposes. In our current implementation (Section 5), $\mathcal{M}_t$ is updated at the same pace as the dynamic network is updated. Nonetheless, we note that $\mathcal{M}_t$ does not need to be updated at each time instant t. Rather, it can be updated in batch at specific instants in time, with a much less frequency compared to that the network is updated. The discriminatory indices of the words are analyzed periodically, and the most discriminatory words are used for classification purposes. These discriminative words are used in order to create a new semibipartite representation of the network which is useful for classification purposes.

2.1. The Semibipartite Content–Structure Transformation

One of the goals of the DyCOS algorithm is to create a model which can deal with the content and links in a *seamless* way for the transformation process. For this purpose, both the content and the original links are transformed into a structural representation, which is referred to as the *semibipartite* content–link transformation. The set $\mathcal{M}_t$ provides a more compact vocabulary which is used in order to create a *semibipartite* content–link transformation. The semibipartite representation is a graph in which one partition of nodes is allowed to have edges either within the set, or to nodes in the other partition. The other partition is only allowed to have edges to the first, but it does not have any edges within the set. Therefore, it is referred to as *semibipartite*, as only one of the two node sets satisfies the bipartite property. The semibipartite content–link transformation defines two kinds of nodes: (i) The first kind are the *structural nodes* which are the same as the original node set $\mathcal{N}_t$. This set inherits edges from the original network. (ii) The second kind of nodes are the *word nodes* which are the same as the discriminative vocabulary $\mathcal{M}_t$.

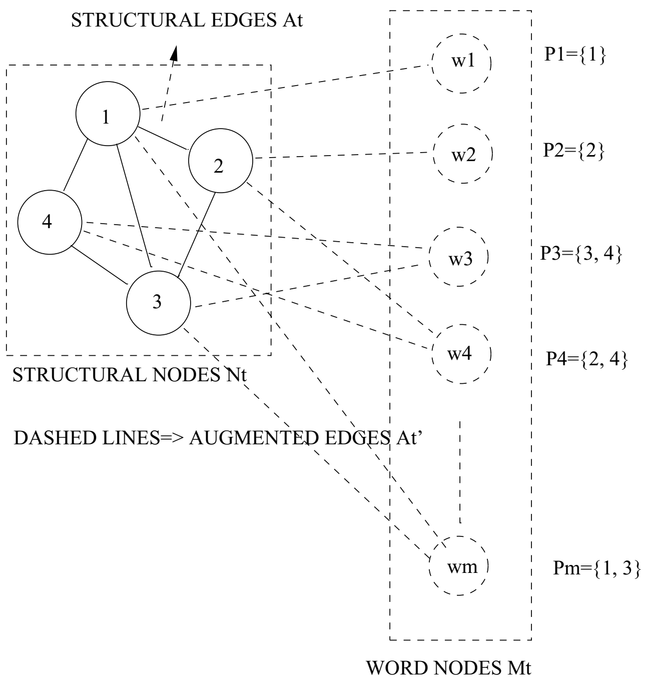


Fig. 1 The semibipartite transformation.

Then, we construct the semibipartite graph $F_t = (\mathcal{N}_t \cup \mathcal{M}_t, A_t \cup A'_t)$, in which $\mathcal{N}_t$ and $\mathcal{M}_t$ form the two sides of the bipartite partition. The set A_t is inherited from the original network, whereas A'_t is constructed on the basis of word-presence in the text in different network nodes. Specifically, an undirected edge exists between the information network node $i \in \mathcal{N}_t$, and the word node $j \in \mathcal{M}_t$, if the corresponding word is contained in the information node i. Thus, the edges in A_t are within a partition, whereas the edges in A'_t are across the partition. An example of this transformation is illustrated in Fig. 1. We have marked the nodes N_t, N'_t, A_t, and A'_t in this figure. The node set which corresponds to the structural nodes has edges which are indicated by solid lines, whereas the connections between structure and content nodes are illustrated by dashed lines. Thus, a walk from one node to another may use either solid or dashed lines. This provides a way to measure proximity both in terms of link and content. The idea of creating such a network is that *structural nodes are linked together based on proximity of contents*. Thus, if a pair of nodes share many words in common, there are an equivalent number of ways of reaching one node from the other in a random walk process. This ensures that the greater the similarity in content, the more biased the random walk process in visiting nodes with similar content in a single walk. Thus, by associating the class label with individual random walks, it is possible to propagate the classes on the labeled nodes to the unlabeled nodes. The ability to utilize such proximity in the context of a classification process helps us combine links and content in a seamless way for classification in terms of the structural proximity in the new transformed network. While it is possible to use n-grams rather than individual word nodes for the transformation, we choose to use individual words in order to maintain efficiency.

In addition, a number of data structures are required in order to allow efficient traversal of the text and linkage structure in our random-walk approach. These data structures are as follows: (i) For each of the word nodes $w \in \mathcal{M}_t$, we maintain an inverted list containing the set of node identifiers which contain the word corresponding to w. We assume that the set of nodes pointed to by word i is denoted by P_i. The set P_i is illustrated in Fig. 1. (ii) For each of the original set of nodes $\mathcal{N}_t$ in the network structure, we maintain an inverted list of words contained in the corresponding document. The set of words pointed to by node i is denoted by Q_i. (iii) For each node identifier, we maintain information about its class label, if the node is labeled. Otherwise, we simply maintain the meta-information that the node is not labeled.

We note that total size of all the inverted lists P_i for different values of i is at most equal to the text size of the collection, if it is represented in terms of only the discriminative words. Similarly, the total size of all the inverted lists Q_i for different values of i is at most equal to the discriminative text collection size. These inverted lists can be updated easily during addition or deletion of nodes to the collection. During addition or deletion of nodes, we need to either add to or delete from inverted lists P_i, such that word i is contained in the added or deleted node. We also need to add (delete) an inverted list Q_r corresponding to the newly added (removed) node r. We note that this incremental update process is extremely efficient, and can be dynamically performed for a data stream. From time to time, we may also want to adjust the word nodes, depending upon the change in discriminatory behavior. In such cases, we need to add or delete corresponding word nodes. The process of updating the inverted lists is similar to the previous case. The update process can also be efficiently applied to a node, when the content within a node changes. In this cases, the corresponding links between the structure and content nodes need to updated.

We note that the complexity of this procedure is essentially proportional to the sum of the number of links that are added to the content–structure transformation and the number of links which are deleted from the content–structure transformation (because some of the nodes no longer remain discriminative and are therefore dropped from $\mathcal{M}_t$). In addition, we need to recompute the gini-index for each node at each update which requires $O(|\mathcal{M}_t| \cdot k)$ time for a k-class problem. As the number of nodes which are added to $\mathcal{M}_t$ far exceeds the number of nodes which are deleted in applications in most practical applications in which the network size constantly increases over time, we will focus on only the added links for simplicity. Let n_s be the number of nodes added to the network since the last update, such that each node contains a total of at most n_r words. Then, at most $n_r \cdot n_s$ links at added. Therefore, the total time-complexity of updating the structure is approximately proportional to $O(n_r \cdot n_s + |\mathcal{M}_t| \cdot k)$.

3. CLASSIFICATION WITH TEXT AND LINK-BASED RANDOM WALKS

In this section, we describe the classification approach of DyCOS. The use of both content and links during the random walk process is critical in creating a system which provides effective classification. Since random walks can be used to define proximity in a variety of ways [21], a natural approach is to construct proximity-based classifiers which use the majority labels of random walk nodes for the propagation process. Since the text is included within the node structure of the semibipartite graph, it follows that a random walk on this graph would implicitly use both text and structural links during the classification process. The *starting node* in this random walk is the unlabeled node in $\mathcal{N}_t$ which needs to be classified. Of course, we would also like to have a way to control the relative impact of text and structural nodes during the classification process. We note that a straightforward use of a random walk over the semibipartite graph F_t may not be very effective, because the walk can get lost by the use of individual word nodes in the random walk. In order to be able to control this relative importance, we will define the walk *only over the structural nodes with implicit hops over word nodes*. Specifically, a step in the random walk can be one of two types:

(i) The step can be a *structural hop* from one node in $\mathcal{N}_t$ to another node in $\mathcal{N}_t$. This is a straightforward step from one node to the next with the use of a link in the original graph. If such a link does not exist, then the structural hop teleports to the starting node.

