On Dynamic Link Inference in Heterogeneous Networks

Charu Aggarwal* Yan Xie† Philip S. Yu‡

Abstract

Network and linked data have become quite prevalent in recent years because of the ubiquity of the web and social media applications, which are inherently network oriented. Such networks are massive, dynamic, contain a lot of content, and may evolve over time in terms of the underlying structure. In this paper, we will study the problem of dynamic link inference in *temporal* and *heterogeneous* information networks. The problem of dynamic link inference is extremely challenging in massive and heterogeneous information network because of the challenges associated with the dynamic nature of the network, and the different types of nodes and attributes in it. Both the topology and type information need to be used effectively for the link inference process. We propose an effective two-level scheme which makes efficient macro- and micro-decisions for combining structure and content in a *dynamic and time-sensitive* way. The time-sensitive nature of the links is leveraged in order to perform effective link prediction. We illustrate the effectiveness of our technique over a number of real data sets.

1 Introduction

In recent years, many forms of networked social media such as *Facebook* and *Flickr* have been rapidly burgeoning in terms of their membership and popularity. Many such social and media networks may contain different kinds of attributes such as text, tags or other meta-data, and may rapidly evolve over time. For example, the web, blog networks and social networks are dynamically interconnected with one another, and may continually experience a change in the node and linkage structure of the network. Such dynamic and heterogeneously connected entities are referred to as information networks. This has lead to a tremendous interest in the field of managing and mining such dynamic and heterogeneous information networks [14].

In many networks, the linkages are inherently dynamic, and continuously arrive over time. This results in a gradual change on the network structure over time. For example, in a social network, new linkages are continuously created over time. This often results in gradual densification of the underlying social network graph. In such cases, it may be desirable to predict future linkages between the entities. The derivation of links between entities is an extremely important problem from the perspective of a number of different social networking applications. This has lead to increasing interest in the problem of *automated inference* of the links in social networks [2, 3, 5, 19, 12, 20, 15, 8]. However, most of the known techniques are designed for homogeneous networks which are static in nature. On the other hand, our techniques will be focusing on dynamic and evolving networks, which contain a combination of different kinds of heterogeneous content and links. More specifically, many of the previously designed techniques for link prediction are not easily applicable to information networks because of the following reasons:

• Most of the available work on link inference is designed for the case of static network data sets, in which the network structure does not change significantly over time. In static cases, it is much easier to design models, since they do not need to be constructed in an incremental way, and the link prediction model can be constructed with the use of a multi-pass pre-processing approach. In this paper, we will examine a *dynamic evolving scenario* in which new links are continuously added and old links may also re-appear over time. For example, a friendship link is a static link, whereas a message sent between two participants is a dynamic link which could recur over time. In many applications, it is useful to be able to predict the most likely links at any given time. Our methods are designed for very large networks, which are not only dynamic, but also contain a very large number of nodes. For example, for a network containing 10^7 nodes, the number of possible pairs of nodes is of the order of 10^{14}. This creates a challenge for predicting the most likely pairs of nodes between which links exist. The challenge is especially great when the link-prediction model needs to be constructed in a dynamic way.

• Current techniques are often designed for homogeneous networks, which are based either purely on structure [15], or on attributes which are all of the same type [19]. On the other hand, in truly heterogeneous networks, both the nodes and the links can be of different types. These techniques are not very effective for cases in which nodes have a very *large number of different types with practically arbitrary content structure in different types*. For example, in a heterogeneous information network, the nodes may be of different types such as author, conference, or paper. Each of these different

*IBM T.J. Watson Research Center, Email: charu@ibm.com
†University of Illinois at Chicago, Email: yxie8@uic.edu
‡University of Illinois at Chicago, Email: psyu@cs.uic.edu

types may have different kinds of content or attributes. Thus, our model is very flexible, and can apply to practically any kind of dynamic and content-based network.

The *dynamic nature* of the network implies that new links are constantly being added to the network. Such new links may also arrive in the context of new nodes being added to the network, or they may correspond to edges between already existing nodes. In some cases, an entire set of nodes together with its associated links may be received by the network, whereas in others, only individual edges or nodes may be received. Both the two scenarios are not very different from a conceptual point of view, because they can both be modeled with the scenario that a single node together with its links is received. Therefore, all future discussions in this paper will focus on this scenario. One challenge with the dynamic approach is that the structure of the network may evolve over time. This may affect our ability to perform effective link prediction.

In order to achieve these goals, we will use a dynamic graph-clustering approach in which fine-grained clusters are constantly maintained in the network. These clusters are created on the basis of structural similarity. The goal of the clustering process is to create a *dynamic summarization* which can be efficiently used for dynamic inferences in a very large network. The higher level of macro-processing divides the network into regions of high density in which more fine-grained decisions with the use of types and contents can be made. This structural behavior is used for *macro-decisions*, whereas both structural and attribute behavior is used for *micro-decisions* of deciding where the links should be placed. Thus, the link inference decisions are made with the use of a combination of content and links, within a particular structural locality of the network. We will show that such a local approach, which combines the content and structural information in a careful way, is very useful for the case of content-rich and heterogeneous information networks. We will refer to our algorithm as *DYNALINK* which corresponds to the fact that it is as Dynamic and Heterogeneous Content-based Link Prediction Algorithm.

This paper is organized as follows. The remainder of this section discusses related work and contributions. In section 2, we will discuss how to leverage the statistics for the problem of link-prediction. Section 3 will study a number of experimental results. Section 4 contains the conclusions and summary.

1.1 Related Work and Contributions The problem of link prediction has been studied extensively in the data mining and machine learning community [7]. Much of the work on this problem is based on defining proximity-based measures on the nodes in the underlying network [1, 15, 16]. The work in [15] studied the usefulness of different topological features for link prediction. It was discovered in [5] that none of the features was particularly dominant in different kinds of situations. A second approach is to study the problem in the context of statistical relational models [6, 9, 10, 11, 21]. However, these methods are restricted to relational models, and are not designed for dynamic networks, or cannot handle attributes of a relational nature. Recently, the problem of link prediction has also been studied in the context of wikipedia and web data [2, 22].

The link prediction problem has also been studied more generally in the context of the classification problem [5, 12, 20], since the link prediction problem can be considered as a classification problem in which features and class labels (corresponding to existence or absence of links) can be associated with links to be predicted. While some work has focussed recently on some aspects of the heterogeneous scenario [17, 18], previous work has not been designed for *massive*, *heterogeneous* and *dynamic* networks, in which the content in different nodes can take on almost any free form. The paper is particularly unique in its approach to *dynamic heterogeneous networks* by using a dynamic network clustering approach which combines local content predictability with temporal decay-based structural probability. This paper takes a unique approach towards such large scale networks by using topological behavior for higher-level decisions by using them in the clustering process, and the attribute behavior for more fine grained decisions.

