We can't solve problems by using the same kind of thinking we used when we created them.

– Albert Einstein, 1965.

University of Alberta

RANKING ENTITIES IN HETEROGENEOUS MULTIPLE RELATION SOCIAL NETWORKS USING RANDOM WALKS

by

Farzad Sangi

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2011
Edmonton, Alberta

To my parents

Ahmad Sangi and ***Safiyeh Daneshgar***

For their fully support and for always being there for me.

Abstract

In most machine learning and data mining tasks, data is typically assumed to be independent and identically distributed. In real applications, this assumption is not always correct. Data points are related to each other. Taking these relationships into account is a challenge but if done properly can provide new insights into the data. Data represented with its inner relationships is called *information networks*. A Social Network or Information Network is a structure made up of nodes representing entities, and edges representing the relationships among nodes. Understanding the behaviour of social networks is known as Social Network Analysis (SNA). SNA is used to study organizational relations, to analyse citation or computer mediated communications, etc. One of the most important applications of SNA is to find the similarity/relevance among entities in the network for a specific query. Finding the relevance between different entities, we are able to rank them based on each other. Ranking a set of entities with respect to one instance is required in many application domains. For example, in E-Advertisement, the goal is to show the most related advertisement to each user. This essentially means to rank the advertisements based on each user and to show the high ranked ones to the user. A researcher has a new idea in a particular topic. Wanting to publish the idea in the right place, an ordered list of conferences with respect to that particular topic is required. A person is eager to know the similar games to a favourite game, to prevent buying many irrelevant games. Taking a query, a search engine wants to rank all the web pages with respect to it. Consequently the user explore the most relevant pages rather than reviewing all the pages which only match with the keywords in the query.

In this study we focus on ranking the entities in heterogeneous multiple relation social networks, networks for which nodes belong to different classes and relationships have different types. We investigate social networks from bibliographic databases with authors, conferences and topics. Analysing such networks is a non-trivial task dealing with large k-partite graphs. We propose an algorithm to find the most related entities for each instance. We develop a tool called DB-Connect which applies our method on the academic social network.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Social networks are structures made of entities that are connected to each other by relationships representing some semantics. In fact, in every social network, entities interact with each other in one or more different ways. These interactions could be in any form such as: a friendship between two individuals in the network of students of a class, a link between two pages in the network of Internet web pages, a financial transaction between two companies in the network of companies of a country, and so on. One might think of a wide range of fields dealing with social networks. In fact wherever there are entities, there is a social network corresponding to them. Entities such as human-beings, virtual profiles, organizations, animals, web pages, etc.

Intuitively a social network is represented by a graph where nodes are the entities and edges are the relationships. As an example, the graph corresponding to the social network of the members of a karate club is depicted in Figure 1.1. The relationship between the members is the friendship relationship. Other information in the network could also be represented in the graphs. For example the significance of a relationship between two entities may be modeled as a numerical weight on the edge between the corresponding two nodes in the graph. In some cases, relationships may not be two-ways. Consequently instead of having undirected edges on the graph there are arrows showing the direction of the relationship between two entities.

Social networks have different categories. One might generally classify social networks based on the type of the entities and the relationships. Social networks are categorized in two groups of *Homogeneous* (i.e all entities are from the same class) and *Heterogeneous* (i.e entities have different types). Also, we may categorize social networks in two groups of *Unique Relation* (i.e all the relationships are from the same class) and *Multiple Relation*

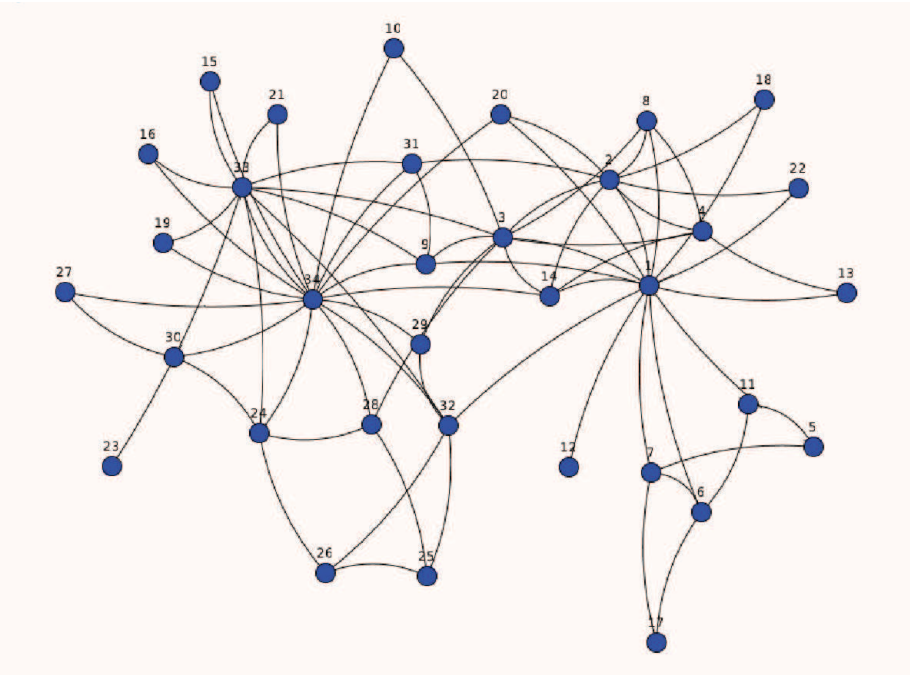


Figure 1.1: The members of a karate club and their friendship relations

(i.e relationships have different types). Furthermore, social networks exist in many fields such as (but not limited to): sociology, anthropology, psychology, marketing, economics, criminology, medical sciences, computer sciences. The broad usage of social networks makes the understanding of their properties and attributes important. *Social Network Analysis* (SNA) is the study of structural and behavioural properties of social networks. SNA aims to find useful information about social networks such as: the most influential node(s) in the network, the hubs in the network, the groups/communities in the network, and central nodes in the groups. SNA is also used to study the evolution of social networks (e.g detecting the formation, expansion, and changes in the communities within a social network). In addition, SNA may be used to provide an ordering for the entities of the social network.

In many applications, there is the need to rank the entities in the corresponding social network. This ranking may be as simple as sorting entities based on a specific attribute. For example, having the social network of employees within an organization, the manager wants to promote the employees based on their work experience in the organization. Consequently, the manager needs to rank them based on the period of time they have worked in the organization. However, the ranking task may be more sophisticated. For example assume that the manager wants to promote the employees based on their interactions with the clients. The more clients an employee serves and the more positive feedback the employee

gets from the clients, the more bonus the employee receives. This scenario requires a ranking of employees based on their interactions with the clients, which means based on the properties of the social network of employees and clients (e.g the number of connections of a node, the sum of weights of the connections).

In a more complicated scenario, when the organization offers a variety of services/products and targets a large number of clients regularly (e.g banks, chain supermarkets), the manager may want to know which clients usually use a specific service or buy a specific product to make some business decisions. This means ranking of entities (i.e clients) based on one specific entity (i.e service/product) in the social network.

Ranking the entities based on each other in social networks with different entity types and different relationships (i.e heterogeneous multiple relation social networks) is not trivial because analysing and understanding the properties of such social networks is problematic. Although ranking is not a new topic, not so many studies have been done on ranking entities in such social networks.

1.2 Thesis Statement

In this thesis, we investigate the feasibility of ranking entities in social networks with different types of entities and relationships. The main thesis statement of this research is presented as follows:

> ***Ranking entities with respect to each other in heterogeneous multiple relation social networks is possible***

Modeling heterogeneous multiple relation social networks as k-partite graphs, this work investigates a ranking approach based on random walks on bipartite graphs to propose a solution for the problem of ranking in such networks. We demonstrate the feasibility of ranking in such social networks theoretically and practically. More particularly, we claim the followings:

- It is possible to provide an analytical ranking of the entities in heterogeneous multiple relation social networks.
- It is possible to perform the ranking task for the entities of the social network in parallel.
- It is possible to perform the ranking task for reasonable portions of the social network, yet achieving acceptable results.

1.3 Thesis Contribution

The major contribution of this thesis can be summarized as follows:

1. Modeling heterogeneous multiple relation social networks with k-partite graphs, we propose an algorithm to rank all the nodes in the graph (i.e entities in the social network) based on one specific instance of the network. More specifically, using random walks on bipartite graphs, we assign relevance scores to the nodes of the graph with respect to the target instance. Therefore we are able to sort the entities based on their scores. We prove that the proposed ranking approach is theoretically scalable for larger values of k. Developing a framework called *DB-Connect*, we apply our ranking approach to a real heterogeneous multiple relation social network (i.e tripartite graph) and argue the accuracy of the outcome.

2. The proposed ranking approach only uses the information available in the social network. In other words, it computes the ranking for each instance separately without using the results of other entities. Furthermore, ranking of different entities can be done in parallel. Designing DB-Connect to be multi-threaded, we show the parallelized ranking in practice.

3. Performing our ranking approach on a reasonable portion of the real dataset and comparing the results with the outcome of the real-size dataset, we show that there is a negligible difference between the two results. This suggests that for large datasets, where there is not enough computation resources available, one may partition the dataset into smaller pieces and apply our ranking approach, yet achieve acceptable results. Note that the partitioning should be done in a reasonable manner (e.g clustering approaches)

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows:

- In Chapter 2, after introducing social networks and their different types, we discuss their broad applications. We elaborate the necessity of analysing social networks. Next we talk about ranking, one of the hot challenges in social networks. Discussing the significance of ranking, we explain the issues in this area.

- In Chapter 3, we review the related work to this study. Surveying Social Network Analysis (SNA) in the literature, we inspect different approaches in SNA toward ranking. Next, we explore the ranking approaches proposed in the context of social networks. Finally, we study the online systems which are working close to the goal of this thesis.

- In Chapter 4, first we explain the Random Walk with Restart (RWR) in bipartite graphs as the building block of our method. Second, we explain our solution for tripartite graphs. Next, we propose our ranking approach for k-partite graphs. Finally, we wrap up this chapter by suggesting some computational improvements for our approach.

- In Chapter 5, we describe a real world application for our ranking approach. Introducing our data source, we explain how we build the corresponding social network for this domain. Next, we introduce our framework, DB-Connect, which is developed to apply our ranking approach on the social network of this domain.

- In Chapter 6, we report the outcome of applying our ranking approach on the real dataset. First we review the way DB-Connect presents the ranking results for different entity types (i.e how the user should interact with the system). Next, we discuss the quality of the ranking results for few selected entities (i.e one instance of each entity type). Explaining the capability of our ranking approach in computing the rankings in parallel, we describe how DB-Connect is deployed on WestGrid to use the High-Performance Computation facilities for the ranking task. We also compare the ranking results from the whole dataset with the similar results from a reasonable portion of the database. Finally, we finish this chapter by a blind comparison between the old prototype version of DB-Connect, which uses a different method for ranking, and the new one.

- In Chapter 7, we present a brief summary of this dissertation. We review the contributions of this study. We wrap-up the chapter by exploring unresolved challenges.

Chapter 2

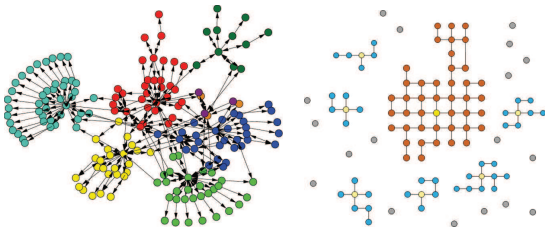
Social Network Analysis and Ranking in Graphs

2.1 Social Network Analysis

Social networks are structures made of entities that are connected to each other by relationships representing some semantics. *Entities* are tied to one another via *relationships* such as (but not limited to): friendship, common interest, financial exchange, dislike, WWW hyperlink, physical connection, the presence in the same place. One might think of a wide range of fields which deal with social networks. In fact, wherever there are individuals, there is a social network corresponding to them. Individuals are not only human-beings but they can be any entity such as profiles, companies, animals, web pages, etc. For instance, a network of people and their relationships in the real life (e.g Facebook) can be modeled via a social network so can a network of electric pieces on a board and their connections. Figure 2.1 shows some examples of different social networks. Figure 2.1 (a) corresponds to a web domain with several sub-domains. Nodes of the same colour represent web pages in the same sub-domain. Directed edges show the links between pages. Figure 2.1 (b)[1] depicts *Yahoo! 360*'s users. As seen in the image there is a large group of people at the centre who are strongly connected to each other. There are also several groups consisting of a few number of people who are connected to each other but not to the rest of the network. Finally there are people who are isolated from the network. Figure 2.1 (c)[2] represents the email flow network among people working in a large project. People from different departments are shown with nodes with different colours except gray nodes which stand for people who are not formal team members. Each edge represents an email exchange between two people.

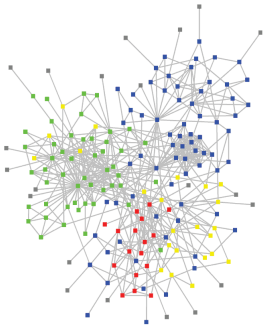
Intuitively a social network is represented by a graph where nodes are the entities and

[1] http://www.boxesandarrows.com/view/social-networks
[2] http://www.orgnet.com/email.html



(a) Web Site

(b) Users of *Yahoo! 360* and their connections



(c) Email flows among a large project team

Figure 2.1: Examples of different social networks

edges are the relationships. Other information in the network could also be represented in the graph. Relationships may not be always two-way. Consequently instead of having undirected edges on the graph there are arrows showing the direction of the relationship between two entities. Figure 2.2 shows a part of the internet graph where pages are nodes and the hyperlinks between them are the edges. As seen in the image, the links are directed because of the nature of hyperlink in the web. Sometimes the relationships have different levels of importance. This means while the relationships are from the same class, some of them are stronger than others. In these cases, usually numerical weights are used on the edges of the network. As an example, consider Figure 2.3 which represents the database of a grocery store. Each customer has one or more transactions in the database. If customer c purchases product p, there is a relationship between c and p. Based on the weights, we observe that the relationship between *Customer B* and *Orange* is 4 times stronger than the relationship between *Customer C* and *Orange*.

Social Network Analysis (SNA) is the study of structural and behavioural properties of social networks. SNA is used to construct the social network for different problems. It also answers questions about social networks such as: Which node is the most influential node?

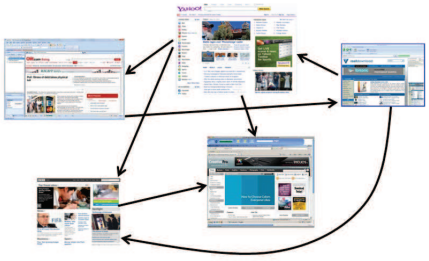


Figure 2.2: An example of a directed graph: Internet

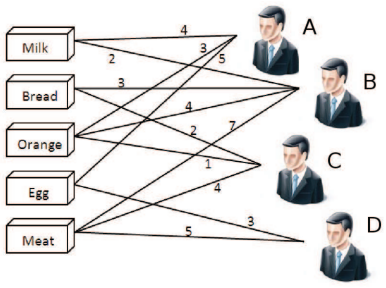


Figure 2.3: An example of a weighted graph: Customer-Product, weights are the frequency of buying an item

What are the hubs? What are the groups? Which node is the most central in a group? SNA is used to predict the future of social networks (usually predicting the new relationships among entities). SNA discovers the communities and their evolution during the time in social networks [30] . In addition, it finds ways to approximate large social networks.

Social Network Analysis has been used as a key technique in many fields such as sociology, anthropology, epidemiology, psychology, etc. In the past few years, Social Network Analysis has also gained a significant role in marketing, economics, criminology, medical sciences, computer sciences.

Marketing: Facts of social networks such as most influential nodes, growing groups, central nodes in groups, and so on, are the key concepts in marketing. SNA is a powerful tool for companies to do a better marketing for their products/services. Figure 2.4 depicts that US companies spend more and more budget on the social network marketing.

Criminology: SNA helps police and other law enforcement decision-makers to achieve an awareness situation and enables them to plan and execute their actions in a pro-active

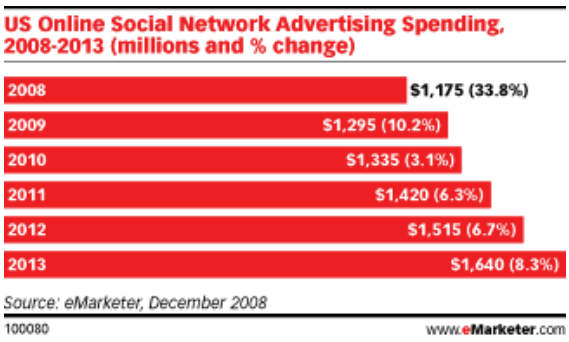


Figure 2.4: US online social network advertising spending [52]

manner. [53], [54], [47].