(ii) The step can be a *content-based multihop* from a node in $\mathcal{N}_t$ to another node in $\mathcal{N}_t$. This step uses the linkage structure between the structural and word nodes during the hop. Thus, each hop really uses an *aggregate analysis of* the *word-based* linkages between one structural node in $\mathcal{N}_t$ and another structural node in $\mathcal{N}_t$. The reason for this aggregate analytical multihop approach is to reduce the noise which naturally arises as a result of the use of straightforward walks over *individual word nodes* in order to move from one structural node to the other. This is because many of the words in a given document may not be directly related to the relevant class. Thus, a walk from one structural node to the other with the use of a single word node could diffuse the random walk to less relevant topics.

We will discuss more details about how this content-based multihop is computed slightly later. We use a statistical analysis of the nodes encountered during the random walk in order to perform the classification. A key aspect here is to be able to control the importance of structure and content during the hops. For this purpose, we use a *structure parameter* p_s. This parameter defines the probability that a particular hop is a structural hop rather than a content hop. When the values of p_s is set at 1, then it means that content is completely ignored during the classification process. On the other hand, when the value of p_s is set at 0, then it means that only content is used for classification. We will discuss more details about the classification process below.

3.1. Classification Process

The process of classification uses repeated random walks of length h starting at the source node. The random walk proceeds as follows. In each iteration, we assume that the probability of a structural hop is p_s. Otherwise, a content multihop is performed with probability $(1 - p_s)$. By varying the value of p_s, it is possible to control the relative importance of link and content in the classification process. While defining the length of a walk, a content-hop is defined as a single hop in the same way as a structural hop, even though a content walk is really performed using analysis of intermediate word nodes. A total of l such random walks are performed. Thus, a total of $l \cdot h$ nodes are visited in the random walk process. These nodes may either belong to a particular class, or they may not be labeled at all. The most frequently encountered class among these $l \cdot h$ nodes is reported as the class label. If no labeled node is encountered through all random walks (which is a very rare situation), DyCOS simply reports the most frequent label of all nodes currently in the network. This is specific to the current time stamp and does not depend on the particular source node. A high-level pseudo-code sketch of the classification algorithm is presented in Algorithm 1.

Next, we will discuss the efficient implementation of structural and content hops. This is done with the use of the inverted indices which are available at the different nodes of the graphs. At each node in the random walk process, we

Algorithm 1 DyCOS Classi- cation Process

```
Data: Network $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{A}_t, \mathcal{T}_t)$, number of random walks, $l$, walk length, $h$, structural
      hop probability, $p_s$
Result: Classification of $\mathcal{T}_t$, accuracy, $\theta$
1 for Each node $v$ in $\mathcal{T}_t$ do
2     for $i$ from 1 to $l$ do
3         Perform an $h$-hop random walk from $v$, with structural hop probability, $p_s$;
4     Classify $v$ with the class label most frequently encountered;
5 $\theta \leftarrow$ the percentage of nodes correctly classi- ed;
6 Return classification labels and $\theta$;
```

flip a coin with probability p_s. In the event of a success, we perform a structural hop; otherwise we perform a content hop. Structural hops are straightforward, because we only need to look up the adjacency list for that node, and perform the corresponding hop.

For the case of the content-based hops, a two-step approach is required. First, we need to determine the nodes with the top-n most frequent two-hop paths from a node in $\mathcal{N}_t$ to another node in $\mathcal{N}_t$ with the use of an intermediate word node. The first step is to determine all the nodes which are reachable in paths of length 2. Let the relative frequency of the number of two-hop paths which lead to these n nodes be denoted by $r_1 \ldots r_n$. Then, we sample the ith among these nodes with probability r_i. By truncating the random walk process to only the top-n nodes, we ensure that the random walk is not lost because of nontopical words in the documents. In order to actually perform the walk, we need to use the inverted lists at the nodes in $\mathcal{N}_t$ and $\mathcal{M}_t$. For each node in $\mathcal{N}_t$, we can determine the word nodes contained in it. Then, for each word node, we can determine the structural nodes which contain that word. This can again be achieved by using the inverted lists at the word nodes. The union of these lists is the set of nodes which can be reached in a content-walk of length 2. The top-n most frequent nodes among these are sampled for the purposes of determining the next node in the walk. We note that content hops are less efficient to perform than structural hops, but the use of inverted lists greatly speeds up the process.

3.2. Analysis

An important point to note is that we are essentially using Monte Carlo sampling of the paths from different nodes. The use of such a sample can result in some loss of accuracy, but the advantage is that it is much more efficient than the use of an exact computation of node probabilities. This is because the exact computation of node probabilities can require expensive matrix operations based on the structure of the graph adjacency matrix. This is not very helpful for a large network in which many such computations need to be performed. The sampling approach is also critical in being able to utilize the approach effectively in a dynamic scenario in which repeated recomputation of node probabilities is required. Therefore, an efficient sampling approach such as the one discussed in the paper is critical. In this section, we will study the loss of accuracy which arises from the use of such samples. The aim is to show that the use of Monte Carlo samples retains practically the same effectiveness as an approach which can determine the hop probabilities exactly. As mentioned earlier, the class which is visited the maximum number of times during the entire random walk process is reported as the relevant class. As in the previous discussion, we ignore word nodes in the analysis of hops, since they are only used as intermediate nodes during the content hops. Therefore, all hops are considered to be either structural hops from one node in $\mathcal{N}_t$ to another node in $\mathcal{N}_t$, or content hops from one node in $\mathcal{N}_t$ to another node in $\mathcal{N}_t$ with the use of an intermediate word node. The main focus is to show that the *ordering* of different classes in terms of the number of visits does not change significantly because of the sampling process. For this purpose, we will use the Hoeffding inequality. First, we will consider the case of two classes. Then, we will generalize our results to an arbitrary number of classes. Let us consider two classes 1 and 2, for which the *expected fraction of visits* for a particular test node are f_1 and f_2, respectively, so that $b = (f_1 - f_2) > 0$. In this case, class 1 is a more appropriate label for the test node as compared to class 2, because it is the majority class. We further note that the sum of f_1 and f_2 may not necessarily be 1, because many of the intermediate nodes in the hop may be unlabeled. We would like to determine the probability that the Monte Carlo sampling process results in the *undesirable outcome* of the ordering of the classes 1 and 2 being reversed during the random sampling process.

This directly provides us with the classification error probability.

LEMMA 1: Let us consider two classes with expected visit probabilities of f_1 and f_2, respectively, such that $f_1 - f_2 > 0$. Then, the probability that the class which is visited the most during the sampled random hop process is reversed to class 2, is given by at most $e^{-l \cdot b^2/2}$.

Proof: See Appendix. ■

The result above shows that the probability of error because of sampling reduces exponentially with the number of paths that are sampled. For example, consider the case, when we sample 100 different paths, and $b = 0.1$. In that case, the probability of error is given by at most $e^{-1000 \cdot 0.01/2} = e^{-5} < 1\%$. This suggests that the *additional error of approach because of the sampling process* will be less than 1%. In general, the exponential rate of error decrease with sample size is critical in ensuring that the approach can be used efficiently with Monte Carlo sampling. Next, we will generalize the result to the case of k classes. First, we need to define the concept of *b-accuracy* of a sampling process in a case with k classes.