1.2 Link Inference: Problem Definition In this section, we will define the link inference problem for information networks. We assume that each node has a type associated with it. This type may be quite different depending upon the kind of network. For example, in a paper-authorship network, this type could correspond to paper, author, conference, or other corresponding entity. In a movie database, the type could correspond to actor, movie or genre. The links between the different entities represent the nature of the relationships among them. These links could be of different types depending upon the nature of the underlying relationships. For example, a link could be a "co-authorship" relationship between two author nodes, or it could be an "authorship" relationship between an author node and a paper node.

Many of the applications which generate such networks are inherently dynamic. For example, co-authorship networks, or military information networks are inherently dynamic in nature. Therefore, it may be assumed that new nodes or edges are constantly being added to the network, and similarly new nodes or edges are constantly being deleted. In our paper, we assume that each incoming entity may be a *set of nodes* together with the edge relationships between them. Furthermore, some of the incoming nodes may never have been encountered before. For example, in a co-authorship network, new authors and papers are continu-

ously being added to the network.

We assume that a node of type r has d_r different attributes. The type of the node is itself one of these d_r attributes. These values are assumed to be discrete, though numerical values can also be converted to discrete values with the use of a discretization process. We note that there may be nodes of q different types which are denoted by $\{1 \ldots q\}$. These values are thus the relational attributes, which represent the properties of the different nodes. Thus, for a node i with type t_i, these values are denoted by $x_1^i \ldots x_{d_{t_i}}^i$ respectively. These values can often be helpful in link inference, because the correlations in the values across the different nodes can be used for the inference process. We also assume that the *domain of values* for different attributes is distinct. This assumption is without loss of generality, because we can use a transformation in order to ensure that the values are distinct. Specifically, we can concatenate the following strings, in order to create the following new attribute-value string: (a) A string containing the attribute name, (b) the symbol "#", and (c) the attribute value itself. For example, consider the case when attribute 3 of an author-type node is a keyword, for which one of the possible values is "clustering", and attribute 2 of a paper-type node is also a keyword, for which one of the possible values is "clustering". Then, both values are represented as "clustering#keyword". However, if we track demographic attributes, one of which is gender, then the value of the attributes in a node corresponding to a female would be the value "gender#female". We denote the *entire domain of distinct values* across all attributes and node types by $\mathcal{D}$. We assume that $\mathcal{D}$ is the *index of* L distinct values, which are denoted by $\mathcal{D} = \{1 \ldots L\}$. The distinctness of the content values at different nodes ensures that the attribute values at each node can be generally treated as a *bag of values* at each node. This is essentially the same as a vector-space representation of text data. In fact, the techniques of this paper are easily generalized to the case in which each node-type contains text content as opposed to a fixed set of discrete attributes for each node type.

In addition, for any pair of nodes i and j, a link (i, j) may exist, and the type of the corresponding link is denoted by $P(i, j)$. The type of the corresponding link is drawn from $\{1 \ldots p\}$. Different kinds of queries can be formulated in the context of this problem. These are as follows:

(a) For a given pair of nodes, predict the relative importance of a link arriving between them in the future.

(b) For a given node, determine the links of a particular type which are most likely to emanate from that node in the future.

(c) Predict all the links of a particular type in the network in the future.

We note that all these queries need to be resolved in a dynamic way which takes into account the evolving structure and content of the network.

2 Link Inference: The Dynamic Network Clustering Approach

One of the key challenges in link inference in heterogeneous information networks is that the link-prediction process requires the use of both linkage and attribute information. Furthermore, the dynamic nature of the network makes the prediction process even more challenging. One observation in [15] is that the topological linkage structure can be leveraged quite effectively for link prediction. However, in many cases, the attributes may also contain valuable information for the link prediction process. However, the attribute information can often be *sensitive to a particular locality of the network*. For example, in a bibliographic information network, the keyword attribute corresponding to the word "network" may not be very discriminative within the node cluster corresponding to the networking area, but may be quite discriminative within the database or data mining community. Therefore, the link-prediction process can be greatly enhanced with the use of *local* and *context-sensitive* information within a particular topological region. In order to leverage the content information in a more discriminative way, we will use a carefully designed approach which is dynamic and properly accounts for the local structural and content information during link prediction. We use the following broad approach:

- The clustering process is used in order to segment the network into different local regions. Each region is densely populated, and is more likely to contain a larger portion of the links. Furthermore, each local region is likely to provide context-specific linkage behavior of the different attributes.
- We use the clustered network *in conjunction with the relational attributes at different nodes* in order to design rules which relate the attribute combinations as well as the *local linkage structure* to predict the likelihood that a link will arrive in the future between a pair of nodes. Since the network is already clustered, the model for the relational attributes is based on the dense linkage structure within a particular region. This is likely to make the model much more robust, because it is based on the *local characteristics* of the network within a particular region. This sharpens and magnifies the accuracy of the approach.

In addition to the locality-specific advantages of the class discrimination behavior, the clustering process also helps in segmenting the network into regions of much more manageable size. This is essential in a very large information network in which the number of nodes is very large, and there may be too many attribute values to process on a global basis. On the other hand, the set of relevant attribute values

within a particular local region may be much more concise. This greatly helps in magnifying the link predictability properties. Once it is known that most of the edges in the network lie within these clustered regions, we can further use the information in the relational attributes in order to make more nuanced predictions on the linkages. The implicit effect of using this two-stage process is to use global linkage skews in order to make the *macro-decisions* of where most of the links are located, and then the *local correlations* among the attributes in the individual node types for the *micro-decisions* for picking the exact pairs of nodes at which to place these links. Therefore, we will describe the overall approach for link prediction by describing the following in the next subsections: (a) We will describe the methodology for creating the clusters as well as the maintenance of corresponding summary statistics. A proper choice of summary statistics is critical for an effective link prediction process. Therefore, we will first describe the summary statistics. (b) In a later subsection, we will describe how to use these clusters and the associated statistics for link prediction. The described techniques use the summary statistics for the link prediction process.