Medicine: SNA provides doctors and researchers valuable information of how an epidemic disease spreads through its network.

Recommendation Systems and Relevance Ranking: SNA methods have been successfully investigated to analyze the commercial social networks (e.g www.amazon.com). Online stores want their customers to 1) find what they need 2) see what they might need or want. For the former case, they develop user-friendly search interfaces. For the latter case, they want to suggest products which are similar to or related to the searched items. Finding similarity/relevance among entities plays the key role in product recommendation.

2.2 Types of social networks

Since social networks are used in different application domains, there is a variety of types based on the problem to which they are applied. Nodes or edges could be from the same class or multiple classes. Any of these will change the structure and consequently the behaviour of the social network.

2.2.1 Homogeneous Networks

A social network is homogeneous when the nodes of the network are from the same class. This means there is only one type of entity in the domain so all the nodes are representatives from the same class. For example a dating network for high school students: the only entity type in this network is *Student*. Figure 2.5 shows the corresponding graph for this social network.

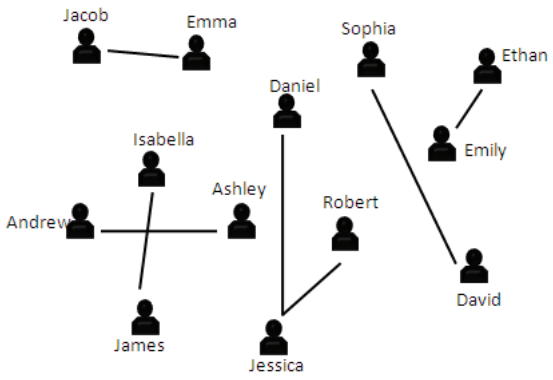


Figure 2.5: An example of a Homogeneous social network: The dating network within a high school

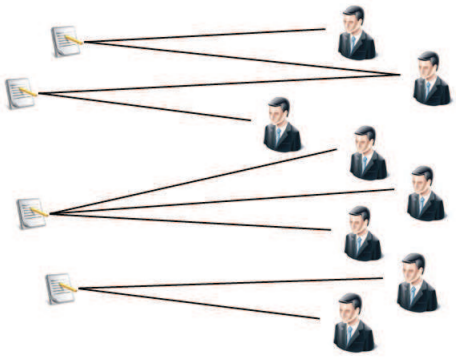


Figure 2.6: An example of a Heterogeneous graph: The publication network of a conference

2.2.2 Heterogeneous Networks

A social network is heterogeneous when there is more than one type of entity in the network. In this type of social networks, each entity type has its own attributes but they are related to other types of entities. Considering a conference, two entities are recognized: Paper and Author. Each *paper* has attributes such as: *title, conference, year*, etc. Each *author* has also its own attributes: *first name, last name, affiliation, birth date*, etc. The relationship between these two entities is: an author a *Publishes* a paper p. Figure 2.6 shows the corresponding graph of a social network for a sample conference. Two types of entities are represented by different icons. As seen in the image, each paper has been published by one or more authors. In some cases an author could participate in more than just one paper in the conference. Note that the structure of this network is a well-known structure called "bipartite" graph. This means there are two parts in the graph (each corresponds to

one class). More importantly the relationships are only between these two parts. It means the entities from one part are only related to the entities from the other part (if there is any connection). In other words, all the relationships are inter-part relationships and there is no intra-part relationship.

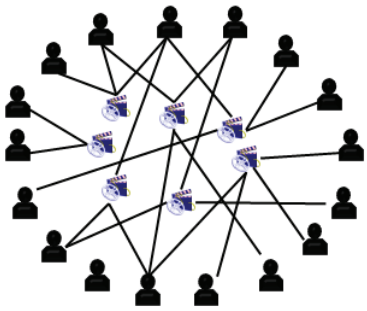


Figure 2.7: An example of a unique relation graph: The acting network for movies

2.2.3 Unique Relation Networks

When the edges of the network are of the same kind, the social network is called a unique relation network. This means there is only one type of relationship in the network. For example consider a network of *actors* and *movies*. The only relationship between entities of this network is *acting*. Figure 2.7 shows the corresponding graph for this social network.

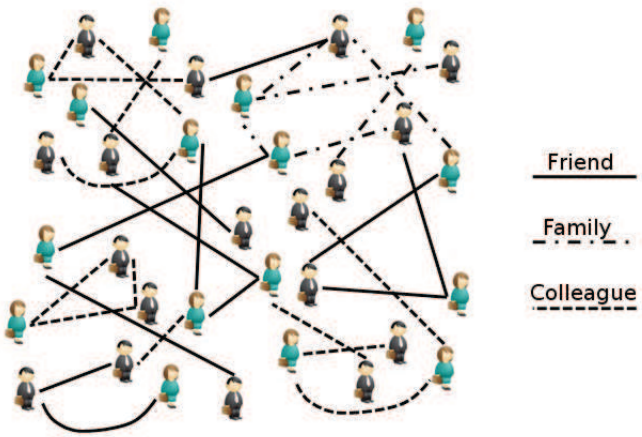


Figure 2.8: An example of a multiple relation graph: The network of people in a town

2.2.4 Multiple Relation Networks

When the edges of the network are from different classes, the social network is called a multiple relation network. In these social networks, there are different relationships among the entities. For example in a network of people, there are relationships such as: Friend, Family, Colleagues. Clearly, they could be interpreted separately for different purposes in SNA. Figure 2.8 depicts an example of such a social network. As shown in the legend, the solid line is used to show if two people are friends, dashed line is used for colleagues, and the dot-dashed line is used for family.

Table 2.1 provides the summary of our discussion about types of social networks.

	Homogeneous	**Heterogeneous**
Unique Relation	All nodes from one class, edges have one semantic	nodes from different classes, edges have on semantic
Multiple Relation	All nodes from one class, edges have different semantics	nodes from different classes, edges have different semantics

Table 2.1: Types of Social Networks

2.3 Ranking in Social Networks

Ranking means to obtain a meaningful order for a set of entities. Ranking is always based on some criteria. The criterion is usually one or more attribute(s) of the entities. Almost all the application domains require/use ranking such as:

- **Biology**: one might rank a collection of blood samples based on their hemogolobin.
- **Astronomy**: one might rank the known galaxies based on the light-year distance from the earth
- **Geology**: one might rank a collection of fossils based on their archaeological period
- **Economics**: one might rank IT companies based on their stock value
- **Education**: one might rank the graduate schools in Canada based on their performance in a specific program
- **History**: one might rank wars occurred in Europe during the 20th centry based on the estimated damage to the economics

The simplest ranking is to sort similar objects based on one attribute. For example, sorting the students of a class based on their GPA or based on their family names. In either case, the problem of ranking is downgraded to finding a *comparison function* which works between any pair of entities based on the target attribute. A comparison function F takes entities e_1 and e_2 and attribute a as inputs and determines the superior entity based on attribute a. Using a comparison function, we are able to sort the entities with a sort algorithm such as: Merge Sort, Quick Sort, Bubble Sort, etc. As long as the ranking is based on a quantitative attribute (i.e an attribute which is measured as a number), the procedure is straightforward. The comparison function works with arithmetic rules. However, if the attribute is categorical, the ranking is not trivial. For example, ranking the students of a class based on their favourite sport. In this case it is not clear how the ranking function should choose between a pair of attributes. Soccer or basketball? Swimming or volleyball?

Sometimes the attribute for ranking does not belong to the entities but the relationship among them. That means the entities and the relationships among them constitute a social network and the ranking is based on the properties of that network. For example, finding the most central entity in a social network. Furthermore, SNA approaches have been investigated to rank entities in different social networks. We will explain this concept in Chapter 3.

To make it more complicated, the criteria for ranking may not be an attribute (related to the entity or to the network) but an object. Assume that you have watched a movie and you liked it. You are prompted to know what movies are close to that one to watch them as well. This means you are asking for a ranking of entities (i.e movies) based on one instance of them (i.e the movie you liked). In other words, you are interested to find the most similar/related entities to an instance in a social network. Ranking a set of entities with respect to one instance is required in many application domains. For example, in E-Advertisement, the goal is to show the most related advertisement to each user. This essentially means to rank the advertisements based on each user and to show the high ranked ones to the user. A researcher has a new idea in a particular topic. Wanting to publish the idea in the right place, an ordered list of conferences with respect to that particular topic is required. A person is eager to know the similar games to a favourite game, to prevent buying many irrelevant games. Taking a query, a search engine wants to rank all the web pages with respect to it. Consequently the user explores the most relevant pages rather than reviewing all the pages which only match with the keywords in the query.

Ranking entities based on each other is not trivial. There are two major issues 1) The comparison function does not deal with one attribute but the whole entity. Thus it is more complicated because the comparison function must deal with all the attributes of the entity (both quantitative and categorical) at the same time 2) Even having a comparison function which prefers one entity out of two, does not solve the problem. Because it only provides a general ordering for all the entities while an ordering with respect to one instance is required. A possible solution for this problem is a *relevance function*. The relevance function R takes a target entity e_1 and another entity e_2 as the input and assigns a numerical score to e_2 with respect to e_1. Applying a relevance function where the target entity is fixed and all other entities are used as the second parameter, a quantitative measure is available to sort entities. Note that this order is specific to the target entity and may change by using another instance as the target entity. It is clear that constructing such a relevance function is not easy. Especially when e_1 and e_2 are from different types. In fact, one cannot assign a relevance score between two entities only by observing the entities themselves. Instead one should examine the interaction between them. While this interaction depends on the direct relationship between those two entities, it is also affected by the relationship between other entities in the network. Furthermore in a complex network, with different types of entities and relationships, ranking entities based on particular instances of the network is not trivial.

In this chapter, we introduced social networks. Presenting different types of social networks by examples, we discussed why social networks are important. We also introduced social network analysis and its applications. Moreover, we discussed the concept of ranking in social networks. In addition, the importance and challenges of ranking in social networks are explored. In the next chapter, we review related works to this study.

Chapter 3

Related Work

Social networks are information networks that are represented by graphs and depict the interactions between individuals (entities, objects). In those graphs, each entity is represented by a node and there is an edge between two nodes if an interaction has occurred, or a relationship exists between the two entities. For example, the exchange of ideas, information, and experiences between people in the web can be modeled as a social network. The network of researchers who publish in conferences or journals is another example.

Finding the elements of social networks, objects and relationships, is not always trivial. In fact in some domains, detecting objects in various data types and finding their associations is a challenge. As an instance, Dong et al. [15] propose methods to find associations in unstructured data. For example, anyone who has a desktop wants to browse his personal information in a meaningful manner. It has been observed that human beings do not think of data in the way that is stored in the computers [15]. Instead of having a directory hierarchy of files (normal file systems), we tend to look for objects and their associations to each other. The same authors introduce SEMEX (SEMantic EXplorer): a personal information management tool [9]. SEMEX extracts objects and their associations from different data sources such as: Word, Excel, PowerPoint, Bibtex, Latex, PDF, Contact, Email. The key challenge in SEMEX is the fact that it has to support a variety of mechanisms to extract the objects and associations. Although in many cases the objects and associations are explicitly defined in data sources (e.g a contact list contains persons, an email message contains fields indicating the sender and the receiver), there are other sources that need special treatment. Considerable number of objects and associations are extracted by analyzing specific file formats. For example, parsing a Latex file or a PowerPoint presentation provides the authors. Combining the information from Bibtex and Latex subsidizes the citations. Furthermore SEMEX makes the objects (e.g Person, Publication, and Message) and associations (e.g

AuthoredBy, Cites and AttachedTo). Having an object-association structure built, SEMEX allows its users to browse their data by following objects or associations. It also provides keyword search on the domain model. Searching for a keyword, it returns not only the direct match objects (the ones which have the keyword in their attributes) but also objects that are strongly related to multiple such objects.

In many other application domains, social networks are well structured. Having the entities and their relationships explicitly defined, one could investigate different properties of entities and analyze the interactions among them to answer critical questions such as (but not limited to): Which entity is the most influential one in the network? How does an entity interact with others in the network? What are the groups and communities in the network? How important is an entity in a group? Which entities have the least interactions with the rest of the network and why? What are the most related entities with respect to a specific node in the network? These questions are addressed in two realms: *Social Network Analysis* and *Ranking Approaches*. In this chapter, in addition to covering significant studies in these fields, we discuss several online systems which provide practical solutions for these problems.

3.1 Social Network Analysis

The analysis of social networks is of interest to many fields such as sociology [51], epidemiology [32], recommendation systems [39], email communication [50], criminology [10], advertisement, etc. Social network analysis includes challenges such as: building the social network graph for a domain or dataset and browsing the data by its inner entities or relationships, detecting the most important/influential entities in the network, investigating groups and communities: formation and evolution, finding common patterns to understand the behaviour of social networks to predict the future, approximating large social networks.

In social network analysis, several metrics have been defined to detect influential entities. The most popular metrics are the centrality measures [19]. These measures include *degree centrality* and *betweenness centrality* and *closeness centrality*. Let $G = (V, E)$ be an undirected graph with the set of vertices V and the set of edges E.

Degree Centrality: the number of edges attached to a node is the degree centrality of that node. The more connections a node has, the more popular the node might be. Equation 3.1 shows the normalized degree centrality of node $v_1 \in V$:

$$C_D(v_1) = \frac{deg(v_1)}{n-1} \tag{3.1}$$

where n is the number of nodes in the graph and $deg(v)$ is the number of edges attached to node v.

Betweenness Centrality: the number of shortest paths passing through a node is the betweenness centrality for that node. The greater the numbers of shortest paths going through a node, the more control that node has over the flow of information in the network. Equation 3.2 shows the normalized betweenness centrality of node $v_2 \in V$.

$$C_B(v_2) = \frac{f_g(v_2)}{f_g} \tag{3.2}$$

where f_g is the number of all shortest paths in G while $f_g(v_2)$ represents the number of shortest paths that go through v_2.

Closeness Centrality: the average shortest path between a node to all other reachable nodes is the closeness centrality for that node. When the total *geodesic distance* (i.e shortest path) of a selected node and all other reachable nodes is small, that node has a high closeness centrality value and it is more central in the network. Equation 3.3 shows a normalized version of closeness centrality for node $v_3 \in V$:

$$C_C(v_3) = \frac{n-1}{\sum_{i=1}^{n} d(v_3, v_i)} \tag{3.3}$$

where n is the number of nodes in the graph (so $n-1$ is the minimum total distance from v_3 to all other nodes) and $d(v_3, v_i)$ represents the geodesic distance between node v_3 and v_i.

Eigenvector Centrality: This centrality measure was introduced later by Philip Bonacich [3]. Let A be the adjacency matrix for G where $a_{ij} = 1$ if there is an edge between nodes i and j, and $a_{ij} = 0$ if these nodes are not connected. Because G is undirected, A is symmetric; thus all its eigenvalues are real and its eigenvectors are orthogonal [22]. Bonacich suggests that the eigenvector corresponding to the largest eigenvalue of an adjacency matrix could make a good network centrality measure. "The centrality of a vertex is proportional to the sum of the centralities of the vertices to which it is connected" [4]. This recursive definition shows that instead of only considering the direct neighbours, the eigenvector centrality takes the whole network into account. Eigenvector centrality can be represented as a

sum.

$$Ax = \lambda x, \qquad \lambda x_i = \sum_{j=1}^{n} a_{ij} x_j, \;\; i = 1, 2, ..., n$$

where λ is the largest eigenvalue of A and n is the number of nodes in the graph.

In addition to finding central individuals in the network, researchers have tried to find similar/relevant nodes in social networks. The concept of similarity between entities, have been studied in the literature for years. Traditional methods use the set of attributes of each entity to find similarities. These attributes make a vector for the entity. The more similar the vectors of two entities, the more similar they are to each other. For example, Keywords in a Document could make the attribute vector of that document. The closer the vectors are in the attribute space, the more similar they are to each other. It means document A is more similar to document B than C, if A shares more keywords with B than C.

There have been many efforts to improve the traditional approaches on finding similarities between entities. Ganesan et al. [20] exploit the concept of hierarchy in the domain of attributes. Using the hierarchy makes the comparison between two attributes more accurate. In traditional methods, two elements in the attribute vector are either equal or not equal but in the domain hierarchy there is a range of closeness between two elements: the distance of those two elements in the hierarchy graph. This approach certainly helps improving the similarity detection between entities which do not share the exact attribute values but do have attributes from the same type in the hierarchy. The main drawback is that finding the domain hierarchy could be difficult or even infeasible in some cases. Also the entities are not always from the same class. Thus the direct comparison of attributes is meaningless.