DEFINITION 1: Let us consider the node classification problem with a total of k classes. We define the sampling process to be b-accurate, if none of the classes whose expected visit probability is less than b of the class with the largest expected visit probability turns out have the largest sampled visit probability.

We note that the above definition is simply a generalization of the case for two classes. The main purpose of defining the concept of b-accurate is to ensure that none of the classes which are too far off from the optimum value are picked as a result of the sampling process. We can directly generalize the results of Lemma 1 in order to prove that the sampling process is b-accurate.

THEOREM 1: The probability that the sampling process results in a b-accurate reported majority class is given by at least $1 - (k - 1) \cdot e^{-l \cdot b^2/2}$.

Proof: See Appendix. ■

Theorem 1 presents the theoretical analysis of the accuracy of the proposed random walk-based approach. It states that even with random walks, the proposed classification method is able to report the same majority class as a process which uses an infinite amount of sampling with at least a certain probability. This suggests that the proposed DyCOS framework is robust to the potential noise inherent in the sampling approach of the random walk process.

4. SUPERVISED KEYWORD-BASED CLUSTERING

In many social networking applications, it is desirable to create supervised clusters from the underlying data with the use of the content. For example, a marketing firm may wish to determine clusters of users that are interested in a variety of different products which are specified by corresponding keywords. In such cases, it may be desirable to determine supervised keyword-based clusters, as opposed to purely unsupervised clusters from the data. Supervised clusters are also often much richer and useful when they are designed with application-specific criteria.

One naive solution is to simply identify pages in the social network for which the text content contains the corresponding keywords. However, such a solution is often not very effective because the text in a social network is quite noisy and sparse. Therefore, many nodes may exist which do not contain the relevant keyword, but which are quire relevant to that topic. Furthermore, clustering is an application in which we wish to not only determine entities which are directly related to a particular keyword, but also those which are indirectly related to it. The linkage information in social networks is extremely relevant to the clustering process, and it can be leveraged with the text-content in order to design an effective clustering algorithm.

We will design an effective voting algorithm which uses alternate phases of propagation and voting in order to construct the clusters around the different nodes. The algorithm works with a set $\mathbb{S}$ which corresponds to nodes that have been currently assigned to one or more clusters. We initially start off with a small set of nodes $\mathbb{S}$, and successively expand with the use of an iterative propagation and voting mechanism. Let $\mathcal{K}$ be the set of keywords, which are input to the supervised clustering algorithm. We assume that the number of keywords in set $\mathcal{K}$ also corresponds to the number of clusters which need to be found. We denote the number of clusters which the algorithm needs to find by $t = |\mathcal{K}|$. In the first step, we set $\mathbb{S}$ to be the set of nodes which contains the keywords in $\mathcal{K}$. Each node is initially assigned to the index of cluster corresponding to its contained keyword. We further note that the assignment of nodes in $\mathbb{S}$ to cluster indices can change in later iterations depending upon the results of the propagation process.

In the first iteration, we start random walks from nodes in $\mathbb{S}$, each of which is already associated with a cluster index. As in the case of the classification application, we perform l hops of length h for each. For each node in the entire graph, we maintain a set of votes for the different cluster indices. These votes are initially set to zero for each cluster index and each node. For each node visited by the random walk, we add a vote to the cluster index of the starting point of that walk. We repeatedly pick a random node in $\mathbb{S}$ in order to perform the random walk of length h. This process is

Algorithm 2 DyCOS Clustering Process

```
Data: Network $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{A}_t, \mathcal{T}_t)$, number of random walks, $l$, walk length, $h$, structural
      hop probability, $p_s$, keyword set, $\mathcal{K}$, clustering threshold, $\sigma$
Result: $\mathcal{K}$-supervised clusters $\mathcal{C}_\mathcal{K}$
1  $\mathbb{S} \leftarrow$ nodes containing the keywords in $\mathcal{K}$;
2  Assign each node in $\mathbb{S}$ with a cluster index;
3  while $\mathbb{S}$ not converged yet do
4      for Each node $v$ in $\mathbb{S}$ do
5          for $i$ from 1 to $l$ do
6              Perform an $h$-hop random walk from $v$, with structural hop probability,
               $p_s$, in which for each node visited, add a vote to the cluster index of $v$;
7      Tally the total number of votes received by each node and for each cluster index;
8      For those whose maximum cluster index is greater than $\sigma$, add them to $\mathbb{S}$ and
       associate them with the corresponding cluster index;
9      Update $\mathcal{C}_\mathcal{K}$;
10 return $\mathcal{C}_\mathcal{K}$;
```

repeated l times. Thus, each walk increases the number of votes at the different nodes. At the end of the entire process, we tally the total number of votes received by each node and for each cluster index. For each node which has received nonzero votes, we determine the cluster index which has received the largest number of votes. If this cluster index has received greater than σ votes (for some threshold σ), then we add this node to the set $\mathbb{S}$ and associate it with the corresponding cluster index. We note that nodes which are already present in $\mathbb{S}$ may also have votes associated with them. In most cases, their votes will be in agreement with the current cluster index. However, if their votes are not in agreement with the current cluster index, then the assigned cluster index of that node is changed. We note that at the end of the iteration, several nodes would have been added to $\mathbb{S}$.

In the next iteration, we start off with this expanded set $\mathbb{S}$, and repeat this process in an identical way. In this case, the starting points for the walk can be any node which is randomly chosen from this expanded set $\mathbb{S}$. This process is continuously repeated until the set $\mathbb{S}$ no longer expands significantly in a given iteration. At this procedure the algorithm terminates. The overall procedure is illustrated in Algorithm 2.

We note that not all nodes may be included in $\mathbb{S}$ at the termination of the algorithm. This is natural, because some nodes may be outlier nodes which are not relevant to any natural cluster. Furthermore, a supervised clustering algorithm does not always naturally include all nodes in relevance to its initial supervision. In the next section, we will show the interesting clusters which are naturally obtained with the use of such a technique.

One interesting characteristic of the random walk method for clustering is that the number of votes received by each node is somewhat analogous to a page-rank style importance of that node. Of course, since the random walks are performed on the basis of both link and content, the page-rank style importance also naturally incorporates this goal. Therefore, this approach also provides a natural way of determining the most influential (or well connected) nodes in each cluster *both* from a link and content point-of-view.

5. EXPERIMENTAL RESULTS

In this section, we validate the effectiveness and efficiency of DyCOS with experiments on real data sets. The effectiveness is measured by classification accuracy, which is the proportion of correctly classified nodes as to the total number of test nodes classified. The efficiency is measured by the run time of classification. We report the wall-clock time. In order to establish a comparative study, we compare the performance of DyCOS to that of NetKit-SRL toolkit,[1] which is an open-source network learning toolkit for statistical relational learning [22]. The results obtained in a multiclass classification environment demonstrate that DyCOS is able to improve the average accuracy over NetKit-SRL by 7.18–17.44%, while reducing the average runtime to only 14.60–18.95% of that of NetKit-SRL. Note that NetKit-SRL package is a generic toolkit without particular optimization for our problem definition. In order to illustrate

[1] http://www.research.rutgers.edu/~sofmac/NetKit.html.

Table 1. Data set description.

Name	Nodes	Edges	Classes	Labeled nodes
CORA	19 396	75 021	5	14 814
DBLP	806 635	4 414 135	5	18 999

the efficiency of DyCOS in a dynamic environment, the model update time in the presence of incrementally arriving data is also reported. This confirms that the classification model can be dynamically maintained in an efficient way.

5.1. Experiment Setup

Two real data sets are used, the CORA data set and the DBLP data set, as in Table 1. The class number is the number of distinct classes the nodes belong to and the labeled node number is the number of nodes whose class label is known. Each data set is segmented into a set of subgraphs. The classification is performed dynamically when the graph increasingly evolves from containing only the first subgraph to containing all subgraphs.