2.1 Cluster Summary Statistics The main idea is to create a *compact characterization of the linkage behavior* which is *local to* each cluster. The compactness of the characterization is useful in ensuring an efficient link-prediction process. We assume that each cluster consists of a set of nodes $\mathcal{C}$, which are densely connected to one another with the use of links of different types. The summary statistics which are stored with the clusters are as follows:

- We compute the frequency $f(m, \mathcal{C})$ of the attribute value $m \in \mathcal{D}$ over the cluster $\mathcal{C}$. Therefore $f(m, \mathcal{C})$ is the number of nodes of cluster $\mathcal{C}$ in which the attribute value m is present.
- We maintain the number of links of each type such that both of its end points lie in the cluster $\mathcal{C}$. The number of such links of type k in cluster $\mathcal{C}$ is denoted by $B(k, \mathcal{C})$.
- For all links (i, j) of type k for which both ends lie in the cluster $\mathcal{C}$, we compute the number of occurrences of each attribute value $m \in \mathcal{D}$ which are present at the source of the link. If multiple links of type k emanate from i, then the corresponding link is also counted multiple times. This value is aggregated over all nodes i in the cluster. This is the *origination frequency* of attribute value $m \in \mathcal{D}$ for links of type k, and is denoted by $O(m, k, \mathcal{C})$.
- For all links (i, j) of type k for which both ends lie in the cluster $\mathcal{C}$, we compute the number of occurrences of each attribute value $m \in \mathcal{D}$ which are present at the destination of the link. If multiple links of type k are incident to j, then the corresponding link is also counted multiple times. This value is aggregated over all nodes j in the cluster. This is the *destination frequency* of attribute value $m \in \mathcal{D}$ for links of type k, and is denoted by $E(m, k, \mathcal{C})$.
- For all links (i, j) of type k for which both ends lie in the cluster $\mathcal{C}$, we compute the number of occurrences of the attribute pair $m_1 \in \mathcal{D}$ and $m_2 \in \mathcal{D}$, such that m_1 occurs at i and m_2 occurs at j. This value is denoted by $I(m_1, m_2, k, \mathcal{C})$.
- In addition, the similarity in attribute values across links can also be an indicator of linkage behavior. Therefore, for each attribute value m, we compute the number $Qn(m, k, \mathcal{C})$ of links of type k which have the attribute value m at both ends within cluster $\mathcal{C}$.

We note that all of the above statistics are based on attribute and link behavior, which are *local to* a particular cluster. We also maintain *global statistics* which are true across the different clusters. The main difference is that these statistics are maintained at the global level, rather than simply about the local behavior of particular clusters. These statistics are useful for making predictive decisions about the links, when the end points may lie in different clusters. We track the following analogous statistics:

- The number of nodes at which the attribute m occurs globally is denoted by $h(m)$.
- The number of links of type k in the entire network is denoted by $A(k)$.
- For all links (i, j) of type k in the network, we compute the number of occurrences of each attribute value $m \in \mathcal{D}$ which are present at the source of the link. If multiple links of type k emanate from i, then the corresponding link is also counted multiple times. This value is aggregated over all nodes i in the network. This is the *origination frequency* of attribute value $m \in \mathcal{D}$ for links of type k, and is denoted by $OG(m, k)$.
- For all links (i, j) of type k in the network, we compute the number of occurrences of each attribute value $m \in \mathcal{D}$ which are present at the destination of the link. If multiple links of type k are incident to j, then the corresponding link is also counted multiple times. This value is aggregated over all nodes j in the network. This is the *destination frequency* of attribute value $m \in \mathcal{D}$ for links of type k, and is denoted by $EG(m, k)$.
- For all links (i, j) of type k for which the source i and destination j lie in different clusters, we compute the number of occurrences of the attribute pair $m_1 \in \mathcal{D}$ and $m_2 \in \mathcal{D}$, such that m_1 occurs at i and m_2 occurs at j. This value is denoted by $J(m_1, m_2, k)$.

Algorithm *MaintainCluster*(Incoming Graph Object: O);
begin
  for each new node in object O assign it to cluster
   which results in the least number of new
   inter-cluster links;
  Sort other nodes in O in random
   order and check if re-assignment
   improves objective function;
  Update cluster link statistics;
end

Figure 1: Incremental Clustering Process

- For each attribute value m, we compute the number $Pn(m, k)$ of links of type k which have the same attribute value m at both ends.

We will use these summary statistics in conjunction with the clustering process in order to accomplish the link prediction process. We note that these statistics are maintained dynamically along with the clusters. In the next section, we will discuss the process of dynamic cluster maintenance.

2.2 Dynamic Cluster and Statistics Maintenance In this section, we will propose techniques for dynamic cluster maintenance with a focus on link prediction. The clustering process partitions the node set N into a group of clusters, $\mathcal{C}_1, \mathcal{C}_2 \ldots \mathcal{C}_r$. Since the link inference method of this paper is designed for *dynamic information networks*, the clustering process needs to be dynamic as well. As an initialization step, we start off with the initial state of the information network clusters which are derived with the use of any of the standard node clustering algorithms [4]. For the purpose of this paper, we will use a simple graph partitioning algorithm which divides nodes into r partitions, so as to minimize the number of inter-partition edges. A classic example of this is the Kernighan-Lin algorithm [4]. Subsequently, we need a method to maintain *both* the clustering structure and the link statistics in the presence of dynamic changes of the information network. We assume that this dynamic nature of the information network is reflected by *incoming graph objects*, each of which may have a set of nodes and links. For example, in a co-authorship network, each object may correspond to a research paper, in which the nodes are papers, authors, or conferences. The links may represent an authorship relationship, or paper-conference relationships. It is possible that some of these nodes may not currently be present in the information network at all. For example, when a paper is written by an author who has not published before, this corresponds to a completely new node. We note that the incoming graph objects may not immediately affect the clustering structure of the underlying nodes, which are already present in the network. However, the addition of such objects may result in the movement of nodes from one partition to the other and vice-versa. Such changes may happen over time, when the structure of the network changes. While the network structure may be very large, its detailed structure needs to be tracked during the clustering process. Specifically, we need to maintain information about the nodes, their attribute values and their adjacent nodes. For this purpose, the adjacency list representation can be used very efficiently.

We dynamically maintain the sets of clusters $\mathcal{C}_1 \ldots \mathcal{C}_r$. When a new graph G_r arrives, its nodes (which are already present in the information network) are assumed to belong to the corresponding clusters. The *new* nodes are greedily assigned to the clusters which result in the *least* number of links across the different clusters. After this assignment process, we update the inter-cluster and intra-cluster link based statistics in the previous section. In many instances, it may be the case that the network structure changes over time, and therefore the assignment of nodes to clusters may change as well. For this, we only check the nodes *involved in the current object* to be re-assigned. For each node in the current object, we sort them in random order, and check if a re-assignment to any of the other clusters reduces the number of links across the different clusters. If such is the case, then the re-assignment is performed. We note that this step may also result in adjustment of inter-cluster and intra-cluster statistics. It is fairly straightforward to update these summary statistics by using the disk-resident representation to examine the node attributes and its outgoing links on disk. The re-assignment of clusters changes the links within the clusters as well as the links across the clusters. Correspondingly, the statistics are also modified in order to reflect this re-adjustment. The overall update process for an incoming object is illustrated in Figure 1.

2.3 Computing Content Predictability In this section, we will discuss how to leverage the statistics collected in the afore-mentioned sections for the purpose of link-prediction. In order to achieve this goal, we will first construct rules which relate the attribute values at the source and destination of the link to the probability of link prediction. For this purpose, the summary statistics maintained in each cluster are very useful. Therefore, we define the concept of local *predictability* of links with respect to particular attribute pairs.