Another approach to find relevant nodes in a network is to take the domain of attributes into account. Instead of matching attribute values of entities, in most of the cases we could consider the attributes as instances of other type of entities. Recall the document matching example where *keywords* were attributes for *documents*, keywords themselves could be instances of an entity. Thus we have two entities: Document, Keyword; and there is a multiple relationship between these two as each document could contain multiple keywords, and any keyword could be in multiple documents(i.e a social network). SNA methods have been developed to investigate these relationships in social networks to find similar/relevant entities.

Proximity or *similarity* is tied to the definition of an edge in the network. For instance, while edges represent phone or email communication, proximity measures the potential information exchange between two non-linked nodes through other nodes. Proximity can

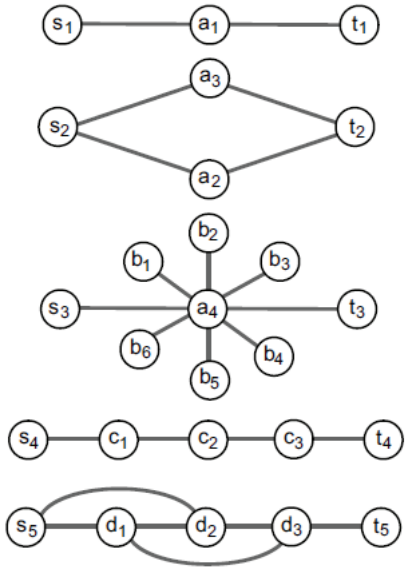


Figure 3.1: Sample graphs with all edge weights equal 1 [28]

also measure the likelihood of two nodes to belong to the same group. In addition, proximity estimates the link existence in the future. For example, if two people speak on the phone to many common friends, the probability is high that they will talk to each other in the future.

There are several approaches to measure the closeness of two nodes in social networks. An intuitive candidate is the length of the shortest path between two nodes. In weighted graphs this would be the sum of weights of edges on the shortest path. Figure 3.1 shows why this is not a good candidate as the length of the shortest path between s_i and t_i $(i = 1, 2, 3)$ is 2, yet we have different conclusions about their proximity. s_2 is closer to t_2 than s_1 to t_1 because they have more friends in common, while s_3 and t_3 are not probably close because they are only connected through a large degree node (e.g two web pages that link to www.RedCross.org for Haiti earthquake donations).

Another candidate is the maximal network flow: maximal number of units that can be simultaneously delivered from one node to another. Although this definition favours (s_2, t_2) over (s_1, t_1) in Figure 3.1 because we can deliver twice as many units between s_2 and t_2, there is no notion of length in this definition. We get the same proximity between $s_4 - t_4$ and $s_1 - t_1$. In the other hand, the maximal flow between $s_5 - t_5$ is the same as $s_4 - t_4$ even though there are more paths between s_5 and t_5.

The next candidate comes from modeling the graph as an electric circuit where edges are resistors whose conductance is their weights. *Effective Conductance* (EC) favours shorter

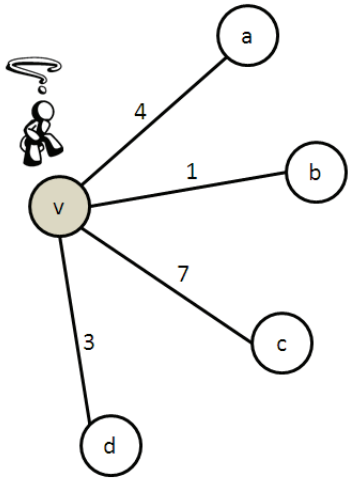


Figure 3.2: A part of a weighted graph: How a random walker chose the next edge in each state

paths and the more number of alternative paths. The formal definition of the effective conductance is $EC(s,t) = deg(s).P_{esc}(s \rightarrow t) = deg(t).P_{esc}(t \rightarrow s)$ where P_{esc} is the escape probability. In fact $EC(s,t)$ is the expected number of "successful escapes" from s to t where the number of attempts equals the degree of s. Furthermore, EC has an intuitive interpretation based on *random walks*.

A "random walk" on a weighted graph $G = (V, E)$ is a sequence of nodes in V which an agent visits by taking random edges. The agent starts from a specific node, then selects one of the out-going edges randomly with respect to their weights. This process continues iteratively in each node. For example, if the agent is on node v in Figure 3.2, either of its neighbours, nodes a, b, c, d, may be chosen with the probability of $\frac{4}{15}, \frac{1}{15}, \frac{7}{15}, \frac{3}{15}$ respectively. The agent keeps traversing until a certain condition is reached (e.g. after a certain number of times, having no change in the state probability of the nodes, etc). The relevance between two nodes in the graph can be defined by a random walk from one to another. The relevance of node t with respect to node s is the steady-state probability of the random agent visiting node t when it starts from node s.

The effectiveness of random walks in finding "importance" scores for nodes or ranking in general is proven especially after Google [8] successfully applied it on the web to rank search query results.

Another property of effective conductance is that it is monotonic. That means increasing the conductance of any resistor (i.e. increasing an edge weight), or adding a new resistor

(i.e. adding an edge) to the network, can only increase the conductance between any two nodes. While this property is desirable in some cases (e.g. in Figure 3.1 $EC(s_5, t_5) > EC(s_4, t_4)$ and $EC(s_2, t_2) > EC(s_1, t_1)$ which is intuitive), sometimes it contradicts the notion of proximity. An example of this situation is presented in Figure 3.1. As mentioned before, s_3 and t_3 are not close because they are only related through a high degree node.

Faloutsos et al. [16] introduce *Sink-Augmented Effective Conductance* to solve the problem of monotonicity by connecting each node to a universal sink. The authors also explain how to extract a subgraph from the main original one, so that all the important information about node N_1 and N_2 is absorbed in that subgraph. This resolves the scalability problem. Although their approach solves some issues, it completely destroys monotonicity. As the graph becomes larger if we add more links between node s and t, the proximity decreases because the delivered current from s to t is negligible compare to the current that goes to the universal sink. In addition, this method needs a parameter k which is the size of subgraph to extract from the original one. Finding this parameter is not trivial and affects the proximity between nodes.

Inspired by the above study, Koren et al. [28] propose *Cycle Free Effective Conductance* (CFEC) to measure the proximity of two nodes in a social network. CFEC solves the problem of monotonicity. The key idea is to avoid visited nodes in computing the effective conductance.

In the aforementioned approaches ([16],[28]) all the relationships are from the same type. Also nodes are from the same class. This means these approaches do not apply to heterogeneous social networks or multiple relation networks. That is why the corresponding graph could be imagined as an electric circuit.

3.2 Ranking Approaches

Page et al. [38] introduce PageRank, a method primarily to rank Web pages. PageRank exploits the link structure of hypertext in Web pages which makes it more powerful than any other method that indexes the text. PageRank is the essence of Google [8], a powerful search engine which has gained popularity since its appearance.

PageRank assigns numeric values to rank Web pages or as the writers of the paper quote: “Bringing Order to the Web”. In fact these numeric values provide a notion of importance for each web page. The key idea is that an important page is being linked from important pages or many other pages. In other words, the citations of a page show its significance.

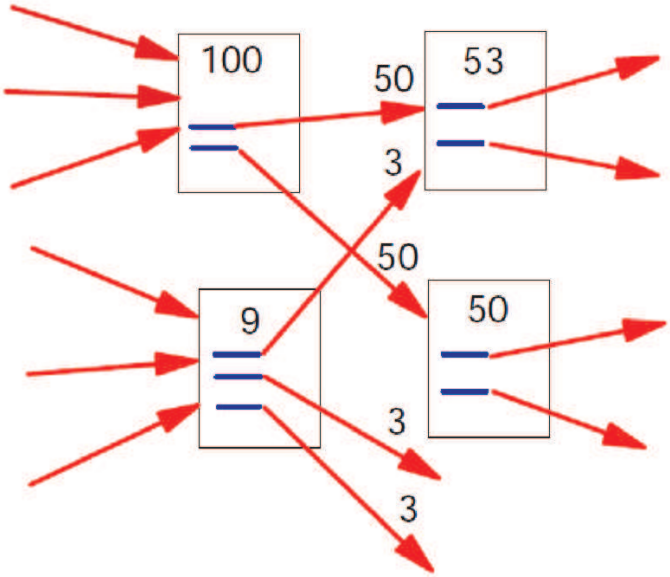


Figure 3.3: Simplified PageRank Calculation [38]

Note that the idea of using the citations as a factor of importance was not new at the time. For example, [21] demonstrates the effectiveness of citation analysis as a useful measure of objectivity into the evaluation process.

The simplified formula for computing PageRank of page u is:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} \tag{3.4}$$

Where B_u is the set of pages that point to u, N_v is the number of links from v to other pages, and c is a factor to normalize the rank scores. Figure 3.3 presents the PageRank calculation for a pair of pages. The numbers on the pages indicate the rank for them. The page with rank 100 propagates its rank to the pages that it points (i.e 100 splits into two for each hyperlink), similarly the page with rank 9 splits its value to the outgoing links, each takes 3.

It has been shown that this approach converges to a steady state except when there is a *rank sink*. A "rank sink" is a loop which does not point to any other page outside of it, but it has at least one incoming link. An example of rank sink is shown in Figure 3.4. Since the rank values flow in the directed graph, when there is a rank sink, the loop accumulates rank but never distributes any rank.

To solve this problem, the authors suggest the concept of *rank source*. Let E be a

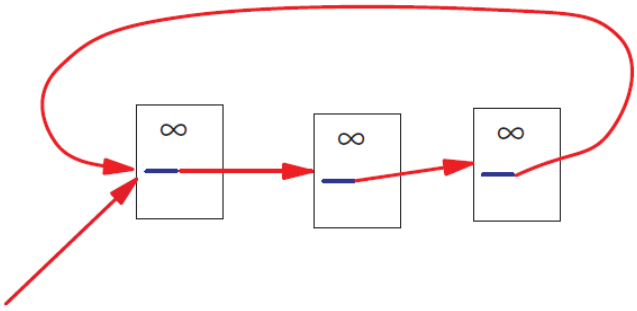


Figure 3.4: Rank Sink loop which causes the problem[38]

vector of pages that corresponds to a source of rank. Then the definition of PageRank in Equation 3.4 changes to the Equation 3.5.

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u) \tag{3.5}$$

such that c is maximized and $|| R' ||= 1$.

The definition of PageRank is equivalent to the random walks on graphs. The simplified version corresponds to the state probability distribution of a random surfer on the graph of Web. The *rank sink* emulates the cases when the random surfer is bored and jumps into a random page. This solves the problem of surfer getting stuck in a loop, because the surfer eventually gets bored and continues from another random page.

PageRank orders entities of the network in a meaningful manner, however, in many applications this is not enough. One might ask for an ordered list of entities with respect to a particular instance or query. As an example, in the context of searching in the World Wide Web, the user enters a query and wants to see the pages relevant to it. Normally this need is satisfied by looking into all web pages and collecting the ones with the keywords in the given query. Nevertheless, if the query is general enough, there are thousands of thousands we pages containing those keywords (A.K.A. *broad-topic queries* problem). For instance finding information about the Java programming language is considered a broad-topic query. Dealing with broad-topic queries, the main obstacle is that the number of relevant pages, the ones which contain the keywords, is too large to be reviewed by a human user. Kleinberg [27] proposes a solution which tries to find the most *authoritative* and *definitive* pages among all the relevant pages. In the other hand, the authority pages about query q usually do not contain the keywords in q. For example, if someone wants to know

the URL of a search engine and enters the query "search engines", he would not get any good results because search engines do not usually include this keyword in their home page. In another example, *www.harvard.edu* should be returned as the most authoritative page for the query "Harvard" but there is no way for a pure text-based search engine to favour this page over thousands of other pages which have the keyword "Harvard" in them. Thus the goal is to focus on a collection of pages which is:

- 1) Small
- 2) Rich in relevant pages
- 3) Contains many of the strongest authorities in that topic

To achieve conditions 1 and 2, the authors get the top t results of the query topic q on a text-based search engine (e.g. AltaVista). The graph of these t pages, R_σ, does not usually include the authority pages of q. Moreover, it is most likely for an authority page to be pointed by a page in R_σ. So all the pages that are being pointed by at least one page in R_σ are added to achieve S_σ. While S_σ satisfies the third condition, it is not too large and it is rich in relevant pages.

To actually find the authorities of q in S_σ, the authors sort pages in S_σ based on their in-degree (the number of links that point to them). The issue here is that there are universally popular authorities which no matter what q is, they are in S_σ. Furthermore they introduce the concept of "hub" pages. Authority pages relevant to q, not only have a large in-degree, but share pages who point to them, since they are about the same topic. These pages that point to many authorities in q are called hubs. "A good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs."[27]

Defining authority-weight and hub-weight for each page p, Kleinberg proposes an iterative algorithm called "HITS" to find the authorities and hubs in query topic q. If p points to many pages with large authority-weights, then it should receive a large hub-weight; and if p is pointed to by many pages with large hub-weights, then it should receive a large authority-weight.

Jeh et al. [26] propose SimRank: a recursive algorithm on directed graphs to compute the similarity between pairs of nodes based on the structural context of the network. SimRank uses the entity-to-entity relationship and builds a graph for the social network where entities are the nodes of the graph and the relationships represented as the edges. The key idea is that two entities are similar to each other if they are related to similar entities. Let

$s(a,b) \in [0,1]$ be the SimRank similarity between objects a and b. If $a = b$ then $s(a,b) = 1$ otherwise it is computed as in Equation 3.6.

$$s(a,b) = \frac{C}{\mid I(a) \mid\mid I(b) \mid} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \tag{3.6}$$

where C is a constant (confidence level or decay level), $I(a)/I(b)$ represents the set of in-neighbours for node a/b. Note that if either a or b has no in-neighbours, the similarity is defined to be 0.

In fact, to compute the similarity between two nodes, SimRank adds the similarity between all the in-neighbour pairs of those two nodes. In other words, the similarity flows between pairs of nodes. Furthermore, if we build another graph which its nodes are ordered pairs of the original graph (e.g if G is the original graph with nodes $a, b, c, d, ...$ then G^2 contains nodes such as $(a,a), (a,b), (b,d), (a,c), ...$), the similarity flows between the nodes. Similarity flows from node (x,y) to (z,w) in G^2 when node x is related to z, and y is related to w in G. Since it is assumed each entity is similar to itself (i.e $s(x,x) = 1$), the sources of similarities in G^2 are $(a,a), (b,b), (c,c), ...$. An intuitive model for SimRank is based on random walks. In fact SimRank similarity score, $s(a,b)$, shows how soon two random walker will meet at the same node if they start at node a and b. Another interesting fact about SimRank is that even though SimRank is defined on directed graphs, the similarity score assigned to objects is symmetric based on Equation 3.6 (i.e $s(a,b) = s(b,a)$).

To compute SimRank, the authors suggest using an iterative approach. Let $R_k(a,b)$ represent the score between a and b on iteration k. Defining $R_0(a,b)$ as:

$$R_0(a,b) = \begin{cases} 0 & (if \quad a \neq b) \\ 1 & (if \quad a = b) \end{cases}$$

The score on the k^th iteration is computed as shown in Equation 3.7.

$$R_{k+1}(a,b) = \frac{C}{\mid I(a) \mid\mid I(b) \mid} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) \tag{3.7}$$

as the number of iterations becomes larger, $R_k(a,b)$ converges to $s(a,b)$.

The authors also define SimRank for bipartite graphs to compute similar nodes in each partition. The intuition is that the instances of A in a bipartite graph of A and B, are similar to each other if they have similar neighbours in B, and vice versa.

The main drawbacks of the reviewed ranking approaches are scalability and efficiency issues. Also the weights of the edges are not taken into account. Assuming the same level

of importance among all entities is not consistent with the real world problems. Also, these approaches are only applied to homogeneous social networks.

In this study, not only the weights of edges (i.e levels of relationship between two entities) are taken into account for finding the ranking, there are more than one types of entities and relationship in the network. In addition, we find the ranking of entities with regard to any specific instance in the network. This is different from finding similar nodes. In other words, node v_1 may be top ranked with respect to node v_2, however, node v_2 may not be ranked high with respect to node v_1.

3.3 Online Systems

In the past few years, as the concept of ranking entities in different domains has become a hot topic, a few teams developed tools to provide online services for this purpose. All these systems tend to bring some kind of order for the entities of their corresponding social network. The most common social network used in these systems is the academic social network. This is more likely due to the large number of public data sources and datasets containing the bibliographic and academic information. For example, *CiteSeer*[1] is a digital library which lets the users search in scientific literature primarily in computer and information sciences. *ACM DL*[2] is another digital library containing all the ACM publication and bibliographic citations from major publishers in computing sciences. Another major source of information for computer sciences social network is *DBLP*[3] which stands for Digital Bibliography and Library Project. This website lists the computer science journal and conference articles.