CORA data. The CORA graph is downloaded from http://www.cs.umass.edu/~mccallum/code-data.html. It contains a set of research papers and the citation relations among them. There are 19 396 distinct papers and 75 021 citation relations among them. Each node is a paper and each edge is a citation relation. A total of 12 313 English words are extracted from the titles of those papers to associate each paper with keywords. The CORA data set is well-suited for our experiments because the papers are classified into a topic hierarchy tree with 73 leaves. Each leaf represents a specific research area in computer science. We reconfigure the hierarchy to achieve a more coarse-grained classification. We extract five classes out of the 74 leaves, and 14 814 of the papers belong to these five classes. As the CORA graph does not include temporal information, we segment the data into 10 subgraphs, representing 10 synthetic time periods. Table 2 shows the graph size for each time period.

DBLP data. The DBLP graph is downloaded from http://www.informatik.uni-trier.de/~ley/db/ and updated until March 27, 2010. It contains 806 635 distinct authors and 4 414 135 collaboration edges among them. Each node is an author and each edge represents a coauthor relationship. A total of 194 English words in the domain of computer science are manually collected to associate authors with keyword information. The number of occurrences of each word is calculated based on the titles of publications associated with each author. We use five class labels, which denote five computer science domains: computer architecture, data mining, artificial intelligence, networking, and security. We associate some of the authors with ground-truth class labels using information provided by ArnetMiner,[2] which offers a set of comprehensive search and mining services for academic community. In total we have collected class labels for 18 999 authors. We segment the whole DBLP graph into 36 annual graphs from year 1975 to year 2010. Table 3 shows the average graph size for three time periods: 1975–1989, 1990–1999 and 2000–2010.

NetKit-SRL toolkit. The well-known NetKit-SRL toolkit was used for comparative study. NetKit-SRL, or NetKit for short, is a toolkit for learning from and classifying networked data. It is open-source and publicly available. It is aimed at estimating the class membership probability of unlabeled nodes in a partially labeled network [22]. NetKit contains three key modules: local classifier, relational classifier, and collective inferencing. For more details on NetKit, we refer the readers to Ref. [22]. In our experiments, we use domain-specific class-prior as the local classifier, network-only multinomial Bayes classifier as the relational classifier and relaxation labeling for collective inferencing.

Several parameter settings were tested in order to examine DyCOS's performance under various conditions.

1. The number of most discriminative words, denoted by m (Section 2).
2. Number of top two-hop paths, n (Section 3.1).
3. The size constraint of the inverted list for word i (Section 2.1), denoted by a, that is, $|P_i| <= a$. This is required to control the extent to which each word should be expanded to form 2-hop paths from the current source node. Such a constraint is necessary to increase the efficiency of the random-walk process.
4. The structure parameter p_s (Section 3).
5. Number of random walks for each source node, l (Section 3.1).
6. Number of hops in each random walk, h (Section 3.1).

In the following sections, the classification performance of DyCOS is first compared to that of NetKit. We then examine how well DyCOS performs in a dynamic environment. The model sensitivity to two important parameters, the number of most discriminative words, m, and the size constraint of inverted lists for words, a, is examined later. Finally, we show case studies on the supervised keyword-based clustering using DyCOS. All experiments are run in Fedora 8 on an one-core Intel Xeon 2.5 GHz machine with 32 GB RAM.

[2] http://www.arnetminer.org/.

Table 2. Segment graph size of CORA.

	1	2	3	4	5	6	7	8	9	10
$\|V\|$	1500	1500	1500	1500	1500	1500	1500	1500	1500	1314
$\|E\|$	4743	1110	611	384	263	262	201	147	0	0

Table 3. Average segment graph size of DBLP.

	1975–1989	1990–1999	2000–2010
$\|V\|$	7950	41 192	129 638
$\|E\|$	7166	57 624	267 787

5.2. Classification Performance

The classification performance is measured by both accuracy and efficiency. The accuracy is the fraction of correctly classified nodes. The efficiency is reflected by the wall-clock run time in seconds. The parameter setting is: $m = 5$, $n = 10$, $a = 30$, $p_s = 0.7$, $l = 3$, and $h = 10$. We will show later DyCOS is not significantly sensitive to these parameters. The setting for NetKit is as aforementioned. We utilize the '-test' option in NetKit to ensure a consistent set of test samples compared to DyCOS.

5.2.1. Comparative study on CORA data set

In this section, we compare the DyCOS and Netkit algorithms on the CORA data. Figure 2(a) shows the average classification accuracy of both DyCOS and Netkit for each synthetic time period. Clearly, DyCOS enables a performance gain ranging from 9.75 (time period 1) to 22.60% (time period 7). The average accuracy increase by DyCOS is 17.44% on CORA.

The comparison of run time is shown in Fig. 2(b). As shown, DyCOS is much more efficient in terms of run time. The run time of DyCOS is only a portion of that of NetKit, ranging from 12.45% (time period 10) to 16.70% (time period 7). The average run time of DyCOS is 14.60% that of NetKit on CORA.

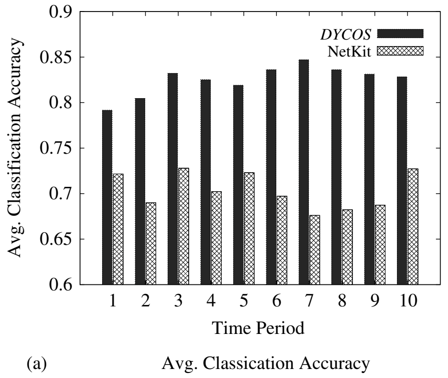
5.2.2. Comparative study on DBLP

Next, we present the comparative results on DBLP data. In order to establish a dynamic view, we divide the entire 36-year course of time into three periods, 1975–1989, 1990–1999, and 2000–2010. Figure 3(a) presents the average accuracy of both DyCOS and Netkit, for each time period. DyCOS achieves a performance gain ranging from 1.75 (time period 2000–2010) to 13.73% (time period 1975–1989). The average accuracy increment induced by DyCOS is 7.18% on DBLP.

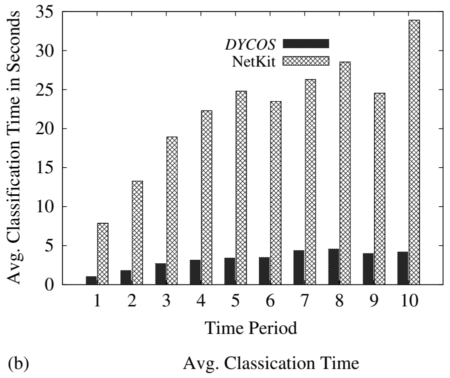
The comparison of run time is shown in Fig. 3(b). As shown, DyCOS again decreases the run time significantly. The run time of DyCOS is only a portion of that of NetKit, ranging from 11.30 (time period 2000–2010) to 21.30% (time period 1975–1989). The average run time of DyCOS is 18.95% that of NetKit on DBLP.

5.3. Dynamic Update Efficiency

In this section, we investigate the efficiency of DyCOS in the presence of dynamically arriving data. DyCOS handles temporally-evolving graphs with efficient update mechanisms. Table 4 presents the average model update time (in seconds) of DyCOS when new data from the next synthetic time period arrives, on CORA data. The average model update time over all 10 time periods is 0.015 s. Table 5 presents the average annual model update time (in seconds)

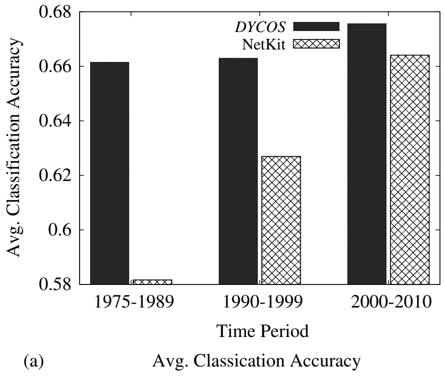


(a) Avg. Classication Accuracy



(b) Avg. Classication Time

Fig. 2 Comparative study on CORA.