DEFINITION 1. (LOCAL PREDICTABILITY) *The local predictability $S(m_1, m_2, k, \mathcal{C})$ of attribute-pair (m_1, m_2) and link-type k with respect to the cluster $\mathcal{C}$ is the probability that for a given node-pair (i, j) completely contained within cluster $\mathcal{C}$, the link (i, j) of type k exists, conditional on the fact that node i contains attribute-value m_1 and node j contains attribute-value m_2. This local predictability is estimated as a weighted average of four quantities, for weights $\alpha_1 \ldots \alpha_4$,*

which satisfy $\sum_{i=1}^{4} \alpha_i = 1$*:*

$$S(m_1, m_2, k, \mathcal{C}) = \alpha_1 \cdot \frac{O(m_1, k, \mathcal{C})}{f(m_1, \mathcal{C}) \cdot |\mathcal{C}|} + \alpha_2 \cdot \frac{E(m_2, k, \mathcal{C})}{f(m_2, \mathcal{C}) \cdot |\mathcal{C}|} + \alpha_3 \cdot \frac{I(m_1, m_2, k, \mathcal{C})}{f(m_1, \mathcal{C}) \cdot f(m_2, \mathcal{C})} + \alpha_4 \cdot \frac{B(k, \mathcal{C})}{|\mathcal{C}|^2}$$

We note that the weights $\alpha_1 \ldots \alpha_4$ *can be learned by testing over a grid of values and picking the optimum combination on a small hold our portion of the training data. In the event that any of the fractions above is indeterminate*[1]*, we exclude that term from the computation.*

Each of these terms corresponds to a different way of using the attribute structure in order to estimate the local link probability. A detailed explanation for each of these terms is as follows:

- The first fraction $\frac{O(m_1,k,\mathcal{C})}{f(m_1,\mathcal{C})}$ performs the estimation by analyzing the behavior of the links which emanate from a node containing a particular attribute type.
- The second fraction performs the estimation by analyzing the behavior of the links which are incident on a node containing a particular attribute type.
- The third fraction uses both the source and destination behavior of a particular node.
- The last fraction uses only the frequency behavior of the links in the cluster and does not use attribute structure at all. This is useful in cases, where much information about the behavior of a particular kind of link may not be available.

A similar concept can be defined in terms of the *global* predictability $G(m_1, m_2, k)$ of the attribute-pair (m_1, m_2), with respect to link type k.

DEFINITION 2. (GLOBAL PREDICTABILITY) *The global predictability* $G(m_1, m_2, k)$ *of attribute-pair* (m_1, m_2) *and link-type* k *is the probability that for a given node-pair* (i, j)*, the link* (i, j) *of type* k *exists, conditional on the fact node* i *contains attribute-value* m_1 *and node* j *contains attribute-value* m_2*. Let* N *be the total number of nodes in the network currently. This probability is estimated as a weighted average of four fractions, with weights* $\beta_1 \ldots \beta_4$*, which satisfy* $\sum_{i=1}^{4} \beta_i = 1$*:*

$$G(m_1, m_2, k) = \beta_1 \cdot \frac{OG(m_1, k)}{h(m_1) \cdot N} + \beta_2 \cdot \frac{EG(m_2, k)}{h(m_2) \cdot N} + \beta_3 \cdot \frac{J(m_1, m_2, k)}{h(m_1) \cdot h(m_2)} + \beta_4 \cdot \frac{A(k)}{N^2}$$

[1]This refers to the fact that the numerator and the denominator of the fraction may be 0.

As in the previous case, the values of $\beta_1 \ldots \beta_4$ can be learned by testing over a grid of values and picking the optimum combination. Note that the global-predictability is useful in capturing the behavior of those links for which the end points do not lie in the same cluster. The concept of predictability essentially defines rules for the link-prediction process.

Therefore, the first step is to determine the values of $S(m_1, m_2, k, \mathcal{C})$ and $G(m_1, m_2, k)$, for each attribute-value pair (m_1, m_2), and sort them in descending order. Note that this is done offline periodically in the case of a dynamic network, because it may be time-consuming to compute this statistic over all pairs of attribute-values (m_1, m_2).

In addition, we determine the discriminatory attributes-values which are based on *content-similarity*. The locally discriminatory attribute values for cluster $\mathcal{C}$ and link type k, denoted by $La(k, \mathcal{C})$, are all the attributes m for which the value of $Qn(m, k, \mathcal{C})/f(m, \mathcal{C})^2$ is larger than the mean value over all the attribute values. Similarly, we define $Pa(k)$ as the set of all attributes for which the value of $Pn(m, k)/h(m)^2$ is larger than the mean value over all attribute values. Note that $La(k, \mathcal{C})$ and $Pa(k)$ define attribute values for which similarity between nodes also defines higher probability of a link.

2.4 Dynamic Structural Measures In addition to the local content-based similarity measures, we also calculate the pairwise structural similarity between nodes. It is important to note that while pairwise similarity measures between *attributes* are on the basis of content, the pairwise similarity measures between *nodes* are on the basis of structure. The pairwise structural similarity *between nodes* is computed as a weighted function of the following quantities: (a) The *decay-weighted* number of links between the two entities (b) The *decay-weighted* similarity in neighbors between the two entities.

In order to enable *efficient* and *dynamic* computation of the link prediction process, we do not use more complex structural measures such as path lengths between nodes. Since the techniques discussed in this paper are designed for the case of a dynamic network, it is important to use *temporal decay* in the process of modeling the number of links between the two entities. We did not use the decay behavior in the content-based similarity because we generally found the content-behavior across the network to be much more stable with time as compared to the structural behavior. Therefore, it is more critical to use the decay-behavior in structural computations as compared to the content computations. We define the decay-weighted frequency of a link as follows:

DEFINITION 3. *Let* t *be the current time, and* $t_1 \ldots t_r$ *be the time stamps at which the link between a particular pair of links* (i, j) *were received. Then, the decay weighted frequency* $DF(i, j, k, t)$ *is of link* (i, j) *of type* k *at time* t

defined as $\sum_{i=1}^{r} 2^{-\lambda \cdot (t-t_i)}$. *Here* λ *is the decay parameter.*

The decay-based frequencies of the neighbors of each node are dynamically tracked over time. We keep track of only the neighbors of each node which have non-zero frequency. We note that this can be a challenge, because the decay based frequencies are continuously changing at each tick, and we do not want to update all the frequencies at a given time. However, the update can be performed in a *lazy fashion*, since the decay-based frequencies for all nodes *decay at the same rate unless a new link is added*. This refers to the following observation:

OBSERVATION 1. *If the link* (i, j) *is not received in the time interval* $[t_1, t_2]$*, then we have:*

$$DF(i, j, k, t_2) = DF(i, j, k, t_1) \cdot 2^{-\lambda \cdot (t_2 - t_1)} \quad (2.1)$$

Therefore, we make the multiplicative update for the decay function only when a new link is added. Therefore, if t_s was the last time a link (i, j) of type k was received, and t_c be the current time at which a link is received, then, we update the decay-based frequencies only at times t_s and t_c. At current time t_c, we first multiply the link frequency $DF(i, j, k, t_s)$ by $2^{-\lambda \cdot (t_c - t_s)}$, and then add 1.