In this section we present the most popular frameworks which provide services on academic social networks.

3.3.1 ArnetMiner: Academic Researcher Social Network Search

Tang et al. [48] have built a comprehensive researcher social network called "ArnetMiner" which provides search and mining services. ArnetMiner crawls the web and finds the homepage of researchers by using a binary SVM classifier. Applying Conditional Random Fields (CRF) tagging model, it extracts the profile information for researchers from these pages. At the same time it collects the publication information from digital libraries (e.g CiteSeer,

[1] http://citeseerx.ist.psu.edu
[2] http://portal.acm.org/
[3] http://www.informatik.uni-trier.de/ ley/db

ACM digital Library, DBLP) using explicitly defined rules. Finally it uses the "names" to integrate the publication information into the profile information. Using names as the identifiers creates many name collisions. Therefore they have proposed a probabilistic framework to deal with the name ambiguity. The integrated data is stored in a MySQL database.

ArnetMiner is one of the first online services which may be used for researcher interest finding and academic suggestions. Tang et al. propose three topic models to simultaneously model the topical aspects of different entities (e.g authors, publication venues, publications). In fact the authors try to find a topic distribution for each entity type. One advantage of the topic models is that ArnetMiner may be used to estimate the relevance between two different types of entities (e.g author, conference) based on their topic distribution. Figure 3.5 shows a screenshot from ArnetMiner website. The basic information for the given researcher (i.e name, position, affiliation, address, phone, email, etc.) is extracted from his/her homepage. The box on the right side of the page lists other entities that are assumed to be related to the given researcher. More specifically, this box contains three types of entities: 1) other researchers 2) expertise or keywords 3) conferences or journals. The related researchers are the top co-authors with the given researcher based on the number of mutual publications. The expertise is the frequent keywords in the publications by the given researcher. The related conferences are the ones that the given researcher has published papers on them. Finally, at the bottom of the page, the research interests are listed for the given researcher. These research interests are detected based on the proposed topic model in [48].

3.3.2 CIMple project: DBLife

Doan et al. [14] introduce the Community Information Management (CIM) problem. CIM refers to the problem of monitoring, querying, and inferring the entities and their relationships in the online communities. For example recognizing entity mentions, deciding if two mentions refer to the same real-world entity, recognizing a relationship between two entities, etc. To deal with CIM problems, the authors of [14] propose CIM Platform (CIMple). The goal is to build a general framework to adopt broad range of domains with minimum adjustments (i.e any online community could be quickly deployed to manage its data). DBLife [13] is a prototype for CIMple project which only covers the database research community. The following is the workflow of CIMple with *community of database researchers* as an example:

- Feeding CIMple with relevant data sources (e.g. homepages of database researchers, conference pages), domain knowledge about the entities and relationships, and hints

Figure 3.5: The screenshot of ArnetMiner for query "Christos Faloutsos" on June 10, 2011

to extract the entities in that domain, by a domain expert;

- Crawling the sources regularly and finding the relevant mentions (e.g researcher names, conference names, paper titles) ;
- Matching the mentions and grouping them into entities (e.g Papers, Conferences, People);
- Discovering the relationships between entities (e.g If there is a mention of an entity X on the seminar page of a department Y , then X probably is giving a talk at Y).

A challenging issue here is the need to crawl the data sources on a regular basis. This is necessary to keep the constructed Entity Relation (ER) graph consistent with the evolving data and to provide temporal keyword search (e.g querying over the data only in the past two weeks to get updates for a specific entity in the social network). The challenge is how to update the ER graph efficiently. There are two options: 1) Making a new one from scratch 2) Updating the existing one incrementally. Of course updating incrementally is ideal but it is not trivial. There are issues such as: How to match newly found mentions with the

existing entities? How to merge the new data into the old one? In the other hand making a new ER graph with every crawl is time consuming and not scalable for highly dynamic domains (e.g. finance). Also it prevents providing services like temporal keyword search. To our knowledge, DBLife rebuilds the ER graph from scratch. The module that builds the ER graph, uses a dictionary of names supplied by the domain expert to find entity mentions.

Building the ER model from the raw data gathered from Internet, CIMple aims to provide services such as:

- (1) Keyword Search: given a keyword query, it returns the match entities, relationships, and fragments of the ER graph.
- (2) Entity profiling in *Super Homepages*: it creates a super homepage for each entity and put all the information about the entity, gathered by CIMple, in this page.
- (3) ER graph browsing: viewing an entity profile, the user is able to navigate to related entity profiles through the relationships.
- (4) Temporal keyword search and structured querying: temporal query capabilities like querying the network over a period of time.

Figure 3.6 shows a screenshot from DBLife website[4]. This is the superhomepage for *Christos Faloutsos*. At the top of the page, there are top Google Image Search results for the name of the given researcher. Below that, there is the list of publications from DBLP for the given researcher. At the top right of the page, there is a box containing the basic information for the given researcher such as: position, homepage, affiliation, citation number. Similar to ArnetMiner, DBLife tries to suggest related entities for each other. There are a few tables at the right side of the page showing the related people, related topics, related services, and related organizations. Note that these suggestions are merely based on the constructed ER graph for the social network. In other words, there is a direct link between the given researcher and all the suggested entities. Presumably, entities of each class are sorted based on the weight of the link to the given researcher.

3.3.3 Microsoft Academic Search

Microsoft Academic Search is a free academic search engine developed by Microsoft Research[5]. This website provides a user friendly interface to search and browse in scientific

[4] http://dblife.cs.wisc.edu
[5] http://academic.research.microsoft.com

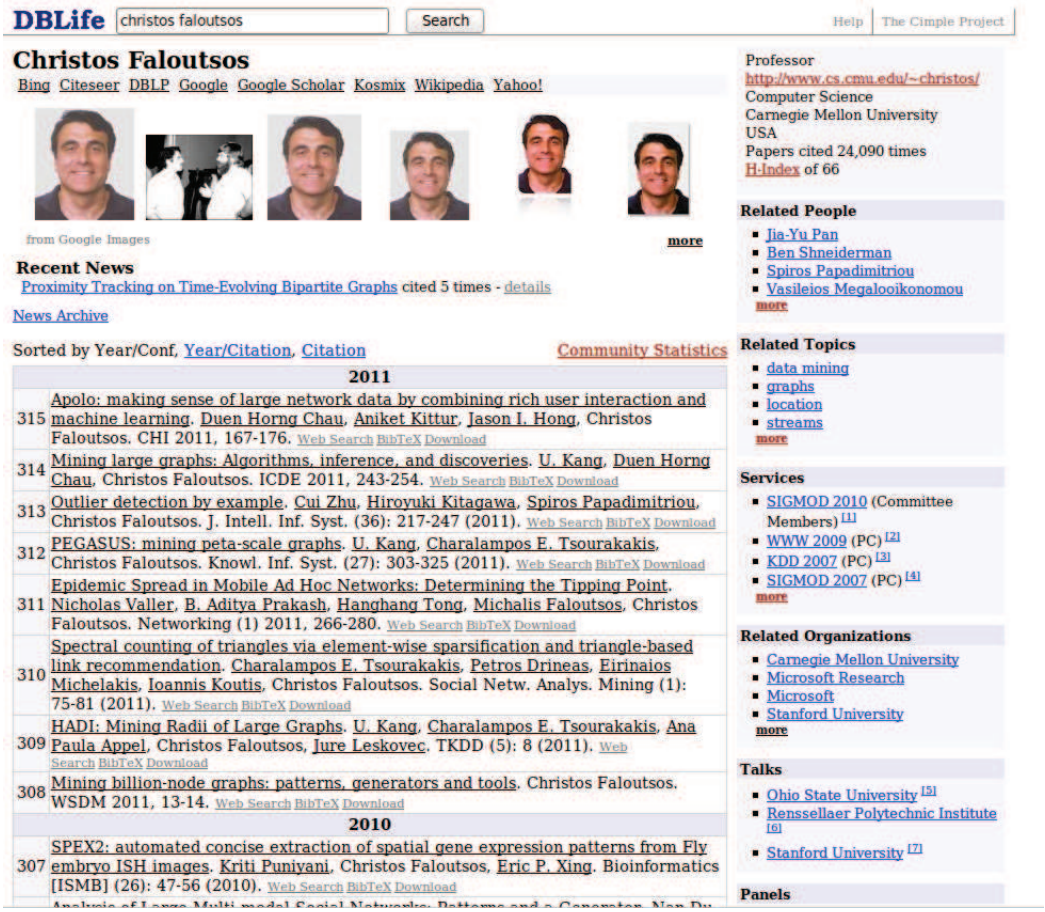


Figure 3.6: The screenshot of DBLife for query "Christos Faloutsos" on June 10, 2011

publications, journals, conferences, authors. Even though Microsoft Academic Search is still on a beta version, the results are encouraging. To best of our knowledge, Microsoft Research has not disclosed its approach for this website yet. However, based on the information provided in the website, entities in the search results are sorted both by their relevance to the query and their global importance. Microsoft Research group claims that they are using technologies introduced in [37], [35], [36], [60], [59], [34]. It is said that the relevance score of an entity is computed by its attributes; and the importance score of an entity is calculated by its relationships with other objects.

Figure 3.7 shows a screenshot from Microsoft Academic Search website. This page is the profile of *Christos Faloutsos* in the system. Similar to ArnetMiner and DBLife, the left side bar contains the related entities to the given researcher. Top co-authors, conferences, journals, and keywords based on the publications of the given researchers are listed. Some basic information for the given researcher such as: affiliation, number of co-authors, H-Index, and G-Index are provided at the top centre of the page. In addition to that, the

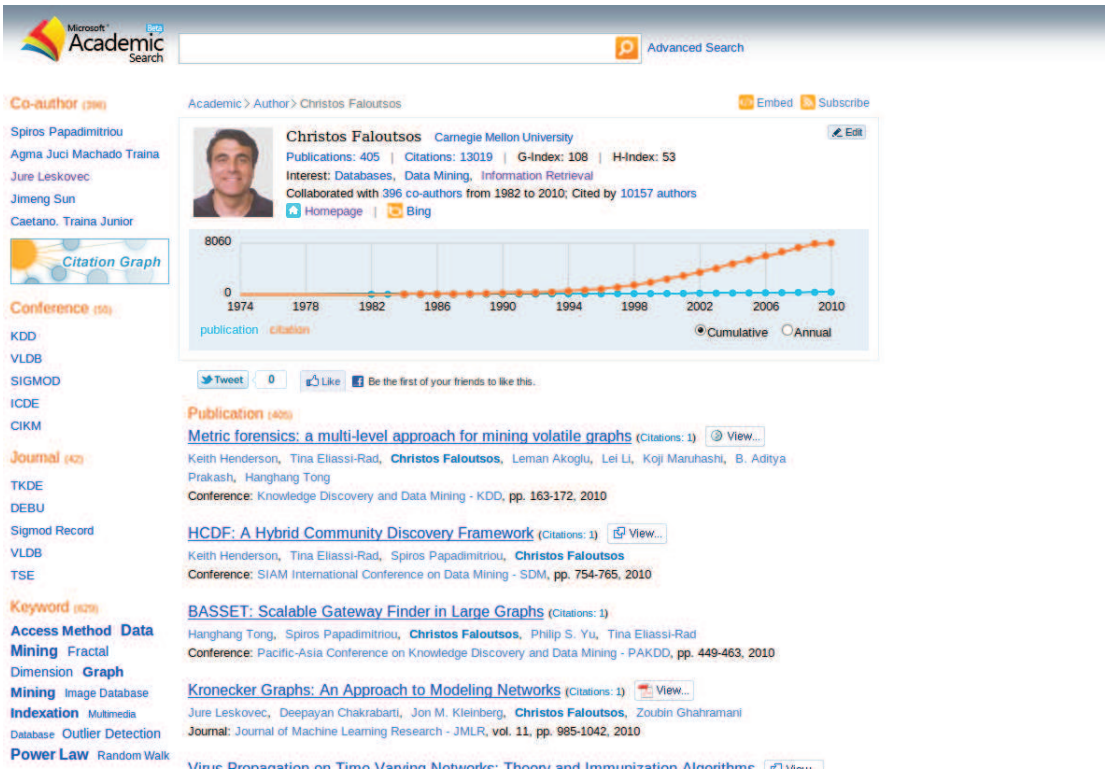


Figure 3.7: The screenshot of Microsoft Academic Search for query "Christos Faloutsos" on June 10, 2011

number of publications and the number of citations in other publications, is provided in a graph. Also the publications by the given researcher are listed chronologically.

3.3.4 DBLP Cloud Mining

Cloud mining lets the users to refine their queries using tag clouds. A particular instance of Cloud Mining is constructed on the bibligraphical data from DBLP. This website[6] provides the user a handy way of querying the data available in DBLP. It also ranks the entities based on the citation numbers provided by CiteSeer. Figure 3.8 represents a screenshot from this website for *Christos Faloutsos*. The list of his publications from DBLP ordered based on the number of citations from CiteSeer is available at the centre of the page. There are three boxes on the right side showing the related researchers, keywords, and venues. These lists represented in a user friendly way which shows entity names in different sizes. The bigger the item is, the more frequent that item is with respect to the query (i.e. Christos Faloutsos). Note that this website does not provide any ranking of entities based on each other rather it lists the entities with a relationship with the queried entity. In other words, if there is

[6]http://dblp.cloudmining.net/

an edge between the two nodes in the corresponding graph, the entity is shown in the list. However, based on the weight of the edge, the item may be larger or smaller.

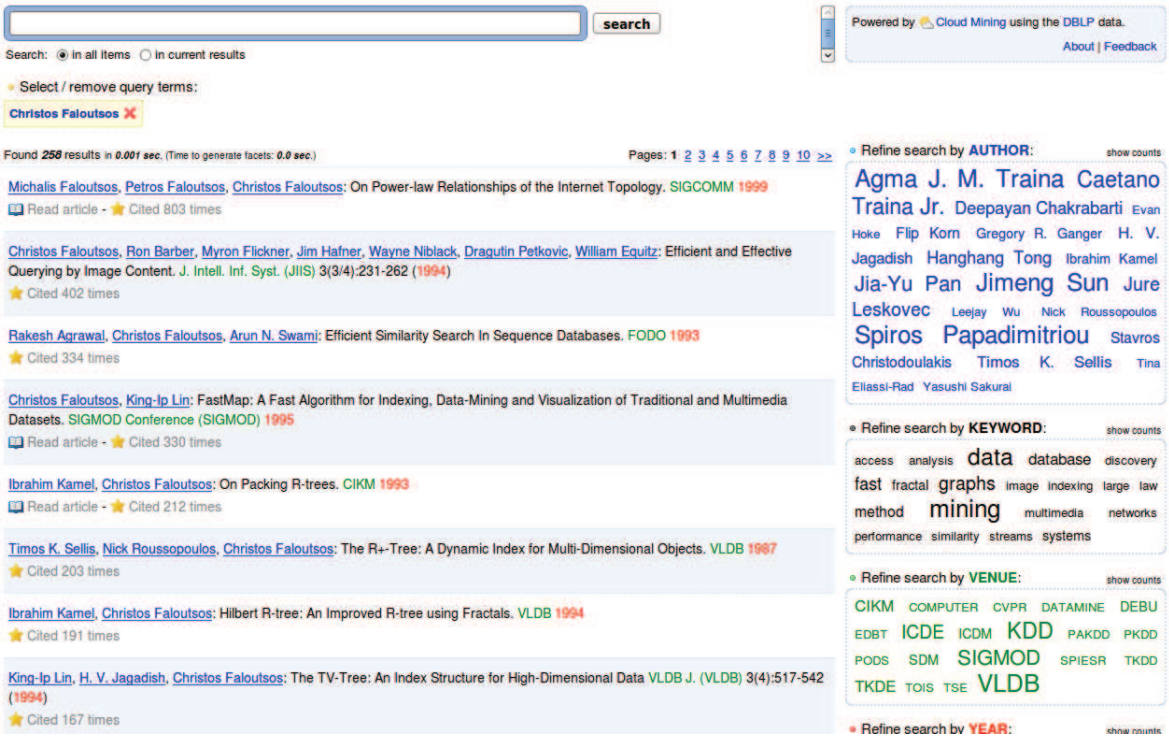


Figure 3.8: The screenshot Cloud Mining for query "Christos Faloutsos" on July 28, 2011

3.3.5 DBConnect: the prototype version

Zaiane et al. [57] propose a random walk solution for the problem of ranking entities in relational databases. Users of large databases are interested in the top-k tuples that are most *related* to their queries. Thus the authors assign a relevance score to categorical data in relational databases. For this purpose, they construct the database graph where the entity tables serve as partitions of nodes and the relation tables serve as edges between partitions. Furthermore, they develop a tool called DB-Connect and apply a random walk approach on the extended k-partite graph to obtain the relevance values for the bibliographic information from DBLP[7].