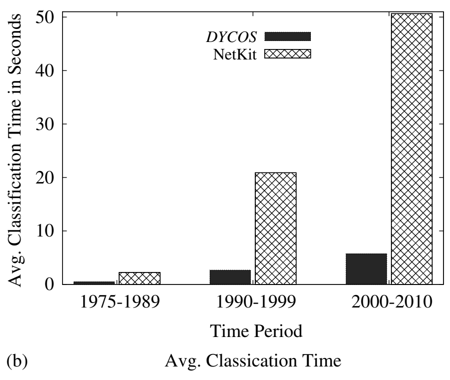


Fig. 3 Comparative study on DBLP.

Table 4. Average dynamic update time on CORA.

Time period	1	2	3	4	5
Update time (s)	0.019	0.013	0.015	0.013	0.023
Time period	6	7	8	9	10
Update time (s)	0.015	0.014	0.014	0.013	0.011

Table 5. Average dynamic update time on DBLP.

Time period	1975–1989	1990–1999	2000–2010
Update time (s)	0.03107	0.22671	1.00154

of DyCOS over various time periods, on DBLP data. The average annual model update time over all 36 years is 0.38 s. The results demonstrate the efficiency of maintaining the DyCOS model under a dynamically evolving network environment. This is a unique advantage of DyCOS in terms of its ability to handle dynamically updated graphs.

5.4. Parameter Sensitivity

The purpose of this section is to examine the sensitivity of DyCOS to various parameters, and their impact on DyCOS performance. Due to limited space, we demonstrate the sensitivity study on two particularly important parameters, the number of most discriminative words, m (m test), and the size constraint of inverted lists for words, a (a test). Results suggest that there is no significance correlation between the performance and the values of m and a. The robustness of DyCOS is therefore illustrated since it does not heavily rely on parameter setting, and more efficient choices can achieve equally effective results.

5.4.1. Parameter sensitivity on CORA

For m test on CORA data, three different scenarios are created, which are (i) $n = 5, l = 3, h = 5, a = 10, p_s = 0.5$, (ii) $n = 8, l = 5, h = 10, a = 20, p_s = 0.6$, and (iii) $n = 10, l = 8, h = 15, a = 30, p_s = 0.7$. Figures 4(a) and 4(b) demonstrate how classification accuracy and run time change over different m values, respectively. It is evident that there is no significant correlation between m and the classification performance. For a test on CORA data, three different scenarios are also created, which are (i) $n = 5, l = 3, h = 5, m = 5, p_s = 0.5$, (ii) $n = 8, l = 5, h = 10, m = 10, p_s = 0.6$, and (iii) $n = 10, l = 8, h = 15, m = 15, p_s = 0.7$. Similar results can be found in Figures 4(c) and 4(d).

5.4.2. Parameter sensitivity on DBLP

The same scenarios are created for m test and a test on DBLP as well. Figures 4(e) and 4(f) demonstrate the sensitivity to parameter m in terms of accuracy and run time. Interestingly, we can observe that, on DBLP data, classification accuracy reduces slightly when m increases. This is expected because a higher value of m implies that less discriminative words are included into computing two-hop paths during random walks. The run time increases with m. This is expected because a content hop takes more time to process if there are more discriminative keywords. The results in Figs 4(g) and 4(h) show no significant correlation between performance and a on DBLP.

5.5. Case Study on Supervised Clustering

As mentioned earlier, DyCOS can be extended to create supervised clusters with the use of the content. In this section, we perform interesting case studies by providing qualitative analysis on some of the clusters discovered by our voting algorithm. Clustering is initiated and supervised by a set of intuitive keywords. We will show that the use of a combination of a propagation scheme with content analysis for the clustering process, creates clusters which are able to incorporate entities that are indirectly related to the

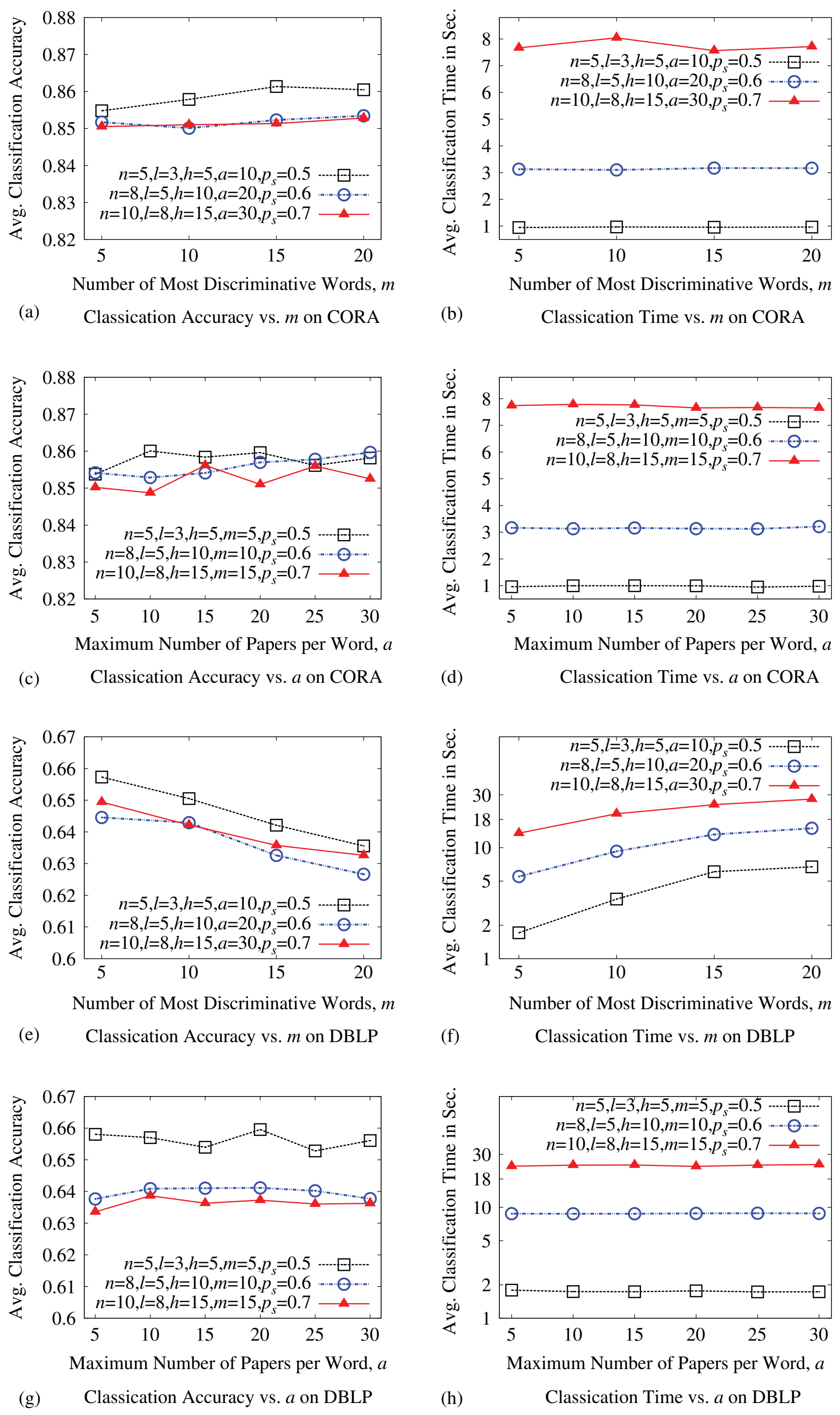


Fig. 4 Parameter sensitivity. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

keywords. This is because the use of linkage information provides useful information to the clustering process.