Thus, for each link type, each node has a vector of decay frequencies which are *dynamically* maintained along with it. The length of this vector is essentially the number of neighbors of that node which are based on links of type k. We define the structural similarity vector at a node as follows:

DEFINITION 4. *The structural similarity vector of a node* i *at time* t_c *for links of type* k *is the set* $NS(i, k)$ *of neighbors of that node together with the value of* $DF(i, j, k, t_c)$ *for each* $j \in NS(i, k)$.

We further note that in many cases, the decay process will ensure that some components of this vector will become smaller and smaller over time. These correspond to those nodes which may have been a neighbor at some point, but have not been *active* neighbors for a while. Such components do not contribute much to the computation process, but they increase the space- and time-requirements. Therefore, it is best to prune such components. Therefore, at the time of updating a node, we check all the components of the vector of that node, and remove all components which are less 0.1% the magnitude of the average component in it. We note that since such networks are typically sparse (and an even smaller percentage of the links are active), the vector maintained at each node is very small. Thus, for each node, we maintain a list of the neighbor nodes with a non-zero component of the decay frequency, and the actual value of the decay frequency. Then, the structural similarity between a pair of nodes for links of type k at time t_c can be computed in the form of the following two measures:

- The first measure is the direct structural similarity $DF(i, j, k, t_c)$.
- The indirect structural similarity is the dot product of the structural similarity vectors of i and j for links of type k. This number is denoted by $IDF(i, j, k, t_c)$. In other words, of $\overline{QV(i, k, t_c)}$ and $\overline{QV(j, k, t_c)}$ be the vectors at i and j respectively, then the dot product is given by:

$$IDF(i, j, k, t_c) = \overline{QV(i, k, t_c)} \cdot \overline{QV(j, k, t_c)} \quad (2.2)$$

2.5 Queries The statistics which are computed above can be leveraged for an effective link prediction process. We describe the techniques below:

Query 1: *Determine the predictability-score of a link of type* k *between a particular pair of nodes* i *and* j.
We note that the predictability-score is a number which helps in the *relative rankings* of linkages between nodes, rather than serving as a true indicator of predictability values. There are several factors which are combined in order to compute the final predictability score. These factors are as follows: (a) The content-based predictability (b) The content similarity (c) The (direct and indirect) structural similarity.
In order to resolve this query, we first determine the sets of attribute values $V(i)$ and $V(j)$ present at nodes i and j. In addition, we determine the cluster memberships of nodes i and j respectively. In the event that the cluster memberships of nodes i and j are not the same, then we use the global-predictability $G(m_1, m_2, k)$ for each attribute-value pair $m_1 \in V(i)$ and $m_2 \in V(j)$. The average of the top t predictability values among the different pairs are computed as a first step in order to create the predictability score. On the other hand, if the nodes i and j belong to the same cluster $\mathcal{C}$, then we repeat the same computation with the use of the local predictability values $S(m_1, m_2, k, \mathcal{C})$. This component defines the content based predictability and is denoted by $CP(i, j, k, t_c)$ at the current time t_c.

Furthermore, we include a contribution for the similarity in attribute values between the node pairs. Specifically, we add the cosine similarity between $V(i) \cap Pa(k)$ and $V(j) \cap Pa(k)$ to the predictability score, or the cosine similarity between $V(i) \cap La(k, \mathcal{C})$ and $V(j) \cap La(k, \mathcal{C})$ if the cluster memberships of nodes i and j are the same. This value is denoted by $CS(i, j, k, t_c)$ at the current time t_c.

Finally, we also have the direct and indirect structural similarity components. These structural similarity values are denoted by $DF(i, j, k, t_c)$ and $IDF(i, j, k, t_c)$. Then, the total link prediction score $TPS(i, j, k, t_c)$ is defined as a weighted sum of these different components with the use of balance parameters $\gamma_1 \ldots \gamma_4$, and is defined as follows:

$$TP(I, j, k, t_c) = \gamma_1 \cdot CP(i, j, k, t_c) + \gamma_2 \cdot CS(i, j, k, t_c) + \\ + \gamma_3 \cdot DF(i, j, k, t_c) + \gamma_4 \cdot IDF(i, j, k, t_c)$$

The value of the balance parameters $\gamma_1 \ldots \gamma_4$ is chosen in a data driven manner by testing for different variations over a small part of the training data, and then picking the optimum value of the combination for the test data. A small grid of values for the balance parameters $\gamma_1 \ldots \gamma_4$ is used, and this is used for the testing for the optimum combination over a small part of the training data set.

Query 2: *Determine the q most likely links of type k emanating from node i.*
In this case, the response to the query is in the form of a **ranked list** of all the links emanating from node i. One possible way to achieve a resolution to this query is to repeat the query over all possible pairs of nodes emanating from node k. This can however be time consuming, since the number of possible nodes in the information network can be very large. Therefore, a natural way to resolve the query is to first identify a small structural locality of the network based on the decay-based values $DF(i, j, k, t_c)$. We first determine all nodes in the network which are within a distance at most h, of node i with the use of only nodes for which $DF(i, j, k, t_c) > \epsilon$, where ϵ is a small number such as 0.1. This effectively uses only the active neighbors of each node in the exploration process. The value of h is typically a small number such as 2 or 3. Once these nodes have been identified, we can repeat the process of query 1, directly on this much smaller subset of nodes.

Query 3: *Determine the q most likely links of type k.*
The naive way of solving this problem would be to apply query 1 *over all pairs of nodes*. However, this can be extremely inefficient, since the number of pairs of nodes is quadratically related to a potentially large number. As in the previous case, we construct a network which is based on edges (i, j) for which the value of $DF(i, j, k, t_c)$ is at least ϵ. For each node i, we compute the aggregate value of $DF(i, j, k, t_c)$ of all nodes j incident in it. We process the nodes in decreasing order of this aggregate value, and repeat the process of query 2 in order to determine the most likely links. We dynamically keep track of the q most likely links. The processing of each node may lead to some new links which join the set of q most likely links. However, as more and more nodes are processed, the updating of the set of most likely links happens less frequently. We terminate, when an update does not happen in at least t consecutive iterations. We note that this is a heuristic termination point, but at large values of t, such as 1% of the number of nodes, this provides an effective solution.

3 Experimental evaluation

In this section, we will test the effectiveness and efficiency of our proposed *DYNALINK* algorithm on a number of real data sets. We will first give a description of the data sets, and discuss the experimental setup. Finally, we will present the experimental results. As a baseline, we choose to use a chance-constrained link prediction formulation (*CBSOCP*) introduced in [8]. As we will see later, our experimental studies sufficiently illustrate that our approach can effectively predict future links in a dynamic scenario on both heterogeneous and homogeneous graphs. Furthermore, in spite of the greater generality of the *DYNALINK* algorithm, it is much more effective and efficient even in the homogeneous scenarios which are designed for the *CBSOCP* method.