The authors apply a variation of random walk called *Random Walk with Restart* (RWR) on the constructed graph from the relational database to find relevant nodes for a specific instance. Random Walk with Restart [46] is similar to the original random walk. The only difference is when choosing the next edge on a node, there is always a probability $P_{restart}$

[7] http://www.informatik.uni-trier.de/ ley/db

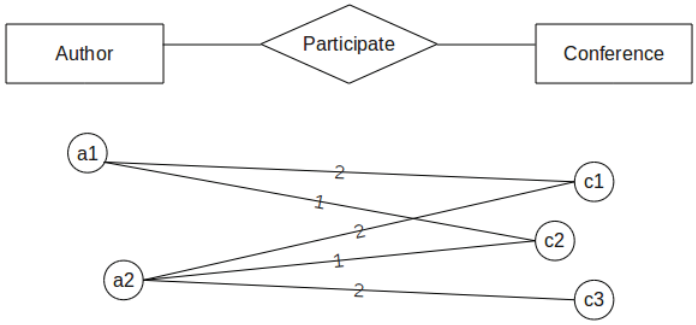


Figure 3.9: Database Graph based on the Entity-Relationship Model

Conference	Author Participation
c_1	$\{a_1, a_2\}, \{a_1\}, \{a_2\}$
c_2	$\{a_1, a_2\}$
c_3	$\{a_1\}, \{a_2\}$

Table 3.1: Conference participation by different authors

which the agent goes back to the initial state. This makes the closer neighbours of the initial node, to get higher probability of being visited by the random agent. Sun et al. [46] used RWR on the bipartite graphs for the first time to compute a relevance score for nodes to find relevant and irrelevant entities for a given individual.

Inspired by the above study, Zaiane et al. [57] apply the idea of using RWR to rank entities for relational databases. The authors believe there are some information in relational databases which are usually ignored in the normal graph representation. Therefore the authors introduce the concept of *returning relations*. A "returning relation" is a relationship between nodes n_1 and n_2 in the same partition via node m in a different partition. For example, the co-author relation between authors a_1 and a_2 is a returning relation because it goes through conference c_1. Based on Table 3.1 which shows the participation for a few conferences, we observe that a_1 has co-authored with a_2 on a publication in c_1 and yet has another publication in c_1. Constructing the weighted bi-partite graph for this domain, we observe it does not capture the co-author relations. Figure 3.9 illustrates this issue in the weighted graph of this problem: the only available information about a_1 is that he has 2 publications in c_1; no information about co-author relation is revealed.

To model *returning relations* in relational databases, [57] proposes the construction of a virtual layer in the database graph, acting as surrogate nodes to replace the original nodes. This results in a larger directed bipartite graph, since each node in one part of the graph

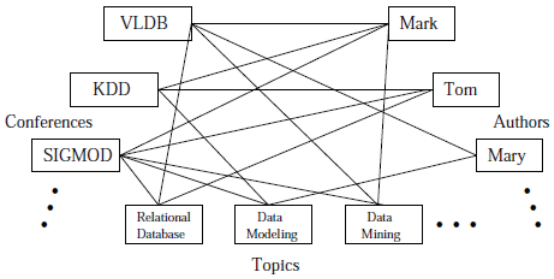


Figure 3.10: [57] Tripartite graph model for Author-Conference-Topic

splits into as many nodes in the other part (e.g for the graph shown in Figure 3.9, each conference node splits into two nodes, representing the number of authors connected to it).

Although the experimental results are promising, this approach does not scale even for average size datasets. On the other hand, it is not clear how the returning relations are implemented when there are more than two entities in the database (i.e when the corresponding graph is not bipartite, instead a k-partite graph where $k >= 3$). Also when the number of *parts* in the graph exceeds two, there is an issue of direction for the proposed random walk. Figure 3.10 illustrates this problem. For example to rank authors for *Tom*, there are two possible direction for the random walker. Starting from Tom, going to the Topic part then going to the Conference part and finally coming back to Author part. In contrast, one could start from Tom, visit Conference part then go to Topic part and finally go back to Author part. The results vary based on the selected direction. It is not clear which direction provides the best results.

In this thesis we propose an algorithm to find the most related entities for a specific individual in heterogeneous multiple relation k-partite social networks. Our approach scales for large size datasets. Since for each individual we compute the relevance scores of all nodes separately, to get the complete ranking for the whole dataset we can parallelize the computation easily.

Chapter 4

Ranking In K-Partite Graphs

The most challenging category of social networks is *heterogeneous multiple relation social networks* where not only the relationships are not from the same class, but the entities in the network have different types. A simple example of such a social network consists of three different classes of entities which are related two by two. The corresponding graph for such a social network is a tripartite graph. Figure 4.1 shows the general configuration of such a graph.

Random Walk with Restart (RWR) provides promising results in assigning relevance scores in bipartite graphs by exploiting the relationships between two partitions (e.g in [46]). Even though there are no connections among entities of the same type (nodes in one partition), they are linked through the entities in the other partition. The similar/related instances in Partition A are most likely connected to the same instances in Partition B. Zaiane et al. [57] explore this idea in bibliographic data. Not only the authors use this idea to rank the bipartite graph of "authors" and "conferences", they investigate the application of RWR on a tripartite graph. Thus instead of moving back and forth between two partitions (the original RWR), the random walker picks a random sequence of partitions and visits them iteratively until the state probability converges. Apart from the quality of results, one drawback of this approach is choosing the sequence of partitions by random. Applying RWR on a k-partite graph when $k \geq 3$, the random walker must choose a sequence to visit each partition. For example in Figure 4.1, ranking nodes for a specific instance of A, one might first visit partition B, then C, and finally A (clockwise) or visit C first, then B, and finally A (counter clockwise). The same study shows that different directions for the random walk generate different relevance scores for the nodes. There is no clear way to find the direction which produces the most accurate results. Also, as the number of entity types in the graph increases (i.e as k becomes larger), there are more and more possible directions

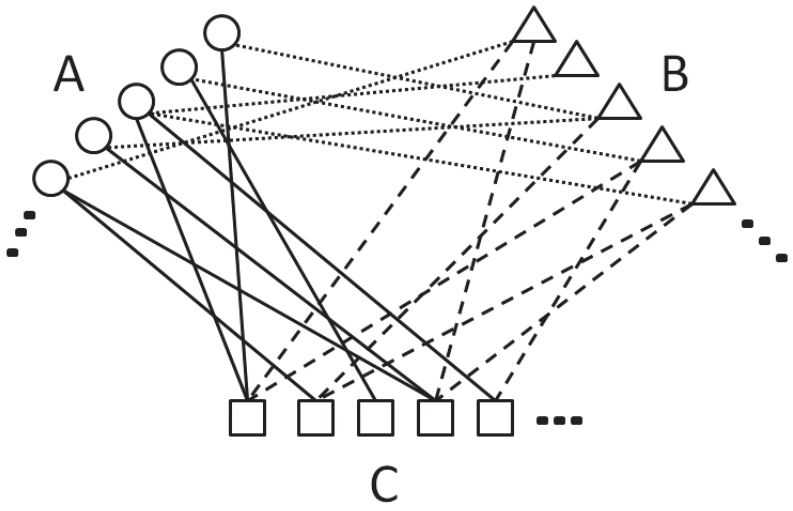


Figure 4.1: Heterogeneous multiple relation social network with 3 different entities A, B, C: The corresponding graph is a tripartite graph

which generate different results. While there are only two different directions for a tripartite graph, there are six different directions in a quadripartite graph, thus six different ranking results are generated. Detecting the best ranking result is not trivial. In fact, comparing the results without ground truth is defective because it brings up this question again: "which entity is more related to this specific instance?"

In this chapter we first explain the original RWR on a bipartite graph which is the building block of our approach. Next we explain our solution for a tripartite graph. Then we propose a general algorithm for k-partite graphs based on RWR on bipartite graphs. The proposed algorithm exploits the fact that any k-partite graph consists of several bipartite graphs. We also resolve the issue of the direction for the random walker. Finally we discuss some computational improvements for our algorithm.

4.1 Random Walk with Restart in Bipartite Graphs

Let $G = (A, B, E)$ be a bipartite graph where A and B are two separate partitions with s_A and s_B nodes respectively. E is the set of edges while every edge connects two nodes from different partitions (i.e one node is in partition A and the other one is in partition B).

The Problem: given a node n_1 in a bipartite graph, we want to rank other nodes (both in the same partition as n_1 and in the other partition) based on their relevance to n_1. This essentially means to assign a relevance score to all other nodes.

In general, to apply RWR on a graph one may use the *adjacency matrix* of the graph. The adjacency matrix is a matrix representing which vertex in the graph is adjacent to which vertices. Let P be the adjacency matrix of a graph with s_{total} nodes. Then the following lemma holds:

Lemma 1 Let c be the probability of restarting the random walk from node α. Then the steady-state vector u_α satisfies Equation 4.1:

$$u_\alpha = (1-c)Pu_\alpha + cv_\alpha \tag{4.1}$$

where v_α is a one-column vector consisting of s_{total} elements, v_α is a vector of zeros except that the element corresponds to α is set to 1, u_α is the steady-state vector which contains relevance scores of all nodes in the graph for α. See [45] for proof of the lemma.

Let M be the adjacency matrix from A to B where rows present instances in A and columns present instances in B. In each step the random walker chooses an edge proportional to the edge weight over the sum of the weights of all outgoing edges. Furthermore, we normalize M such that every row sums up to 1. $M(\alpha, \beta)$ indicates the normalized weight of the edge from node α to β in G (0 means there is no edge between two nodes).

$$M = \begin{array}{c|cccc} & b_1 & b_2 & ... & b_{s_B} \\ \hline a_1 & w_{11} & w_{12} & ... & w_{1s_B} \\ a_2 & w_{21} & w_{22} & ... & w_{2s_B} \\ ... & ... & ... & ... & ... \\ a_{s_A} & w_{s_A1} & w_{s_A2} & ... & w_{s_As_B} \end{array}$$

where $\sum_{c=1}^{s_B} w_{rc} = 1, r = (1, 2, ..., s_A)$.

Since graph G is a bipartite graph, the adjacency matrix would be:

$$P_{(s_B+s_A)\times(s_A+s_B)} = \begin{bmatrix} 0 & M^T \\ M & 0 \end{bmatrix}$$

Applying Equation 4.1 iteratively, u_α converges to the steady-state probability of visiting all the nodes starting from α. More specifically, u_α is a column vector of size $s_A + s_B$ where $u_\alpha(1 : s_B)$ contains the relevance scores of nodes in partition B with respect to α and $u_\alpha(s_B : s_B + s_A)$ contains the relevance scores of nodes in partition A to α.

4.2 Ranking Solution for Tripartite Graphs

As we discussed in Chapter 2, when one has to deal with a large number of objects, often there is a need to rank them. For example in the context of searching in the web, the large number of related pages matching a query, which is not feasible to be reviewed by a human

user, makes the search engine to rank web pages based on their relevance and importance. In fact, users are not willing to check more than a few results for their queries. In many other cases ranking is required because the user needs to know the most related items with respect to one specific instance. For example, in a product recommendation system, products need to be ranked based on each user. Consequently, instead of getting lost in the huge list of all the available products, the user deals with the top ranked products based on his/her shopping patterns and behaviours. In other words, the problem of ranking is reduced to finding the top n related items with respect to a query[1]. n could vary based on the application.

In the previous section, we explained how to compute a relevance score for all entities with respect to one specific instance in a bipartite graph using RWR. Breaking a tripartite graph into bipartite graphs, we use the top score entities found by RWR, to rank entities in the tripartite graph. While the RWR on bipartite graphs computes a relevance score for every ordered pair (a, b), where a and b are arbitrary entities, our solution for tripartite graphs does not compute a relevance score for all ordered triple (a, b, c). Instead we only compute the relevance scores for top related items with respect to each instance. In other words, we calculate a relevance score for a small subset of all ordered triple. We generalize our method for k-partite graphs in the next section. We also propose an efficiency improvement on our algorithm when the number of partitions in the graph becomes larger.

The Problem: Having a tripartite graph of entity types A, B, and C, an ordered list of top n relevant entities of each type with respect to one specific instance (e.g. p_{start}) is required. More specifically we are interested in three lists: $TOP_n^A(p_{start})$ which contains the top n most related instances of A with respect to p_{start}; $Top_n^B(p_{start})$ which contains the top n most related instances of B with respect to p_{start}; and $Top_n^C(p_{start})$ which contains the top n most related instances of C with respect to p_{start}. Note that n may be as small as 1 or as big as the total number of entities.

To explain our solution we require to define *relevance pair*, *relevance set* and *weighted union set*. We define $\omega(O, S)$ as a "relevance pair" where O represents any object/individual in the network and $S \in \Re$ represents its relevance score. Two relevance pairs such as $\omega_1(p, s_p)$ and $\omega_2(q, s_q)$ are equal if and only if $p = q$. A set of relevance pairs is called a "relevance set".

We define $\uplus$ as the *weighted union operator* which is applied to two or more relevance

[1]Note that in some applications the least related items are favourable. In this thesis we only consider the positive case but we believe finding the least related items for query q can be relaxed to find the most related items for $\neg q$; Thus the discussed methods are adaptable for those applications as well.

sets to provide the "Weighted Union Set". Note that the weighted union set is a relevance set. As an example, let us assume X and Y are two relevance sets as follows:

$$X = \{(x_1, s_{x_1}), (x_2, s_{x_2}), ..., (x_m, s_{x_m})\}$$

$$Y = \{(y_1, s_{y_1}), (y_2, s_{y_2}), ..., (y_t, s_{y_t})\}$$

$\biguplus^{X,Y}(n)$ is the weighted union set of X, and Y. Calculating the weighted union set requires four steps. First, we find equal "relevance pairs" between the relevance sets. Any number of equal "relevance pairs" will be merged into one "relevance pair" where its "object" is the same as all of them but its "relevance score" is the summation of the relevance scores of all of them. For example, if x_i equals y_j, the corresponding "relevance pair" in the weighted union set is $(x_i, s_{x_i} + s_{y_j})$. The second step is to copy the remaining relevance pairs, the ones which do not match with any "relevance pair" in the other relevance sets, into the weighted union set. Third, we order the final relevance pairs based on their scores (i.e the highest score comes the first and the lowest score comes at the last). Finally, we select the top n pairs to construct the "weighted union set" where $n \leq max(m, t)$.

Let us calculate $\biguplus^{X,Y,Z}(4)$ where:

$$X = \{(a, 0.030), (b, 0.020)\}$$

$$Y = \{(b, 0.045), (c, 0.025), (d, 0.010), (e, 0.050)\}$$

$$Z = \{(a, 0.010), (d, 0.065), (b, 0.030)\}$$

The first step is to find the equal relevance pairs among these relevance sets and merge them:

$$(a, 0.030)\ ,\ (a, 0.010)\ \longrightarrow\ (a, 0.040)$$

$$(b, 0.020)\ ,\ (b, 0.045)\ ,\ (b, 0.030)\ \longrightarrow\ (b, 0.095)$$

$$(d, 0.010)\ ,\ (d, 0.065)\ \longrightarrow\ (d, 0.075)$$

The second step is to copy the relevance pairs that do not match with any other pair:

$$(c, 0.025)\ ,\ (e, 0.050)$$

Which makes the initial weighted union set as follows:

$$\{(a, 0.040), (b, 0.095), (c, 0.025), (d, 0.075), (e, 0.050)\}$$

The third step is to order the pairs based on their scores:

$$\{(b, 0.095), (d, 0.075), (e, 0.050), (a, 0.040), (c, 0.025)\}$$

The fourth step is to choose the top $n = 4$ relevance pairs:

$$\biguplus_{X,Y,Z} (4) = \{(b, 0.095), (d, 0.075), (e, 0.050), (a, 0.040)\}$$

Now we explain our algorithm to compute the top n related instances for each entity in tripartite graphs.

Let $RWR_{(E_1,E_2)}(p_{start})$ represent a random walk with restart on the bipartite graph of two entities E_1 and E_2 when the starting point is p_{start}. Essentially, $RWR_{(E_1,E_2)}(p_{start})$ generates a *relevance score* for each node in the graph with respect to p_{start}. Note that this score is the probability of visiting the node when the random walker starts from p_{start} and follows edges randomly with respect to their weights. Furthermore we are able to construct a list of "relevance pairs" for instances of each entity.