Table 6 shows some interesting clustering results using three sets of keywords (KW set) on the DBLP graph of year 2008. The three keyword sets are: {'Data', 'Mining', 'Knowledge', 'Information'}, {'Statistical', 'Stochastic', 'Probability'}, and {'Artificial', 'Intelligence', 'Learning', 'Recognition'}. After clustering, each keyword will be associated with a group of authors. The motivation lies in observing the author community discovered for each keyword of interest. Among the authors clustered for a given keyword, some might directly contain that keyword

Table 6. Case study: supervised keyword-based author clustering on DBLP, year 2008.

KW Set	KW		Author cluster	Perc.
# 1	"Data"	KW Authors	Dan Suciu, Alon Y. Halevy, Charu Aggarwal, Divyakant Agrawal, Amr El Abbadi, Johannes Gehrke, Jian Pei, etc.	21.67%
		Non-KW Authors	Gerhard Weikum, Vassilis J. Tsotras, Sara Cohen, Laurence Tianruo Yang, Wagner Meira Jr., Feng Liu, etc.	
	"Mining"	KW Authors	Xifeng Yan, Stephen R. Schach, Gerhard Friedrich, Michael Philippsen, Dingyi Han, Frans Coenen, etc.	14.96%
		Non-KW Authors	Joann J. Ordille, Clement H. C. Leung, Indrajit Bhattacharya, Xuhua Ding, Zoran Duric, etc.	
	"Knowledge"	KW Authors	Yoram Moses, Amit Sahai, Yoav Shoham, Salil P. Vadhan, Rafail Ostrovsky, Chitta Baral, etc.	18.57%
		Non-KW Authors	Mark A. Musen, Joachim Biskup, John Shepherd, Ralph Bergmann, Avinash C. Kak, Wilhelm Hasselbring, etc.	
	"Information"	KW Authors	Jiawei Han, Rajiv Gupta, George Karypis, Philip A. Bernstein, Daniel S. Weld, Craig A. Knoblock, etc.	22.18%
		Non-KW Authors	Stanley Y. W. Su, Colette Rolland, Weijia Jia, Camille Salinesi, William Donnelly, Michael Kohlhase, etc.	
# 2	"Statistical"	KW Authors	Hermann Ney, Timothy F. Cootes, Christopher J. Taylor, Franz Joseph Och, David Blaauw, Vladimir Zolotov, etc.	23.48%
		Non-KW Authors	Stefanos Gritzalis, Andrew Turpin, Attilio Giordana, Ravi Mukkamala, Qiang Zhu, Giacomo Cabri, etc.	
	"Stochastic"	KW Authors	Eli Upfal, Michael L. Littmanm, Eitan Altman, B. John Oommen, Joost-Pieter Katoen, David B. Shmoys, etc.	21.33%
		Non-KW Authors	Shimon Whiteson, Martin Wirsing, Aly A. Farag, Wen Gao, Jing Zhang, Michael Jurczyk, etc.	
	"Probability"	KW Authors	Henri Prade, Glenn Shafer, Prakash P. Shenoy, Didier Dubois, Irwin King, Gert de Cooman, etc.	25.13%
		Non-KW Authors	Kevin Leyton-Brown, Alex Acero, Daniel M. Reeves, Davide Rossi, John Illingworth, Xiaolong Wang, etc.	
# 3	"Artificial"	KW Authors	Jonathan Grudin, Andrei Voronkov, Marco Colombetti, Cristiano Castelfranchi, Rino Falcone, Xin Yao, etc.	19.30%
		Non-KW Authors	Christophe G. Giraud-Carrier, Jerzy W. Grzymala-Busse, Alexander Pokahr, Lars Braubach, Erin Shaw, etc.	
	"Intelligence"	KW Authors	Umeshwar Dayal, Sotiris Malassiotis, Epaminondas Kapetanios, Mukesh K. Mohania, Divyakant Agrawal, Panos Vassiliadis, Volker Markl, etc.	16.99%
		Non-KW Authors	Ning Zhong, Partha Pratim Pal, Michael Atighetchi, Indrajit Bhattacharya, Paul A. Beardsley, etc.	
	"Learning"	KW Authors	Avrim Blum, Michael Kearns, Thomas G. Dietterich, Rocco A. Servedio, Sanjay Jain, Lorenza Saitta, etc.	21.50%
		Non-KW Authors	Raghu Ramakrishnan, Pat Langley, Vipin Kumar, Michael Steinbach, Stuart M. Shieber, Heikki Mannila, etc.	
	"Recognition"	KW Authors	Rajiv Gupta, Luca Benini, Stuart M. Shieber, Kotagiri Ramamohanarao, Mannes Poel, Chin-Chen Chang, etc.	19.88%
		Non-KW Authors	Michael Collins, Andrea Cavallaro, Nikos Nikolaidis, Michel Scholl, Peter Sanders, etc.	

Table 7. Case study: supervised keyword-based author clustering on DBLP, year 2009.

KW Set	KW		Author Cluster	Perc.
# 1	"Data"	KW Authors	Serge Abiteboul, Michael Stonebraker, Dan Suciu, Alon Y. Halevy, Rakesh Agrawal, Yuri Breitbart, etc.	21.70%
		Non-KW Authors	Jeffrey D. Ullman, Yannis E. Ioannidis, Henry F. Korth, Curtis E. Dyreson, Fabio Martinelli, Takeshi Naemura, etc.	
	"Mining"	KW Authors	Athman Bouguettaya, Sharma Chakravarthy, Xiaohui Liu, Athena Vakali, Dimitrios Gunopulos, Maguelonne Teisseire, etc.	15.14%
		Non-KW Authors	Gregory F. Cooper, Craig W. Thompson, Lawrence V. Saxton, Pranam Kolari, Steven C. H. Hoi, etc.	
	"Knowledge"	KW Authors	Shamkant B. Navathe, Joseph Y. Halpern, Ronald J. Brachman, John Mylopoulos, Amit P. Sheth, Henry Lieberman, etc.	17.88%
		Non-KW Authors	Pierangela Samarati, Richard T. Snodgrass, Jennifer Widom, Davide Fossati, Werner Geyer, Wilfried Lemahieu, etc.	
	"Information"	KW Authors	W. Bruce Croft, C. J. van Rijsbergen, Daniel S. Weld, Jiawei Han, Craig A. Knoblock, Alon Y. Halevy, etc.	21.02%
		Non-KW Authors	Daniela Rus, Paul Johannesson, Eugene J. Shekita, Xue Li, Alexandros Nanopoulos, Ioannis P. Vlahavas, etc.	
# 2	"Statistical"	KW Authors	Hermann Ney, Salil P. Vadhan, David Blaauw, Surajit Chaudhuri, Peter J. Haas, Thierry Lecroq, etc.	6.39%
		Non-KW Authors	Jennifer Widom, Gail E. Kaiser, Dorothy E. Denning, John D. Lafferty, Arie Shoshani, etc.	
	"Stochastic"	KW Authors	Michael I. Jordan, Gianfranco Balbo, Gianfranco Ciardo, B. John Oommen, Eitan Altman, Michael L. Littman, etc.	6.27%
		Non-KW Authors	Amit P. Sheth, Clement T. Yu, Umeshwar Dayal, Stavros Christodoulakis, Weiyi Meng, Alfons Kemper, etc.	
	"Probability"	KW Authors	Ronald Fagin, Tao Jiang, Reynold Cheng, Eibe Frank, Cory J. Butz, Vincent Rijmen, Joan Daemen, etc.	6.56%
		Non-KW Authors	John A. Stankovic, Paul C. van Oorschot, Fernando Lyardet, Matthias Baaz, Dario Maio, Patrick Cousot, etc.	
# 3	"Artificial"	KW Authors	Marco Colombetti, Teresa Bernarda Ludermir, Leandro Nunes de Castro, Dietmar Jannach, Ioannis P. Vlahavas, etc.	19.51%
		Non-KW Authors	Vipin Kumar, Gerald J. Sussman, Leif Kobbelt, Marek R. Ogiela, Wolfgang Maass, etc.	
	"Intelligence"	KW Authors	Georg Gottlob, Andrea Omicini, Enrico Denti, Anton Nijholt, Alkis Simitsis, Isabelle Bloch, etc.	14.94%
		Non-KW Authors	Stefanos D. Kollias, Alan P. Sprague, Omer F. Rana, Antonio Ortega, Rajkumar Buyya, etc.	
	"Learning"	KW Authors	Elisa Bertino, Felix Naumann, Wolfram Burgard, Michael I. Jordan, Dana Angluin, Thomas G. Dietterich, Dan Roth, etc.	21.99%
		Non-KW Authors	Jeffrey D. Ullman, Jennifer Widom, Philip A. Bernstein, Richard T. Snodgrass, Victor Vianu, etc.	
	"Recognition"	KW Authors	Xiaoou Tang, Bir Bhanu, Ching Y. Suen, P. Jonathon Phillips, Harry Wechsler, Ramez Elmasri, etc.	21.77%
		Non-KW Authors	Daniel A. Keim, Hatice Gunes, Bertrand Meyer, Norio Shiratori, Atsushi Nakazawa, etc.	