Table 1: Data Description

Dataset	# papers	# authors	# edges
DBLP	23,329	25,950	107,997
Genetics	11,463	41,868	159,746
Biochemistry	14,151	49,982	184,029

3.1 Data Sets The algorithms were tested on three real data sets, which are similar with the ones used in the baseline algorithm described in [8]. The main difference is that a heterogeneous network structure was derived from some of the data sets in order to test the effectiveness of the *DYNALINK* algorithm in this scenario. Furthermore, a dynamic environment was simulated to test the dynamic aspects of our algorithm. The three data sets used are described below.

The first data set is a heterogeneous co-authorship network derived from the well-known *DBLP* data set[2]. We extract all the papers published in 20 conferences related to database, data mining, information retrieval and machine learning from 1996 to 2009.

The other two are the *Genetics* and *Biochemistry* data sets, which are derived from the popular PubMed database[3]. In particular, the *Genetics* dataset includes a collection of publication in 14 journals related to genetics and molecular biology from 1996 to 2005, while the *Biochemistry* data set contains articles published in 5 journals related to biochemistry also from 1996 to 2005. The size of three data sets is summarized in Table 1.

3.2 Experimental Setup As discussed earlier, our proposed dynamic approach focuses on predicting linkage in dynamic information networks in a dynamic and temporal way. In order to simulate this dynamic scenario, we divide each data set into three parts as follows:

- The first part is for initialization which includes graph partitioning and feature extraction.
- The second part acts as the dynamic part where we dy-

[2] http://dblp.uni-trier.de/
[3] http://www.ncbi.nlm.nih.gov/entrez

namically update the summary statistics and structural similarities.

- The last part corresponds to the testing set.

For the case of the *DBLP* data set, the papers collected between 1996 and 2003 are used for initialization, while we treat all the later publications as new incoming objects till 2008. The papers in 2009 are used to generate the test set. For both the *Genetics* and *Biochemistry* data sets, the initialization set includes all articles in the first 4 years (1996 to 1999), and then we continuously receive new publications from 2000 to 2004. As in previous cases, the publication collection of the last year is used for testing purposes.

Since our proposed algorithm is expected to work well on both heterogenous and homogeneous cases, we generate our input from both perspectives. In the graph derived from the *DBLP* data set, there are two types of nodes: author and conference. Accordingly, we generate two types of links: author-author links and author-conference links. For the other two data sets, we test them in the homogeneous scenario. In other words, we generated only author nodes and author-author links for them.

3.2.1 Feature Description and Parameter Setting Since the data sets used in our experiment are derived from co-authorship networks, we decided to use the words in the paper titles as the attribute of each node. In general, an author who has published many papers has a longer list of attributes. The proposed algorithm has several parameters. In the graph partitioning step, we divided the graph into $r = 100$ partitions. The same number of 100 partitions is consistently maintained in the dynamic phase of the algorithm. The decay parameter λ in the calculation of the decay weighted frequency is set to be 1. We also tested several combinations of the balance parameters γ_1, γ_2, γ_3 and γ_4. Then we pick the combination of $\{0.2, 0.1, 0.5, 0.2\}$ as this is one of the settings that give us an effective value over different data sets. The setting of the *CBSOCP* algorithm is exactly the same as described in [8].

3.3 Accuracy Analysis In order to quantify the effectiveness of our approach, we use the concepts of precision and recall as evaluation method, and compare our result with the chance-constrained based algorithm (*CBSOCP*) [8]. We used an exactly similar testing methodology as discussed in *CBSOCP*. Since each data set has a large number of nodes and it is sometimes infeasible to test all combinations, the *CBSOCP* method considered all the links in the testing graph as positive examples and collect a sample of all the negative links as negative examples. Precisely, half of the negative links were chosen for testing purposes according to the method discussed in [8]. In order to ensure consistent comparison between the two algorithms, we used the same set of positive and negative examples in both cases.

To evaluate the effectiveness of our algorithm, we calculate all test cases with the model from the training process, and rank them in descending order of the prediction scores. Note that in the case of the *CBSOCP* method, the ranking is based on the margin of the classifiers. It is natural to choose the top-k links of the list to be predicted as positive, and thus precision and recall metrics can be calculated by varying the value of k. Here, precision is defined as the percentage of true positive links that are predicted correctly among the top-k predictions and recall is defined as the percentage of true positive links that are predicted correctly out of the complete set of true positive links. Higher values of k lead to lower precision but higher recall.

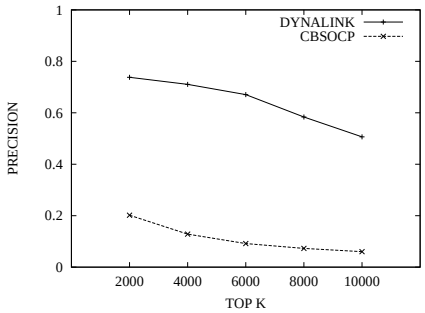

Figure 2: Precision Plot (*dblp* author-author links)

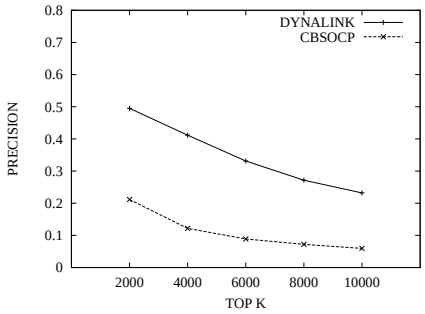

Figure 3: Precision Plot (*dblp* author-conference links)

Figures 2 and 3 depict the prediction precision of the two different types of links in the heterogeneous graph derived from *DBLP* data set over different values of k. The value of k is illustrated on the X-axis, and the prediction precision is illustrated on the Y-axis. The value of k on the X-axis varies from 2,000 to 10,000. Note that the training procedure of our *DYNALINK* algorithm is carried out as a single process for both author-author links and author-conference links. With the same model, we can predict different kinds of links even though the precisions are shown separately. In contrast, since the algorithm *CBSOCP* can only work on a particular type

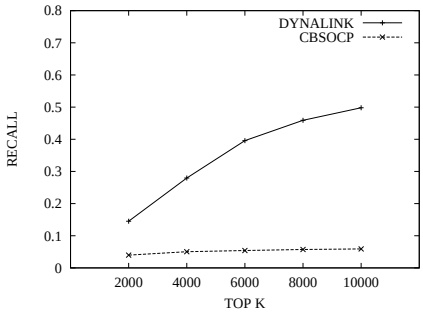

Figure 4: Recall Plot (*dblp* author-author links)