Without loss of generality we assume the starting point is in A (i.e $p_{start} \in A$). To find the top n related instances of C with respect to p_{start} we follow the below procedure:

First we run $RWR_{(A,B)}(p_{start})$ and construct the relevance set for instances in B. Next we save the top n relevance pairs (top score pairs).

$$\Phi_{(A\to B)}(p_{start}) = (b_1^{AB}, s_{b_1}^{AB}), (b_2^{AB}, s_{b_2}^{AB}), ..., (b_n^{AB}, s_{b_n}^{AB})$$

where b_i^{AB} is the i-th related instance of B and $s_{b_i}^{AB}$ is its score with respect to p_{start}. Also $s_{b_1}^{AB} > s_{b_2}^{AB} > ... > s_{b_n}^{AB}$

In the next step, we run $RWR_{(A,C)}(p_{start})$ to detect the top n instances of C with respect to p_{start}, similar to the previous step.

$$\Phi_{(A\to C)}(p_{start}) = (c_1^{AC}, s_{c_1}^{AC}), (c_2^{AC}, s_{c_2}^{AC}), ..., (c_n^{AC}, s_{c_n}^{AC})$$

Next, we run random walks on the bipartite graph of B and C. Each instance of B in $\Phi_{(A\to B)}(p_{start})$ is a new starting point. Therefore we run the following random walks and in each run we save the set of top n relevance pairs as above:

$$RWR_{(B,C)}(b_1^{AB}) \text{---} top\, n \longrightarrow D_{(B\to C)}^1 = \{(c_{11}^{BC}, sc_{11}^{BC}), (c_{12}^{BC}, sc_{12}^{BC}), ..., (c_{1n}^{BC}, sc_{1n}^{BC})\}$$

$$RWR_{(B,C)}(b_2^{AB}) \text{---} top\, n \longrightarrow D_{(B\to C)}^2 = \{(c_{21}^{BC}, sc_{21}^{BC}), (c_{22}^{BC}, sc_{22}^{BC}), ..., (c_{2n}^{BC}, sc_{2n}^{BC})\}$$

...

$$RWR_{(B,C)}(b_n^{AB}) \text{---} top\, n \longrightarrow D^n_{(B\to C)} = \{(c^{BC}_{n1}, sc^{BC}_{n1}), (c^{BC}_{n2}, sc^{BC}_{n2}), ..., (c^{BC}_{nn}, sc^{BC}_{nn})\}$$

Furthermore we multiply the score of the relevance pairs in $D^i_{(B\to C)}$ with $s^{AB}_{b_i}$ where $i = 1, 2, ..., n$. In other words, whenever an instance of C (i.e c^{BC}_{ij}) is reached through an instance of B (i.e b^{AB}_i), we multiply their score to get a more exact relevance score with respect to p_{start}. For example $(c^{BC}_{11}, sc^{BC}_{11})$ is updated to $(c^{BC}_{11}, sc^{BC}_{11} * s^{AB}_{b_1})$ and $(c^{BC}_{2n}, sc^{BC}_{2n})$ changes to $(c^{BC}_{2n}, sc^{BC}_{2n} * s^{AB}_{b_2})$. This update is done because we want to take the effect of B into account when the random walker travels from partition A to C through B.

Using the weighted union operator $\biguplus$, we merge the updated relevance sets . This means if an instance of C is repeated in more than one list, the final score for that instance in the weighted union set is the sum over all the occurrences of that instance. For example if c^{BC}_{1n} equals c^{BC}_{22} the final score for this instance in the union set would be $(sc^{BC}_{1n} * s^{AB}_{b_1} + sc^{BC}_{22} * s^{AB}_{b_2})$. In this way we favour the instances with multiple occurrences; the more an instance is repeated in different lists, the more likely that instance is related to p_{start}.

$$\Phi_{A\to B\to C}(p_{start}) = \biguplus_{i=1}^{n} D^i_{(B\to C)} = \left[(c^{ABC}_1, s^{ABC}_{c_1}), (c^{ABC}_2, s^{ABC}_{c_2}), ..., (c^{ABC}_n, s^{ABC}_{c_n})\right]$$

Finally we combine $\Phi_{A\to B\to C}(p_{start})$ and $\Phi_{(A\to B)}(p_{start})$ using the weighted union operator to find the top n instances of C with respect to p_{start}.

$$Top^C_n(p_{start}) = \Phi_{A\to B\to C}(p_{start}) \biguplus \Phi_{(A\to C)}(p_{start}) = [(c^*_1, s^*_{c_1}), (c^*_2, s^*_{c_2}), ..., (c^*_n, s^*_{c_n})]$$

Figure 4.2 illustrates this procedure.

To find the most related instances of B with respect to p_{start}, we use a similar approach. Random walks are run on the bipartite graph of C and B but this time instances of C in $\Phi_{(A\to C)}(p_{start})$ are the starting points. Therefore we have:

$$RWR_{(C,B)}(c_1^{AC}) \text{---} top\, n \longrightarrow D^1_{(C\to B)} = \left[(b^{CB}_{11}, sb^{CB}_{11}), (b^{CB}_{12}, sb^{CB}_{12}), ..., (b^{CB}_{1n}, sb^{CB}_{1n})\right]$$

$$RWR_{(C,B)}(c_2^{AC}) \text{---} top\, n \longrightarrow D^2_{(C\to B)} = \left[(b^{CB}_{21}, sb^{CB}_{21}), (b^{CB}_{22}, sb^{CB}_{22}), ..., (b^{CB}_{2n}, sb^{CB}_{2n})\right]$$

...

$$RWR_{(C,B)}(c_n^{AC}) \text{---} top\, n \longrightarrow D^n_{(C\to B)} = \left[(b^{CB}_{n1}, sb^{CB}_{n1}), (b^{CB}_{n2}, sb^{CB}_{n2}), ..., (b^{CB}_{nn}, sb^{CB}_{nn})\right]$$

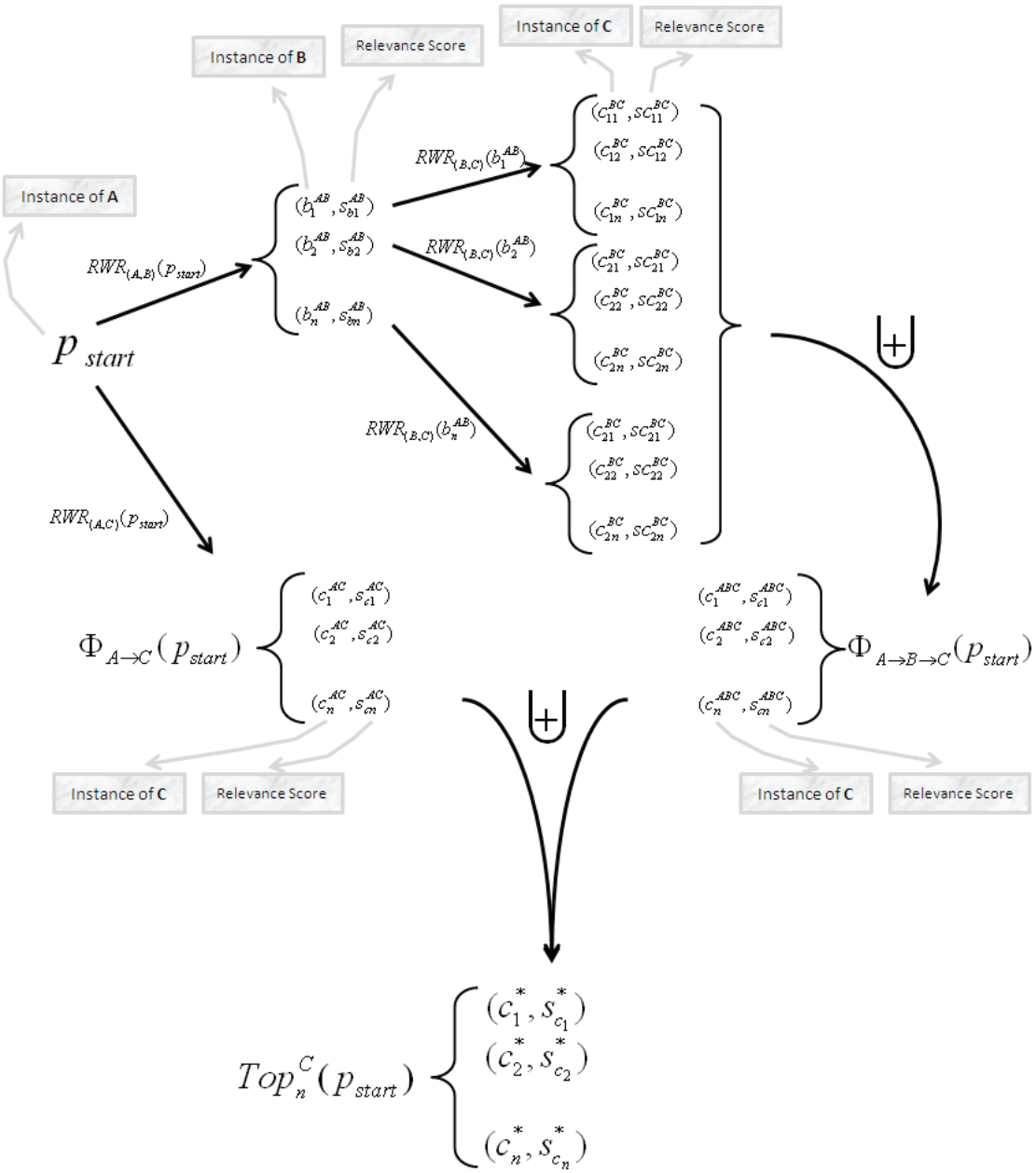


Figure 4.2: Procedure to find the most related instances of C with respect to p_{start}. Note that the procedure for finding the most related instance of B with respect to $p_{start} \in A$ is similar.

Furthermore we update all the relevance pairs by multiplying their score with the score of the corresponding starting point. It means all the relevance scores in $D^i_{(C\to B)}$ are multiplied with $s^{AC}_{c_i}$ for $i = 1, 2, ..., n$.

The weighted union set is constructed using the weighted union operator on the updated relevance sets.

$$\Phi_{A\to C\to B}(p_{start}) = \biguplus_{i=1}^{n} D^i_{(C\to B)} = \Big[(b_1^{ACB}, s_{b_1}^{ACB}), (b_2^{ACB}, s_{b_2}^{ACB}), ..., (b_n^{ACB}, s_{b_n}^{ACB})\Big]$$

Finally we merge $\Phi_{A\to C\to B}(p_{start})$ and $\Phi_{(A\to B)}(p_{start})$ using the weighted union operator $\biguplus$ to find the top n instances of B with respect to p_{start}.

$$Top^B_n(p_{start}) = \Phi_{A\to C\to B}(p_{start}) \biguplus \Phi_{(A\to B)}(p_{start}) = [(b_1^*, s_{b_1}^*), (b_2^*, s_{b_2}^*), ..., (b_n^*, s_{b_n}^*)]$$

Last but not least, we compute the most related instances of A with respect to p_{start}. Since p_{start} is in the same class (partition) as the instances of A, and we assume there is no relationship among entities of the same partition, we use the top related instances of other partitions with respect to p_{start}. In fact we hypothesize that $Top^A_n(p_{start})$ includes instances of A which are close to the top instances of B and C with respect to p_{start}. To do this we use $Top^B_n(p_{start})$ and $Top^C_n(p_{start})$. Each instance of B in $Top^B_n(p_{start})$ would be the starting point of a random walk on the bipartite graph of B and A.

$$RWR_{(B,A)}(b_1^*) \text{---} top\ n \longrightarrow\ D^1_{(B\to A)} = \Big[(a_{11}^{BA}, sa_{11}^{BA}), (a_{12}^{BA}, sa_{12}^{BA}), ..., (a_{1n}^{BA}, sa_{1n}^{BA})\Big]$$

$$RWR_{(B,A)}(b_2^*) \text{---} top\ n \longrightarrow\ D^2_{(B\to A)} = \Big[(a_{21}^{BA}, sa_{21}^{BA}), (a_{22}^{BA}, sa_{22}^{BA}), ..., (a_{2n}^{BA}, sa_{2n}^{BA})\Big]$$

...

$$RWR_{(B,A)}(b_n^*) \text{---} top\ n \longrightarrow\ D^n_{(B\to A)} = \Big[(a_{n1}^{BA}, sa_{n1}^{BA}), (a_{n2}^{BA}, sa_{n2}^{BA}), ..., (a_{nn}^{BA}, sa_{nn}^{BA})\Big]$$

All the relevance sets are updated as before. This means the relevance scores in $D^i_{(B\to A)}$ are multiplied by $s^*_{b_i}$ where $i = 1, 2, ..., n$ (the score of the corresponding starting point in RWR). Furthermore these relevance sets are merged into $\Phi_{A\to C\to B\to A}(p_{start})$ using the weighted union operator $\biguplus$.

$$\Phi_{A\to C\to B\to A}(p_{start}) = \biguplus_{i=1}^{n} D^i_{(B\to A)} = \Big[(a_1^{ACBA}, s_{a_1}^{ACBA}), (a_2^{ACBA}, s_{a_2}^{ACBA}), ..., (a_n^{ACBA}, s_{a_n}^{ACBA})\Big]$$

Similarly each instance of C in $Top^C_n(p_{start})$ would be the starting point of a random walk on the bipartite graph of C and A.

$$RWR_{(C,A)}(c_1^*) \text{ --- } top\ n \longrightarrow\ D^1_{(C\to A)} = \Big[(a^{CA}_{11}, sa^{CA}_{11}), (a^{CA}_{12}, sa^{CA}_{12}), ..., (a^{CA}_{1n}, sa^{CA}_{1n})\Big]$$

$$RWR_{(C,A)}(c_2^*) \text{ --- } top\ n \longrightarrow\ D^2_{(C\to A)} = \Big[(a^{CA}_{21}, sa^{CA}_{21}), (a^{CA}_{22}, sa^{CA}_{22}), ..., (a^{CA}_{2n}, sa^{CA}_{2n})\Big]$$

...

$$RWR_{(C,A)}(c_n^*) \text{ --- } top\ n \longrightarrow\ D^n_{(C\to A)} = \Big[(a^{CA}_{n1}, sa^{CA}_{n1}), (a^{CA}_{n2}, sa^{CA}_{n2}), ..., (a^{CA}_{nn}, sa^{CA}_{nn})\Big]$$

Again, all the relevance scores in $D^i_{(C\to A)}$ are multiplied by $s^*_{c_i}$ (the score of the corresponding starting point in RWR) where $i = 1, 2, ..., n$. Next, we merge the updated relevance sets into $\Phi_{A\to B\to C\to A}(p_{start})$.

$$\Phi_{A\to B\to C\to A}(p_{start}) = \biguplus_{i=1}^{n} D^i_{(C\to A)} = \Big[(a_1^{ABCA}, s_{a_1}^{ABCA}), (a_2^{ABCA}, s_{a_2}^{ABCA}), ..., (a_n^{ABCA}, s_{a_n}^{ABCA})\Big]$$

Finally we combine $\Phi_{A\to C\to B\to A}(p_{start})$ and $\Phi_{A\to B\to C\to A}(p_{start})$ using the weighted union operator to find the top n instances of A with respect to p_{start}.

$$Top^A_n(p_{start}) = \Phi_{A\to C\to B\to A}(p_{start}) \biguplus \Phi_{A\to B\to C\to A}(p_{start}) = [(a_1^*, s_{a_1}^*), (a_2^*, s_{a_2}^*), ..., (a_n^*, s_{a_n}^*)]$$

4.3 Ranking Algorithm for K-Partite Graphs

In the previous section we explained our ranking approach for tripartite graphs. In summary, we exploit the relationship between pairs of partitions by running Random Walk with Restarts on the bipartite graphs and merging the results with the weighted union operator $\biguplus$. Not only breaking the problem into bipartite graphs solves the direction problem for the random walker (refer to Chapter 3 and [57]), but it also enables us to adapt our approach for any number of partitions in the network (i.e. $k > 3$).

Let G be a k-partite graph: k different partitions of nodes corresponding to k different entity types are distinguishable in the graph. There may be an edge between any pair of nodes in different parts but not the ones in the same partition. Thus G consists of several bipartite graphs. For now let us assume G is a complete k-partite graph where there is a relationship between any two partitions. The key idea in ranking different entities in such graphs is to break the graph into imaginary smaller pieces in which we only deal with bipartite graphs.