in their content (KW Authors), which is the set of paper titles of their publications in the year 2008, whereas the others might not (Non-KW authors).

For example, let us examine some cases from Table 6. Even though the author Indrajit Bhattacharya did not publish any paper, whose title contained the word 'Mining' in 2008, he is grouped into the cluster associated with the keyword 'Mining'. One of the reasons for this is his collaboration with other authors, such as Shantanu Godbole, who have published at least one paper in 2008 whose title contains 'Mining'. We argue such grouping is reasonable and conforms with intuition in terms of discovering valuable clusters. Another case in point is Vassilis J. Tsotras, who has been grouped into the 'Data' cluster because of collaboration with authors with this keyword in 2008, one of whom is Erik G. Hoel.

An additional aspect of DyCOS-based clustering is the effect of the content-based hops on the clustering process. In such cases, it is possible for authors with common keywords in their content to be clustered together. These common keywords are often correlated to the seed keywords, because of their presence in the seed documents. For example, Stefanos Gritzalis and Roberto Perdisci both work in security-related research areas, and they share some keywords in common in 2008, including 'clustering' and 'classification'. Clearly such keywords are often relevant to the original seed keywords for the clustering process, and can lead to relevant clusters. In addition, Roberto Perdisci has the word 'Statistical', which renders Stefanos Gritzalis to be grouped into the 'Statistical' cluster. This shows one of DyCOS's principles: nodes with similar content should be grouped together. This example suggests that both structural and content hops can be useful for the clustering process.

The results for DBLP in 2009 are shown in Table 7. An interesting example is the inclusion of Paul Johannesson in clusters related to the keyword 'Information'. Paul Johannesson and Jelena Zdravkovic both work in the area of business modeling and business process, and have collaborated in the year 2009. Jelena Zdravkovic contains the keyword 'Information' in 2009, which contributes to the clustering of Paul Johannesson into the 'Information' group. While the word 'Information' was not explicitly available in the titles of the latter's publications, we note that an important research interest of Paul Johannesson is information systems. This suggests the results of the clustering process are intuitive and reasonable, and clearly benefit from the use of linkage information. As for authors with similar content, examples include Omer F. Rana and Carlo Combi, who share common keywords in their content, such as 'workflow'. Since Carlo Combi has a paper with 'Intelligence' in the title in 2009, Omer F. Rana is also included in the 'Intelligence' cluster.

The rightmost column of both Tables 6 and 7 is the percentage of nonkeyword authors in each cluster. Clearly, our supervised clustering algorithm is able to discover a significant portion of authors who are indirectly related to the keyword of interest. Meanwhile, the algorithm is also able to find *groups of* authors who collaborate frequently. For example, Timothy F. Cootes and Christopher J. Taylor work closely together, and they are both grouped into the same cluster corresponding to the keyword 'Statistical' in 2008. Intuitively, we can consider each keyword-supervised cluster as a set of authors directly related to this keyword, together with their close collaborators, as well as authors that are similar to them in terms of research interests.

DyCOS-based clustering is also able to discover influential authors related to the different keywords by examining the individuals that received the most votes in each cluster. For example, for the cluster 'Knowledge' in 2009, we are able to determine that the most votes were received by Richard T. Snodgrass, Jennifer Widom, and Shamkant B. Navathe among others. For the cluster 'Learning' in 2009, we are able to find Wolfram Burgard, Michael I. Jordan, Jeffrey D. Ullman, Jennifer Widom, etc. Table 8 shows some examples of the discovered authors and the number of votes they get during random walks. In most cases, authors with larger number of votes are either influential in that respective field, or have collaborated with many scholars from that field. This shows that DyCOS-based clustering provides a formal way of determining which authors are important or influential in the field defined by a keyword-based cluster.

Table 9 exemplifies some of the interesting findings on a segment of the CORA graph. Similarly, each cluster contains either papers which directly contain the keyword, or papers that are indirectly related to it. For example, the paper 'Neuron-like adaptive elements that can solve difficult learning control problems' is grouped into the 'Information' cluster, because of the fact that it shares common keywords with some other papers which actually contain the keyword 'Information', such as 'The use of meta-level information in learning situation specific coordination' and 'Advances in neural information processing systems'. Other

Table 8. Case study: DBLP authors and their votes, year 2009.

"Knowledge"		"Learning"	
Author name	Vote #	Author name	Vote #
Pierangela Samarati	269	Jeffrey D. Ullman	360
Richard T. Snodgrass	221	Philip A. Bernstein	154
Jennifer Widom	168	Wataru Sunayama	151
Ulrich Furbach	76	Elisa Bertino	91
Shamkant B. Navathe	53	Felix Naumann	53
Dexter Kozen	51	Wolfram Burgard	52
Gerald DeJong	51	Michael I. Jordan	23
...	...	...	...

Table 9. Case study: supervised keyword-based paper clustering on CORA, Segment 1