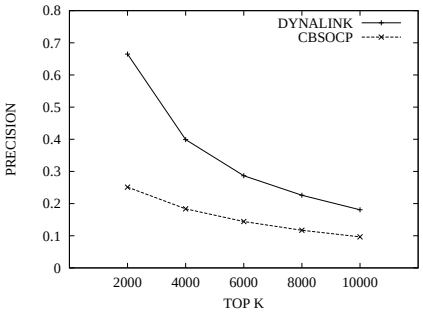

Figure 6: Precision Plot (*Genetics*)

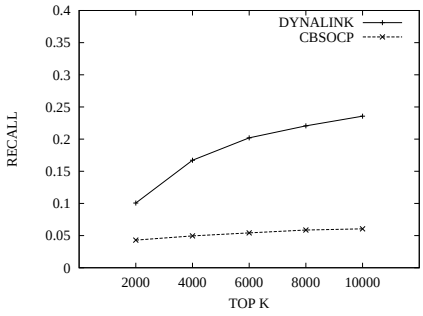

Figure 5: Recall Plot (*dblp* author-conference links)

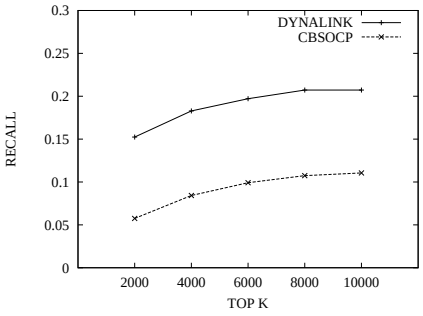

Figure 7: Recall Plot (*Genetics*)

of links, to make the comparison, we have to train the data twice, each of which focuses on a particular type. In spite of this, we can see that our dynamic link prediction scheme is **significantly** superior to the baseline algorithm *CBSOCP* in terms of precision for both types of links. For example, when we set the value of k at 2,000, the precision for our dynamic link prediction scheme is 73.8%, whereas that for the *CBSOCP* method was 20.2%. As expected, the precision drops off for both methods as k increases. However, the *DYNALINK* algorithm continues to maintain reasonably high precision even when the value of k increases.

The recall with increasing value of k of the *DBLP* data set are illustrated in Figures 4, and 5. As in the case of the precision plots, there are two recall plots for the heterogeneous case of *DBLP* data set, each of which contributes to a different type of links. The *DYNALINK* method is superior to *CBSOCP* in terms of recall, which means that our dynamic prediction approach can retrieve more true positive links in the top-k predictions. This phenomenon is more pronounced in the case of *DBLP* data set shown in Figures 4 and 5. It is evident that the recall curve of *CBSOCP* algorithm in these two figures is almost flat, and therefore there is no additional advantage of picking a larger value of k for increasing recall. On the contrary, the corresponding recall curve of our method illustrates a a rapidly increasing trend for larger values of k. For example, in the prediction of author-author links, the recall of the *DYNALINK* method is 14.5% when k is set to $2,000$, and it jumps to 49.8% when the first 10,000 predictions are chosen.

The precision plot with increasing value of k for the *Genetics* data set is illustrated in Figure 6. As in previous case, the *DYNALINK* scheme achieves much higher precision than the *CBSOCP* algorithm. The gap between the two curves in the precision plot is more obvious when the value of k is relatively small. For example, when we aim at top 2,000 predictions, the precision of the *DYNALINK* method is 66.5% while the *CBSOCP* algorithm has a precision of only 25.1%. Figure 7 shows the corresponding recall plot with increasing value of k for the *Genetics* data set. In this case, the recall curve shows an increasing trend for both *DYNALINK* and *CBSOCP* methods. However, the *DYNALINK* scheme consistently reaches a much higher recall value, and retrieves more true positive links over the entire range of values of k.

The precision and recall plots of the *Biochemistry* data set are illustrated in Figures 8 and 9 respectively. As can be seen from the figures, the *DYNALINK* scheme is extremely robust in the sense that it outperforms the *CBSOCP* algorithm for every value of k in both precision and recall plots. In the recall plot, even though the recall of the *CBSOCP* algorithm increases with k, our *DYNALINK* scheme has a much faster increasing trend. At the lower end, when k is

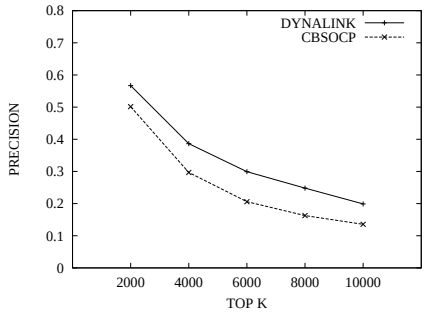

Figure 8: Precision Plot (*Biochemistry*)

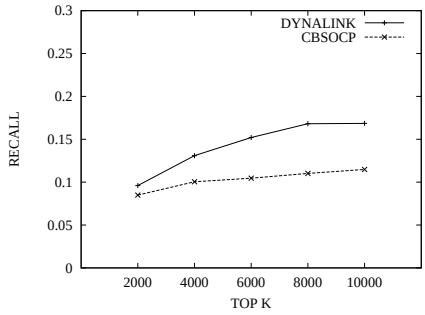

Figure 9: Recall Plot (*Biochemistry*)

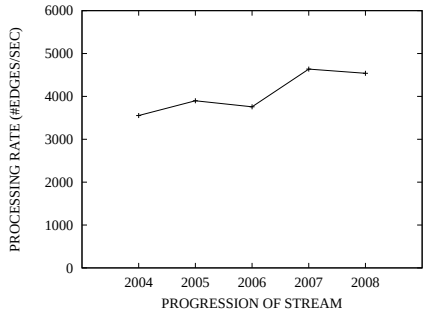

Figure 10: Efficiency on Data Stream (*dblp*)

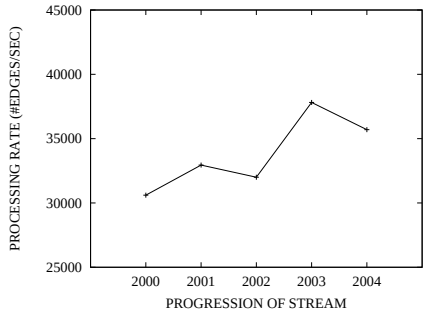

Figure 11: Efficiency on Data Stream (*Genetics*)

set to be 2,000, the recall of the *DYNALINK* method outperforms the *CBSOCP* algorithm by a factor of about 1.1. On the other hand, at the higher end, when we use the top 10,000 predictions, the *DYNALINK* method outperforms the *CBSOCP* algorithm by a factor of 1.5.

3.4 Efficiency Analysis All experiments are done on a Debian GNU/Linux server with two dual-core Xeon 3.0GHz CPUs and 16GB main memory. The software was written in C++.