Let E_1, E_2, ..., and E_k be the k entity types corresponding to k partitions in the graph. Let $p_{start} \in E_\theta$ be the node which we want to rank all entities for it (i.e we want to find

the top n instances of each entity type with respect to p_{start}). Therefore we are interested in the following "relevance sets":

$$Top_n^{E_1}(p_{start})\ ,\ Top_n^{E_2}(p_{start})\ ,\ ...\ ,\ Top_n^{E_k}(p_{start})$$

4.3.1 Top n instances in other partitions

For calculating $Top_n^{E_\omega}(p_{start})$ where $\omega = 1, 2, ..., k\ ,\ \omega \neq \theta$, we consider all the paths between E_ω and E_θ: from the shortest path (the direct relationship between E_ω and E_θ) to the longest paths (paths that go through all other partitions in G). Let Ψ be the set of all entity types except E_θ (i.e the source) and E_ω (i.e the destination).

$$\Psi = \{E_1, E_2, ..., E_{\theta-1}, E_{\theta+1}, ..., E_{\omega-1}, E_{\omega+1}, ..., E_k\}\ \ ,\ \ \mid \Psi \mid = k-2$$

Let $P(\Psi)$ be the powerset of Ψ which also includes all the permutations. For example if $\Psi = \{a, b, c\}$ then

$$P(\Psi) = \{(), (a), (b), (c), (ab), (ba), (ac), (ca), (bc), (cb), (abc), (acb), (bac), (bca), (cab), (cba)\}$$

Since Ψ has $k-2$ elements:

$$\mid P(\Psi) \mid = \sum_{i=0}^{k-2} \frac{(k-2)!}{(k-2-i)!} = \lambda$$

Thus we can present $P(\Psi)$ as $P(\Psi) = \{r_1, r_2, ..., r_\lambda\}$. Furthermore we present all the paths between E_θ and E_ω as $E_\theta r_i E_\omega$ where $i = 1, 2, ..., \lambda$.

We calculate λ relevance sets, one for each path. We merge those λ relevance sets using the weighted union operator $\uplus$ to find the most related instances of E_ω with respect to p_{start}. Here we explain how the relevance set is calculated for each path.

Let $E_\theta E'_1 E'_2 ... E'_m E_\omega$ be an arbitrary path where $0 \leq m \leq k-2$. Figure 4.3 illustrates the procedure of finding the relevance set for this path. In the first step we run a random walk on the bipartite graph between E_θ and E'_1 which provides us a relevance set of instances in E'_1. Then we select the top n relevance pairs.

$$RWR_{(E_\theta, E'_1)}(p_{start}) \text{ --- } top\ n \longrightarrow\ D_{E'_1} = \{(e_{11}, s_{11}), (e_{12}, s_{12}), ..., (e_{1n}, s_{1n})\}$$

From here we follow an iterative procedure which repeats m times. The procedure works on two consecutive entities on the path. In other words we apply the same procedure on $E'_1 E'_2$ and $E'_2 E'_3$ and $E'_3 E'_4$ and ... and $E'_{m-1} E'_m$ and $E'_m E_\omega$. Therefore we explain the algorithm for $E'_i E'_{i+1}$ where i could take any value from 1 to m.

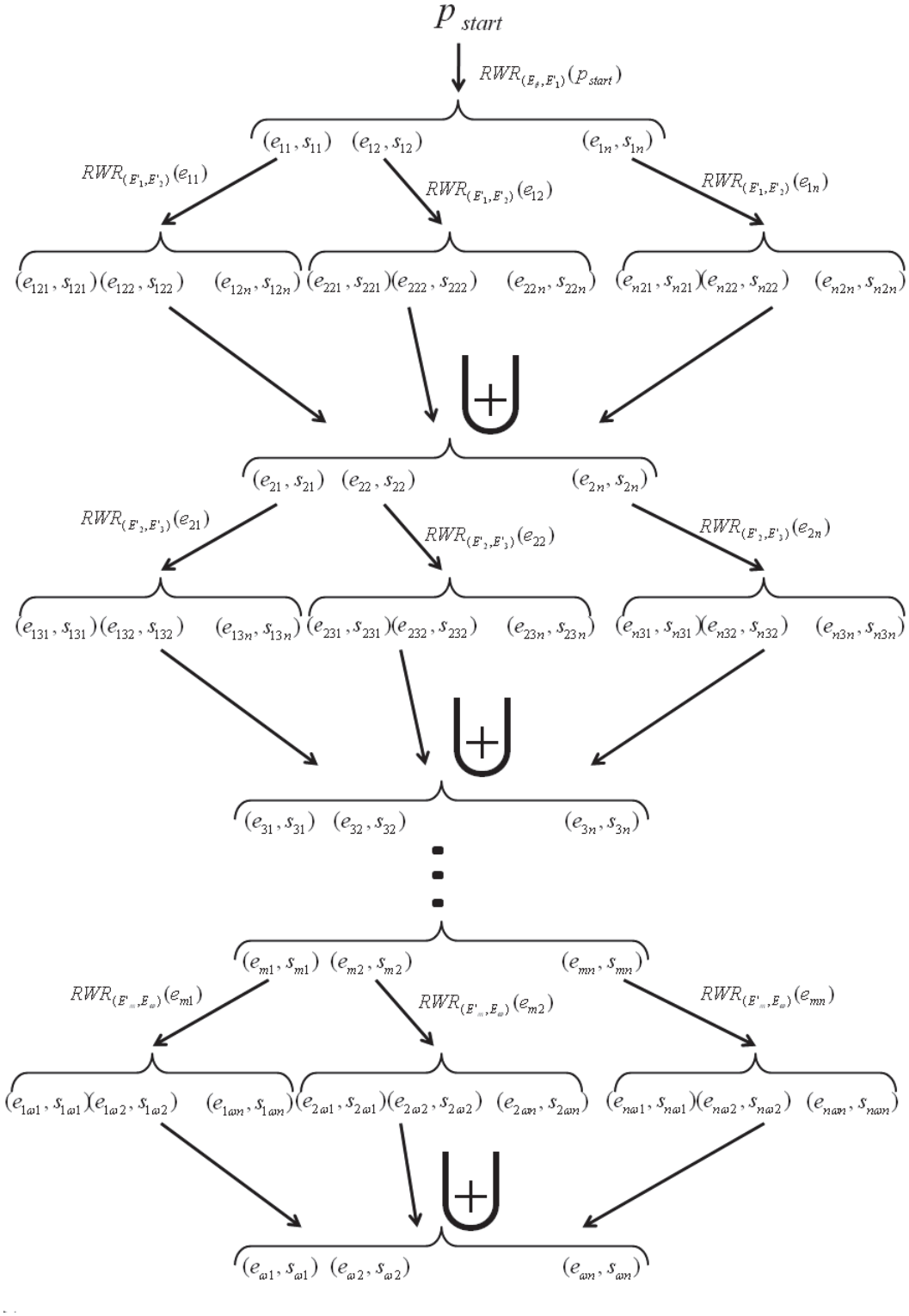


Figure 4.3: The procedure of calculating the relevance set for an arbitrary path $E_\theta E'_1 E'_2 ... E'_m E_\omega$

Each element in $D_{E'_i} = \{(e_{i1}, s_{i1}), (e_{i2}, s_{i2}), ..., (e_{in}, s_{in})\}$ is a new starting point for a random walk on the bipartite graph between E'_i and E'_{i+1}.

$$RWR_{(E'_i,E'_{i+1})}(e_{i1}) \text{---} top\, n \longrightarrow D^1_{E'_i} = \{(e_{1(i+1)1}, s_{1(i+1)1}), (e_{1(i+1)2}, s_{1(i+1)2}), ..., (e_{1(i+1)n}, s_{1(i+1)n})\}$$

$$RWR_{(E'_i,E'_{i+1})}(e_{i2}) \text{---} top\, n \longrightarrow D^2_{E'_i} = \{(e_{2(i+1)1}, s_{2(i+1)1}), (e_{2(i+1)2}, s_{2(i+1)2}), ..., (e_{2(i+1)n}, s_{2(i+1)n})\}$$

...

$$RWR_{(E'_i,E'_{i+1})}(e_{in}) \text{---} top\, n \longrightarrow D^n_{E'_i} = \{(e_{n(i+1)1}, s_{n(i+1)1}), (e_{n(i+1)2}, s_{n(i+1)2}), ..., (e_{n(i+1)n}, s_{n(i+1)n})\}$$

We multiply the scores of all relevance pairs in $D^j_{E'_i}$ with s_{ij} (the relevance score of the starting point of the corresponding random walk) where $j = 1, 2, ..., n$. This is to consider both the length of the path and the importance of visited partitions on the way, in the score of final relevance pairs.

After updating the above relevance sets, we construct $D_{E'_{i+1}}$ by applying the weighted union operator on them.

$$D_{E'_{i+1}} = \biguplus_{j=1}^{n} D^j_{E'_i} = \{(e_{(i+1)1}, s_{(i+1)1}), (e_{(i+1)2}, s_{(i+1)2}), ..., (e_{(i+1)n}, s_{(i+1)n})\}$$

After m iterations we come up with the relevance set for path $E_\theta E'_1 E'_2 ... E'_m E_\omega$.

Now that we are able to compute the relevance set for any arbitrary path, we calculate the corresponding relevance set for $E_\theta r_i E_\omega$ where $i = 1, 2, ..., \lambda$. Let $RS(E_\theta r_i E_\omega)$ be the relevance set for path $E_\theta r_i E_\omega$. To calculate the top n related instances of E_ω with respect to $p_{start} \in E_\theta$, we apply the weighted union operator on the relevance sets:

$$Top^{E_\omega}_n(p_{start}) = \biguplus_{i=1}^{\lambda} RS(E_\theta r_i E_\omega)$$

4.3.2 Top n instances in the same partitions

Here we present the calculation of $Top^{E_\theta}_n(p_{start})$. Recall that $p_{start} \in E_\theta$ and we found the relevant instances to p_{start} of all entity types except E_θ. After finding the top relevant instances of each entity type with respect to p_{start}, we use them to calculate $Top^{E_\theta}_n(p_{start})$. More specifically, we consider the instances in $Top^{E_\omega}_n$, $\omega \neq \theta$ as the starting points of random walks on the bipartite graph between E_ω and E_θ. Therefore for every E_i where $i = 1, 2, ..., \theta - 1, \theta + 1, ..., k$ we run the following random walks:

$$RWR_{(E_i,E_\theta)}(e_{i1}) \text{---} top\, n \longrightarrow D^1_{E_i} = \{(e^i_{11}, s^i_{11}), (e^i_{12}, s^i_{12}), ..., (e^i_{1n}, s^i_{1n})\}$$

$$RWR_{(E_i,E_\theta)}(e_{i2}) \text{ --- } top\ n \longrightarrow\ D^2_{E_i} = \{(e^i_{21}, s^i_{21}), (e^i_{22}, s^i_{22}), ..., (e^i_{2n}, s^i_{2n})\}$$

...

$$RWR_{(E_i,E_\theta)}(e_{in}) \text{ --- } top\ n \longrightarrow\ D^n_{E_i} = \{(e^i_{n1}, s^i_{n1}), (e^i_{n2}, s^i_{n2}), ..., (e^i_{nn}, s^i_{nn})\}$$

Similar to the previous step, we update the scores in $D^j_{E_i}$, multiplying them by s_{ij} where $j = 1, 2, ..., n$. This takes the effect of each entity type in relation with E_θ into account. In other words, the closer[2] the instances of an entity with respect to p_{start}, the more influential they are in ranking the same type instances for p_{start}.

Next, we construct a relevance set for every entity E_i as follows:

$$D_{E_i} = \biguplus_{j=1}^{n} D^j_{E_i}$$

Finally we combine the relevance sets from all entity types to find the top n related instances of E_θ with respect to p_{start}.

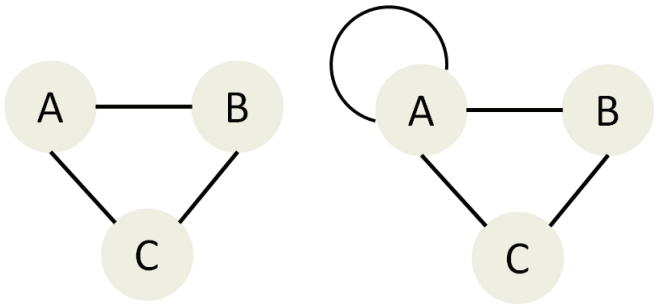
$$Top^{E_\theta}_n(p_{start}) = \biguplus_{i=1}^{k} D_{E_i}\ ,\ i \neq \theta$$

The proposed approach enables us to rank nodes in k-partite graphs. In other words, we are able to find the most related instances of each entity type for any object in heterogeneous multiple relation social networks.

4.3.3 Time Complexity

The proposed ranking approach for a complete k-partite graph considers all the paths between the source and the target partition to compute the weighted union set. As we explained before there are $\lambda = \sum_{i=0}^{k-2} \frac{(k-2)!}{(k-2-i)!}$ paths between the source and the target partitions. The longest paths contain $k-2$ partitions in between. Consequently there are $k-2$ steps, each running n RWR on a bipartite graph. All in all, the time complexity of computing the top n related instances of an entity type with respect to one instance in a complete k-partite graph is $O(nk\lambda)$. More specifically the time complexity is linear to n but it is factorial to k.

[2] closeness here means having higher relevance scores



(a) instances of 3 entity types make a tripartite graph

(b) the same social network with an inner relation on A

Figure 4.4: General schema of a tripartite graph

4.4 Computational Improvements

In the previous section we presented an approach to rank entities in k-partite graphs. In this section we discuss three issues in our approach and suggest an improvement or a solution in each case.

4.4.1 Inner Relations

Presenting our ranking approach on k-partite graphs, we assumed it works on heterogeneous multiple relation social networks. The corresponding graph of such social networks is not always a k-partite graph. If instances of one class are also related to each other, there is a loop in the corresponding graph. In other words, the assumption of having no edges among nodes of the same partition is wrong. To make this more clear let us zoom out the graph in Figure 4.1 in which the instances of each entity type make a node together. Figure 4.4(a) illustrates this idea. Each node in this graph represents an entity type which may have any number of instances. Edges represent the relationships between entity types/partitions. Now if there is a relationship between instances of entity type A the corresponding graph looks like Figure 4.4(b).

Zaiane et al. [57] investigate the issue of having a loop in bipartite graphs. Let A and B be the entity types in a bipartite graph and instances of A are related to both A and B (i.e there is a loop on A). They propose surrogate nodes in which every node in A splits into as many nodes as it is in B. The new nodes make a new layer called A'. Then they generate appropriate edges between A' and B based on $A-A$ and $A-B$ relationships. Not only this approach is not efficient (because it increases the number of nodes and edges in the graph by an order of magnitude), it is not clear how it applies to more than two entities ($k > 2$).

In contrast, we suggest to use loops to filter nodes in the corresponding partitions. Based on the theory of "six degrees of separation"[1] one might speculate when we are looking for top related instances of the same class with respect to a particular individual, say p_{start}, the nodes which are connected to p_{start} with less than 6 intermediaries are more likely to be related to p_{start}. This implies that we can filter the nodes which are not in this range. For example in Figure 4.4(b), computing the top nodes for p_{start} when $p_{start} \in A$, we remove nodes in A which are farther than 6 nodes from p_{start}.

More formally, Let G be a k-partite graph where E_1, E_2, ..., and E_k are the k entity types. G may have inner relations on each entity type. $p_{start} \in E_w$ is the node which we want to rank other entities based on that. If there is an inner relation on E_w, we remove the nodes from E_w which are out of the range of 6 from p_{start} and follow the algorithm presented in the previous section. This particularly reduces the computation complexity of $RWR_{(E_w,E_i)}(p_{start})$ and $RWR_{(E_w,E_ij)}(e_{ij})$ where $i = 1, 2, ..., w-1, w+1, ..., k$ and $j = 1, 2, ...n$.

4.4.2 Computation of Paths

Another computational issue is the large number of valid paths between two entity types when k becomes larger. Recall that in our solution one must compute a relevance set for every possible path between the source and the target entity. This means in a complete k-partite graph one has to compute $\lambda = \sum_{i=0}^{k-2} \frac{(k-2)!}{(k-2-i)!}$ relevance sets to find the top n instances of each entity with respect to one instance. Basically we consider all paths from length 1 to length $k-1$ with all the permutations for each length. Even though for any length, each permutation (i.e different order of visiting entities) generates a slightly different results, for a large number of k we may ignore considering all of the permutations. Instead we pick the "best" permutation for each length. This should not change the ranking results significantly because we aggregate the relevance sets of different paths with different length by the weighted union operator. With this negligible change, we will end up calculating only $k-1$ relevance sets corresponding to $k-1$ paths (length 1 to $k-1$).