KW Set	KW	Author cluster		Perc.
# 1	`"Data"`	KW Papers	'A data locality optimizing algorithm', 'A methodology for implementing highly concurrent data structures', etc.	3.77%
		Non-KW Papers	'Mach: a new kernel foundation for UNIX development', 'Scale and performance in a distributed file system', etc.	
	`"Mining"`	KW Papers	'Mining generalized association rules', 'Mining association rules between sets of items in large databases', 'Efficient and effective clustering methods for spatial data mining', etc.	11.11%
		Non-KW Papers	'A weighted nearest neighbor algorithm for learning with symbolic features', 'Discovery of multiple-level association rules from large databases', etc.	
	`"Knowledge"`	KW Papers	'The Knowledge complexity of interactive proof systems', 'The frame problem and Knowledge producing actions', etc.	7.70%
		Non-KW Papers	'Genetic algorithms in search, optimization and machine learning', 'A method for obtaining digital signatures and public-key cryptosystems', 'Learning to predict by the methods of temporal differences', etc.	
	`"Information"`	KW Papers	'Preserving and using context information in interprocess communication', 'Cooperative information gathering: A distributed problem solving approach', etc.	10.81%
		Non-KW Papers	'Neuron like adaptive elements that can solve difficult learning control problems', 'The stable model semantics for logic programming', etc.	
# 2	`"Statistical"`	KW Papers	'A statistical model for relevance feedback in information retrieval', 'A statistical approach to solving the EBL utility problem', etc.	15.38%
		Non-KW Papers	'Learning with many irrelevant features', 'Minorization conditions and convergence rates for Markov chain Monte Carlo', etc.	
	`"Stochastic"`	KW Papers	'Planning With Deadlines in Stochastic Domains', 'On the convergence of stochastic iterative dynamic programming algorithms', etc.	57.13%
		Non-KW Papers	'Learning to predict by the methods of temporal differences', 'An algorithm for probabilistic least-commitment planning', etc.	
	`"Probability"`	KW Papers	'Probabilistic independence networks for hidden Markov probability models', etc.	77.78%
		Non-KW Papers	'Operations for learning with graphical models', 'Bayesian graphical models for discrete data', etc.	
# 3	`"Artificial"`	KW Papers	'Communicative actions for artificial agents', etc.	66.67%
		Non-KW Papers	'A semantics approach for KQML: a general purpose communication language for software agents', 'Kqml as an agent communication language', etc.	
	`"Intelligence"`	KW Papers	'Intelligence without Robots (A Reply to Brooks)', etc.	50.00%
		Non-KW Papers	'Synchronization of multi-agent plans', etc.	
	`"Learning"`	KW Papers	'Learning to predict by the methods of temporal differences', 'Neuron like adaptive elements that can solve difficult learning control problems', etc.	65.79%
		Non-KW Papers	'A robust layered control system for a mobile robot', 'Introduction to the Theory of Neural Computation', etc.	
	`"Recognition"`	KW Papers	'Connectionist Speech Recognition: A Hybrid Approach', 'A Formal Theory of Plan Recognition and its Implementation', etc.	47.37%
		Non-KW Papers	'A Theory of Networks for Approximation and Learning', 'Recursive estimation of motion, structure, and focal length', etc.	

examples are the inclusion of ‘Minorization conditions and convergence rates for Markov chain Monte Carlo’ into the ‘Statistical’ cluster because of its common keywords with some other papers such as ‘Statistical and systematic errors in Monte Carlo sampling’, and the inclusion of ‘A weighted nearest neighbor algorithm for learning with symbolic features’ into the ‘Mining’ cluster because of its common keywords with papers such as ‘SMART-TV: a fast and scalable nearest neighbor based classifier for data mining’. Each cluster for a particular keyword therefore consists of a set of papers directly containing that keyword and other papers that are similar to them.

6. CONCLUSIONS AND SUMMARY

In this paper, we present an efficient, dynamic and scalable method for node classification in networks with both structure and content. The classification of content-based networks is challenging, because some parts of the network may be suited to structural classification, whereas others may be suited to content-based classification. Furthermore, many networks are dynamic, which requires us to maintain an incremental model over time. Our results show that our algorithms are scalable, and can be be applied to large and dynamic networks. We show the advantages of using a combination of content and linkage structure, which can provide more robust classification across different parts of a diverse network. We also show how to use the technique in order to find supervised clusters in social networks. Experimental results on real data sets show that our algorithms outperform competing algorithms in terms of both effectiveness and efficiency.

APPENDIX

7. DETAILS OF PROOFS

LEMMA 1: Let us consider two classes with expected visit probabilities of f_1 and f_2, respectively, such that $f_1 - f_2 > 0$. Then, the probability that the class which is visited the most during the sampled random hop process is reversed to class 2, is given by at most $e^{-l \cdot b^2/2}$.

Proof: Let X_i be the random variable which represents the fraction of nodes of class 1, which are visited during the ith random walk, and let Y_i be the random variable which represents the fraction of nodes of class 2, which are visited during the ith random walk. Then we define the random variable defining the *differential hop fraction* as $Z_i = X_i - Y_i$. It is clear that Z_i is a random variable which lies in the range $[-1, 1]$, and has an expected value of b. Then, we define the random variable S as the sum of the different values of Z_i over the l different random walks. Therefore, we have:

$$S = \sum_{i=1}^{l} Z_i. \tag{4}$$

As $E[Z_i] = b$, it follows that $E[S] = l \cdot b$. In order for the majority class to be class 2, we need $S < 0$. Therefore, we would like to determine the probability $P(S < 0)$. For this purpose, we will make use of the Hoeffding inequality, because S is expressed as a sum of *bounded* random variables in the range $[-1, 1]$. By using $E[S] = l \cdot b$, we get:

$$P(S < 0) = P(S - E[S] < -l \cdot b).$$

As S is the sum of l independent random variables, which lie in the range $[-1, 1]$, we can use the Hoeffding inequality to bound the probability of error, a proxy for which is the expression $P(< 0)$:

$$P(S < 0) = e^{-l \cdot b^2/2}.$$

■

THEOREM 1: The probability that the sampling process results in a b-accurate reported majority class is given by at least $1 - (k-1) \cdot e^{-l \cdot b^2/2}$.

Proof: The results of Lemma 1 show that the probability of pairwise error is at most $e^{-l \cdot b^2/2}$ because of the sampling process. Therefore, the probability of pairwise error for any of the (at most) $(k-1)$ other classes which have expected visit probability at most b of the optimum is given by at most $(k-1) \cdot e^{-l \cdot b^2/2}$. The result follows. ■

ACKNOWLEDGMENTS

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice hereon. We would also like to thank Xifeng Yan for his comments and suggestions.

REFERENCES

[1] C. C. Aggarwal, and N. Li, On node-classification in dynamic content-based networks, SDM Conference, 2011.
[2] C. C. Aggarwal, and H. Wang, Managing and Mining Graph Data, Springer, 2010.
[3] C. C. Aggarwal, Social Network Data Analytics, Springer, 2011.
[4] R. Duda, P. Hart, and D. Stork, Pattern Classification, Wiley, 2000.
[5] S. Bhagat, G. Cormode, and I. Rozenbaum, Applying link-based classification to label blogs, WebKDD/SNA-KDD, 2007, 97–117.
[6] M. Bilgic, and L. Getoor, Effective label acquisition for collective classification, KDD Conference, 2008, 43–51.
[7] Q. Lu, and L. Getoor, Link-based classification, ICML Conference, 2003, 496–503.

[8] T. Joachims, Text categorization with support vector machines: learning with many relevant features, ECML Conference, 1998, 137–142.

[9] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learn 39(2-3) (2000), 103–134.

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

[11] Y. Yang, An evaluation of statistical approaches to text categorization, Inform Retrieval 1(1–2) (1999), 69–90.

[12] S. Chakrabarti, B. Dom, and P. Indyk, Enhanced hypertext categorization using hyperlinks, SIGMOD Conference, 1998, 307–318.

[13] B. Taskar, P. Abbeel, and D. Koller, Discriminative probabilistic models for relational data, UAI, 2002, 485–492.

[14] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, Learning with local and global consistency, Adv Neural Inform Process Syst 16 (2004), 321–328.

[15] D. Zhou, J. Huang, and B. Schölkopf, Learning from labeled and unlabeled data on a directed graph, ICML Conference, 2005, 1036–1043.

[16] X. Zhu, Z. Ghahramani, and J. D. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, ICML Conference, 2003, 912–919.

[17] Y. Zhou, H. Cheng, and J. X. Yu, Graph clustering based on structural/attribute similarities, PVLDB 2(1) (2009), 718–729.

[18] V. R. de Carvalho, and W. W. Cohen, On the collective classification of email "speech acts", SIGIR Conference, 2005, 345–352.

[19] T. Zhang, A. Popescul, and B. Dom, Linear prediction models with graph regularization for web-page categorization, KDD Conference, 2006, 821–826.

[20] J. S. Vitter, Random sampling with a reservoir, ACM Transactions on Mathematical Software 11(1) (1985), 37–57.

[21] G. Jeh, and J. Widom, Scaling personalized web search, WWW Conference, 2003, 271–279.

[22] S. A. Macskassy, and F. Provost, Classification in networked data: a toolkit and a univariate case study I, J Mach Learn Res 8 (2007), 935–983.