Table 2: Computational Time

	DYNALINK (sec)	*CBSOCP* (sec)
DBLP	1,078	4,599
Genetics	785	12,448
Biochemistry	1,452	4,026

As in the case of the qualitative results, we used the *CBSOCP* method as the baseline approach. The computational efficiency of both algorithms is illustrated in Table 2. Note that the running time shown in the table includes all parts of a complete training procedure. For the *DYNALINK* algorithm, a complete procedure involves initialization, attribute extraction, and model statistics maintenance. On the other hand, the overall process of *CBSOCP* comprises feature calculation, clustering and model training as well. We further note that while the *DYNALINK* algorithm can be maintained online in a dynamic way, this is not the case for the *CBSOCP* algorithm. From the table, we can see that for all three testing data sets, the *DYNALINK* algorithm runs faster than the baseline *CBSOCP*. One major reason is the features used in *CBSOCP* are more complicated and involve more calculation. In addition, the method in *CBSOCP* requires the implementation of a maximum margin classifier. This is also one of the reasons that *CBSOCP* cannot be implemented as an online or real-time algorithm. On the other hand, the *DYNALINK* method is naturally designed to provide efficient and real-time link inference.

To further demonstrate that our proposed algorithm is highly efficient in terms of processing dynamic information networks, we also test the *online model maintenance efficiency* of the *DYNALINK* algorithm. For all three data sets, a new object is a newly published paper and inherently forms a small graph. Figures 10, 11 and 12 depict the processing rate of our algorithm when the three data set continuously receive new objects over time. The X-axis in the figures denoted the publication time of the corresponding objects for temporal identification. The processing rate is defined as the average number of new incoming edges that can be processed every second. Every time when a new object arrives, the *DYNALINK* algorithm is expected to determine or re-assign the cluster membership, update the node attributes, and maintain summary statistics as well as the structural similarity information. We also observe that the processing rate of the

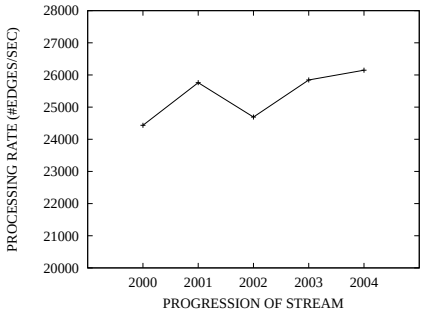

Figure 12: Efficiency on Data Stream (*Biochemistry*)

Genetics and *Biochemistry* data sets is relatively higher than that of the *DBLP* data set. This is due to the fact that each node in these data sets have fewer attributes than that in the *DBLP* graph. In all cases, several thousand edges are processed each second, and therefore the proposed algorithm is very efficient, and can be effectively used for dynamic and online scenarios.

4 Conclusions and Summary

In this paper, we presented an algorithm for dynamic link inference in temporal and heterogeneous networks. The algorithm is designed to be extremely efficient and is able to construct link inference models for online and heterogeneous networks which are continuously evolving over time. We achieve this goal with the use of a dynamic clustering approach in conjunction with content-based and structural models. Our experimental results show that our approach is able to achieve superior accuracy because of its more sophisticated approach. At the same time our method is extremely efficient, and can be made to work effectively for the case of data streams. In addition to being an online algorithm, it is also much more efficient than state-of-the-art methods for link prediction.

Acknowledgements

Research of the first author 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 authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Research of the second and third authors was supported in part by NSF through grants IIS 0905215, DBI-0960443, CNS-1115234, IIS-0914934, OISE-1129076, and OIA-0963278, and Google Mobile 2014 Program.

References

[1] L. Adamic and E. Adar, *Friends and neighbors on the web*, Social Networks, 25, (2001), pp. 211–230.

[2] S. F. Adafre and M. Rijke, *Discovering missing links in Wikipedia*, LinkKDD, (2005), pp. 90–97.

[3] C. Aggarwal. *Social Network Data Analytics*, Springer, (2011).

[4] C. Aggarwal and H. Wang, *Managing and Mining Graph Data*, Springer, (2010).

[5] M. Al-Hassan, V. Chaoji, S. Salem and M. J. Zaki, *Link prediction using supervised learning*, SDM Workshop on Link Analysis, Counter-terrorism and Security, (2006).

[6] M. Bilgic, G. Namata and L. Getoor, *Combining collective classification and link prediction*, ICDM Workshop on Mining Graphs and Complex Structures, (2007).

[7] J. Doppa, J. Yu, P. Tadepalli and L. Getoor, *Link mining: A survey*, SIGKDD Explorations, (2005), pp. 3–12.

[8] J. R. Doppa, J. Yu, P. Tadepalli and L. Getoor, *Chance constrained programs for link prediction*, NIPS Workshop on Analyzing Networks and Learning with Graphs, (2009).

[9] L. Getoor, N. Friedman, D. Koller and B. Taskar, *Learning probabilistic models of relational structure*, ICML, (2001), pp. 170–177.

[10] L. Getoor, N. Friedman, D. Koller and B. Taskar, *Learning probabilistic models of link structure*, Journal of Machine Learning Research, 3, (2002), pp. 679–707.

[11] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller and M. Wang, *A framework for semantic link discovery over relational data*, CIKM, (2009), pp. 1027-1036.

[12] H. Kashima and N. Abe, *A parameterized probabilistic model of network evolution for supervised link prediction*, ICDM, (2006), pp. 340–349.

[13] J. Kunegis and A. Lommatzsch, *Learning Spectral Graph Transformations for Link Prediction*, ICML, (2009), pp. 561–568.

[14] J. Leskovec, *Tutorial summary: Large social and information networks: opportunities for ML*, ICML, (2009), pp. 179.

[15] D. Liben-Nowell and J. Kleinberg, *The link prediction problem for social networks*, CIKM, (2003), pp. 556–559.

[16] M. E. J. Newman, *Clustering and preferential attachment in growing networks*, Physical Review Letters, 64, (2001).

[17] Y. Sun, R. Barber, M. Gupta, C. Aggarwal, J. Han. *Co-author Relationship Prediction in Heterogeneous Bibliographic Networks.* ASONAM, (2011).

[18] Y. Sun, J. Han, C. Aggarwal, N. Chawla. *When will it happen – Relationship Prediction in Heterogeneous Information Networks*, WSDM, (2012).

[19] B. Taskar, M. F. Wong, P. Abbeel and D. Koller, *Link prediction in relational data*, NIPS, (2003).

[20] C. Wang, V. Satuluri and S. Parthasarathy, *Local probabilistic models for link prediction*, ICDM, (2007), pp. 322–331.

[21] K. Yu, W. Chu, S. Yu, V. Tresp and Z. Xu, *Stochastic relational models for discriminative link prediction*, NIPS, (2006), pp. 1553–1560.

[22] J. Zhu, J. Hong and G. Hughesm, *Using Markov models for web site link prediction*, ACM Hypertext & Hypermedia Conf, (2002), pp. 169-170.