Here the challenge is to choose the "best" permutation of entities for each path length. We do that by assigning a score to each permutation and choosing the one with the highest score as the "best" permutation. Let $E_\theta E'_1 E'_2 ... E'_m E_\omega$ be an arbitrary path (length $m+1$) where $0 \leq m \leq k-2$. Except E_θ and E_ω, other entities could change their order. So there are $m!$ combination for this path. To explain how we choose the final combination for this path, first we need to define *inter entity weight*.

Let G be a k-partite graph where E_1, E_2, ..., and E_k are the k entity types with s_{E_1}, s_{E_2}, ..., and s_{E_k} instances respectively. Let n_{ij} represents the number of relationships between instances of E_i and E_j. Also w_{ij} represents the total sum of weights of the relationships between E_i and E_j.

We define the "Inter Entity Weight" (IEW) as the weight between any two entity types in a k-partite graph and it is calculated as in Equation 4.2.

$$IEW_{(E_i,E_j)} = \frac{n_{ij}}{s_{E_i} * s_{E_j}} * \frac{w_{ij}}{n_{ij}} = \frac{w_{ij}}{s_{E_i} * s_{E_j}} \tag{4.2}$$

The intuition behind this definition is simple. $\frac{n_{ij}}{s_{E_i} * s_{E_j}}$ is the ratio of currently existing edges to the maximum possible number of edges between instances of E_i and E_j. In addition $\frac{w_{ij}}{n_{ij}}$ is the average weight per edge.

Back to the problem of choosing one permutation for a fixed length path, we assign a score to each path using IEW. The score of path $E_\theta E'_1 E'_2 ... E'_m E_\omega$ is:

$$S(E_\theta E'_1 E'_2 ... E'_m E_\omega) = IEW_{(E_\theta,E'_1)} + IEW_{(E'_1,E'_2)} + ... + IEW_{(E'_m,E_\omega)}$$

Finally, we choose the permutation that maximizes this score out of $m!$ possible cases. Reducing the number of paths to be considered from $\lambda = \sum_{i=0}^{k-2} \frac{(k-2)!}{(k-2-i)!}$ to $k-1$, this technique decrease the computation complexity when k becomes larger.

4.4.3 Incomplete k-partite Graphs

Another issue is incomplete k-partite graphs. Explaining our approach in Section 4.3, we assumed G to be complete which means there is a relationship between any two entity types. In other words, there is a direct path from any partition to all other partitions in the graph. Now let us consider a more general case where some entity types may not have a direct relationship with each other. We claim our ranking approach still works in such cases. In fact, our approach does not need the complete k-partite graph to rank the entities with respect to each other.

Figure 4.5 depicts an example of an 8-partite graph where not all partitions are connected to each other.

The proposed algorithm only exploits the bipartite graph between any pairs of partitions. Therefore as long as there is a path between any two partitions, the algorithm computes a "relevance set" for that path. If there is more than one path, the algorithm merges the results while each relevance set is weighted inversely based on the number of different partitions along the path. For example in Figure 4.5 when we rank instances of D with respect to an

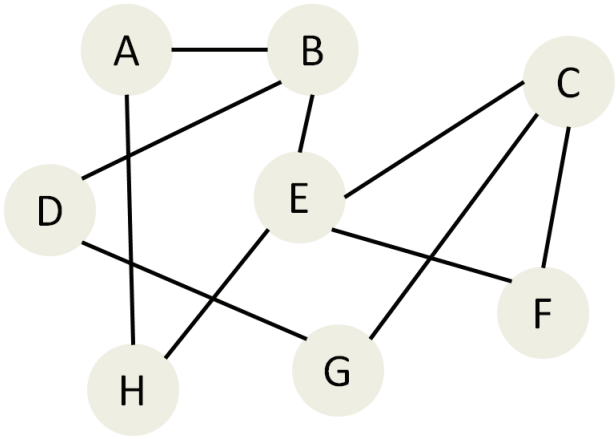


Figure 4.5: An example of an 8-partite graph

individual from E, there are four valid paths for the random walk: EBD and $ECGD$ and $EFCGD$ and $EHABD$. Ranking instances of F for an individual from A, there are six valid paths[3]: $ABEF$, $AHEF$, $AHECF$, $ABECF$, $ABDGCF$, $AHEBDGCF$. All in all, the greater the number of paths between two partitions, the more accurate the ranking results are, while the algorithm is able to find the top n related instances with only one path. Clearly when there is no path between two partitions, there is no relationship between the corresponding entity types. Clearly ranking in such cases is meaningless.

In this chapter, after explaining our solution for ranking entities in tripartite graphs, we proposed a general approach to find the top n related entities with respect to one instance, in k-partite graphs. We should note that our approach finds the top n related instances of each entity type with respect to a node in the network. However, n is not fixed and may change based on the application. In other words, whenever a ranking of all the entities with respect to one instance is required, we simply set n to the total number of entities. In the next chapter, we explain how we apply our ranking approach to a real dataset.

[3]There must not be a duplicate entity type in the path from the source to target

Chapter 5

Application and Methodology

Researchers in different fields publish their new ideas and findings in conferences or scientific journals all around the world. Every publication, annotates a relationship between one or more researcher(s) with the venue of that publication. A deeper look into this area reveals more entities and relationship. Basically, these entities and relationships can be modeled in a social network. We call such social networks as *Academic Social Networks*. In academic social networks, people want to search for papers, conferences, researchers, etc. There are some websites providing the bibliographical information (e.g *CiteSeer*[1], *DBLP*[2], *Google Scholar*[3]). In fact these informations are the main sources of data for such social networks. Several studies have been targeting the analysis of the academic social networks and few systems have been developed to facilitate the need of searching and browsing the academic social networks. (e.g *Microsoft Academic Search*, *ArnetMiner*, *DBLife*). In addition to that, research in the academic world has been increased in the past few years. Not only the members of the academic social networks want to find researchers and publications and conferences, they require more sophisticated information such as: What are the research interests of a given researcher? What are the most related conferences for a given topic? Who are working in the same fields as a given researcher? Who are the most famous researchers with respect to a given topic? and so on.

A quick investigation on these types of questions suggests that there are three critical entities involved in academic social networks.

1 **Author**: researchers who publish papers in conferences and/or journals

2 **Proceeding**: conferences and/or journals which act as communities for researchers

[1] http://citeseerx.ist.psu.edu
[2] http://www.informatik.uni-trier.de/ ley/db
[3] http://scholar.google.com

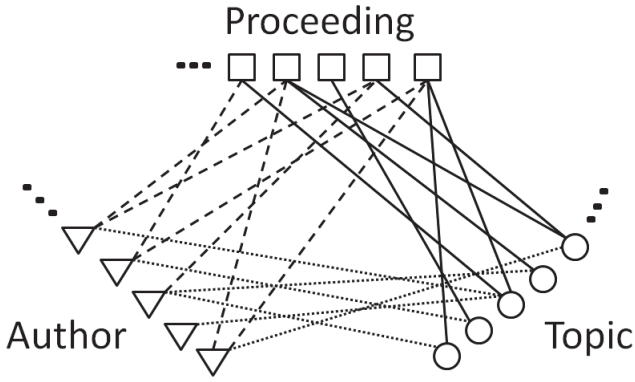


Figure 5.1: An Academic Social Network: tripartite graph of Author, Topic, Proceeding

to interact with each other. (by publishing their ideas)

3 **Topic**: fields of interest for a researcher or topics of interest for a conference or journal

Different entities and various relationships among these entities, make academic social networks to be categorized as Heterogeneous Multiple Relation Social Networks (refer to Chapter 2 for more details). The corresponding graph for such a social network is a tripartite graph as shown in Figure 5.1. To answer such questions for this social network, a mechanism is required to rank entities based on each other. One might think, once we construct the corresponding graph for this social network, it is easy to determine related nodes with respect to a particular node, just by looking at its neighbours in the graph. We should note that the concept of similarity/relationship that we target in this thesis is more sophisticated than just being directly connected to an object in the social network. Moreover, we are interested in discovering the hidden (non-obvious) ties between entities which are based on the structure of the network and the interactions among nodes. For example, even though author A might not have published any paper in conference C (i.e there is no direct link between these two individuals in the social network), they may be related because: 1) author A is working closely with other researchers who publish in conference C. 2) author A is working in areas close to the topics of conference C.

This notion of similarity/relationship is not understood just by looking at the neighbours of nodes in social networks. In the previous chapter we explained our ranking approach for k-partite graphs which finds the top similar/related individuals in k-partite graphs with respect to a particular instance. Developing a framework called "DB-Connect", in this

chapter, we construct an academic social network based on the data from DBLP, then we apply our ranking approach on the tripartite graph of this social network to find top most related entities based on each other.

5.1 Dataset

We choose DBLP as our major data source to investigate our ranking approach on a real academic social network. DBLP is a public data source which provides bibliographic information on major computer science journals and proceedings. Being publicly available, DBLP dataset has been used in many studies as the major dataset to investigate different ideas. An advantage of DBLP is that the entities are explicitly defined in the dataset. This significantly helps building the underlying social network for DBLP. Considering the information available for each publication, DBLP is a heterogeneous multiple relation social network. There are different entities such as: *authors, proceedings, publications*; and there are various relationships such as: an author *publishes* a publication, an author *participates* in a conference, etc. Since we can represent this social network with a k-partite graph, it is a good choice to apply our ranking approach. Also a part of the evaluation of the ranking results for this social network could be done based on the existing background knowledge of the academic domain. For example, it is clear that *KDD* is a conference in *data mining* so we do not expect a researcher in *mobile wireless communication* to be ranked high for *KDD*.

Aside from having a web interface to browse the bibliographic information online, DBLP provides an XML version of its database which is updated frequently. Objects are listed by different tags in the XML file[4]. A version of the XML file has been downloaded on March 13^{th}, 2011. Obviously our experiments do not contain any data added to DBLP after this date.

There are $911,126$ conference papers and $447,628$ journal publications in this XML file. While the oldest publication in this dataset is from year 1936, most of them are only a few years old. In fact 93% of publications are published between 1990 to 2011. Therefore, we filter those publications earlier than 1990, mostly because they act like noise in our dataset. Since the total number of publications left is still large (i.e $1,259,597$), we also extract a smaller version of this dataset. This smaller dataset helps us to rank the entities faster (mostly for test experiments). To do this, we select 150 top conferences based on

[4]For a complete description of the XML file see http://dblp.uni-trier.de/xml/docu/dblpxml.pdf

their number of publications. We also filter publications earlier than 1997. Finally we end up with $248,673$ publications. Table 5.1 illustrates details of these two datasets. Note that all the reported results in Chapter 6 are from *Small Dataset* unless it is explicitly mentioned otherwise.

	Publication	Author	Conference/Journal	Period
Small Dataset	248,673	263,375	150	1997-2011
Large Dataset	1,259,597	822,456	5,593	1990-2011

Table 5.1: Datasets

To complete our data and to construct a more accurate social network, we also use the information fetched from *Microsoft Academic Search*[5] and *CiteSeer*. These websites do not provide an API on their databases. Therefore, all the required information was fetched by querying the web interface. We explain more details about this in the next section.

5.2 Social Network Construction

To construct the social network for DBLP, we first have to detect instances of "Author", "Proceeding", and "Topic". Next, we need to assign the relationships among them. Identifying "Author" and "Proceeding" in DBLP is straightforward. Analyzing journal and conference publications in DBLP XML file provides us the required information. We extract the instances of Authors and Proceedings by their names. Intuitively, each author has a relationship with a proceeding if that author has published a paper in that proceeding. Also, some extra relationships can be extracted. For example all the authors of a paper are co-authors.

DBLP does not explicitly mention the topics. We should note that finding scientific topics is a research problem which many studies have tried to propose solutions for (e.g. [2], [24], [44], [41]). However, the primary focus of this thesis is to rank entities in social networks not to identify them (i.e. we do not focus on finding topics of interest but to rank them). Furthermore, we use a simple, yet effective, approach to find the topics of publications. Detecting topics for publications enables us to discover both the Topic-Author relationship and Proceeding-Topic relationship.

Based on keywords used in the title of a publication, one can guess that publication lies within which area. For example by looking at the title: "Modeling Multi-step Relevance

[5] http://academic.research.microsoft.com/

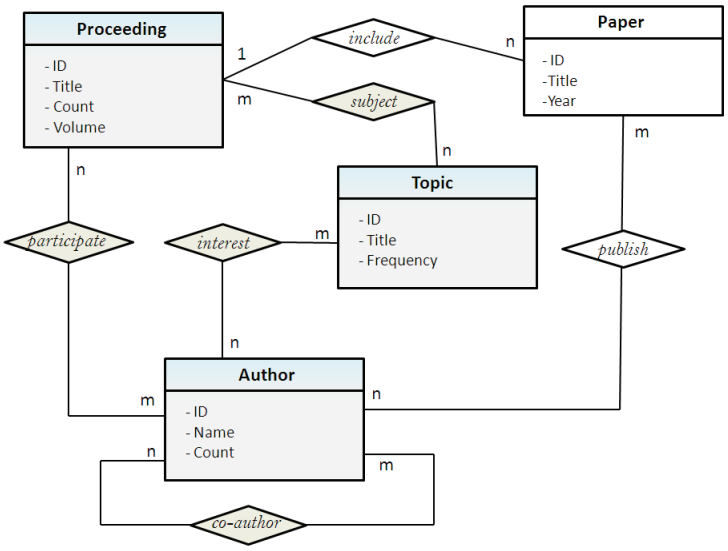


Figure 5.2: Initial class diagram for academic domain

Propagation for Expert Finding", it is clear that this publication is related to *Expert Finding* and *Relevance Propagation*.

Using the title (and when it is available the abstract) of publications, we find the most frequent bi-grams and consider them as topics. We provide a manual black list to skip meaningless frequent words such as: *also, any, of, is, for, but,* etc. We choose bi-grams because most of the topics in computer science[6] consist of two words. For example: Data Mining, Genetic Algorithm, Information Retrieval, Machine Learning, Sensor Networks, etc. Also, we assume when two overlapping bi-grams are frequent (e.g. "Hidden Markov" and "Markov Model"), the tri-gram is frequent as well. Therefore we detect several tri-gram topics such as Hidden Markov Model, Support Vector Machine, Natural Language Processing, World Wide Web, Association Rule Mining, etc.

After fixing the instances of "Topic", we find the relationship between topics and authors and proceedings. The procedure is straightforward: the topic(s) found for any publication, are assigned for author(s) of that publication and for its proceeding as well.

Figure 5.2 shows the data model for this domain. The important entities and relationships are coloured. Each proceeding *includes* one or more paper(s). Each author *publishes* one or more paper(s). Each paper is *published* by one or more author(s). Each author *participates* in one or more proceeding(s). Each proceeding hosts one or more authors. Any author may *co-author* with one or more other authors. Each author may be *interested* in

[6]Recall that DBLP has the bibliographic information on computer science journals and conferences

one or more topic(s). Each topic is the *subject* of one or more proceeding(s) or the *interest* of one or more author(s).

5.3 DB-Connect

To investigate the proposed ranking approach in Chapter 4, on the academic social network of DBLP, we develop a framework called "DB-Connect". DB-Connect is a tool with a web interface to navigate through authors, topics, and proceedings. Applying our ranking approach, DB-Connect provides the top most related entities for every instance of the network. For example, for a specific conference, it determines the most related topics of that conference, the most related authors to that conference, and other similar conferences or journals to that conference. Moreover, having the co-authorship information extracted from the DBLP XML file, DB-Connect suggests new collaborators for researchers. In other words, DB-Connect finds the related researchers for a given author based on their interest and proceeding participation, then it filters the ones who already have co-authored with the given author. The remaining researchers are likely to be potential collaborators for the given author.

DB-Connect has three major modules: 1) Preparation 2) Calculation 3) Interface. The backbone modules (i.e Preparation and Calculation modules) are developed in *C++* to be as efficient as possible. The interface module is in *PHP*. The data is stored in a MySQL database. Figure 5.3 illustrates the architecture of DB-Connect. The DBLP XML file and the information extracted from CiteSeer and Microsoft Academic Search are the only data sources of DB-Connect. Preparation Module reads these data sources, extracts the entity and relationships among them, and finally stores the social network in the right format in *SN-Database*. Next, the Calculation Module loads the social network from *SN-Database* and starts computing the rankings. Note that the computation may be done in parallel for different entities. Calculation Module stores the ranking results in *CM-Database*. Using both databases, Interface Module maps the ranking results on the social network. Furthermore, not only this module provides a navigation tool on the social network, it shows the ranking results related to each entity.

5.3.1 Preparation Module

This module takes the DBLP XML file as input. Parsing the XML file, this module constructs the corresponding entities and the relationships among them: Author, Paper, Proceeding. More specifically, we locate $<$*article*$>$ tags for journal publications and collect

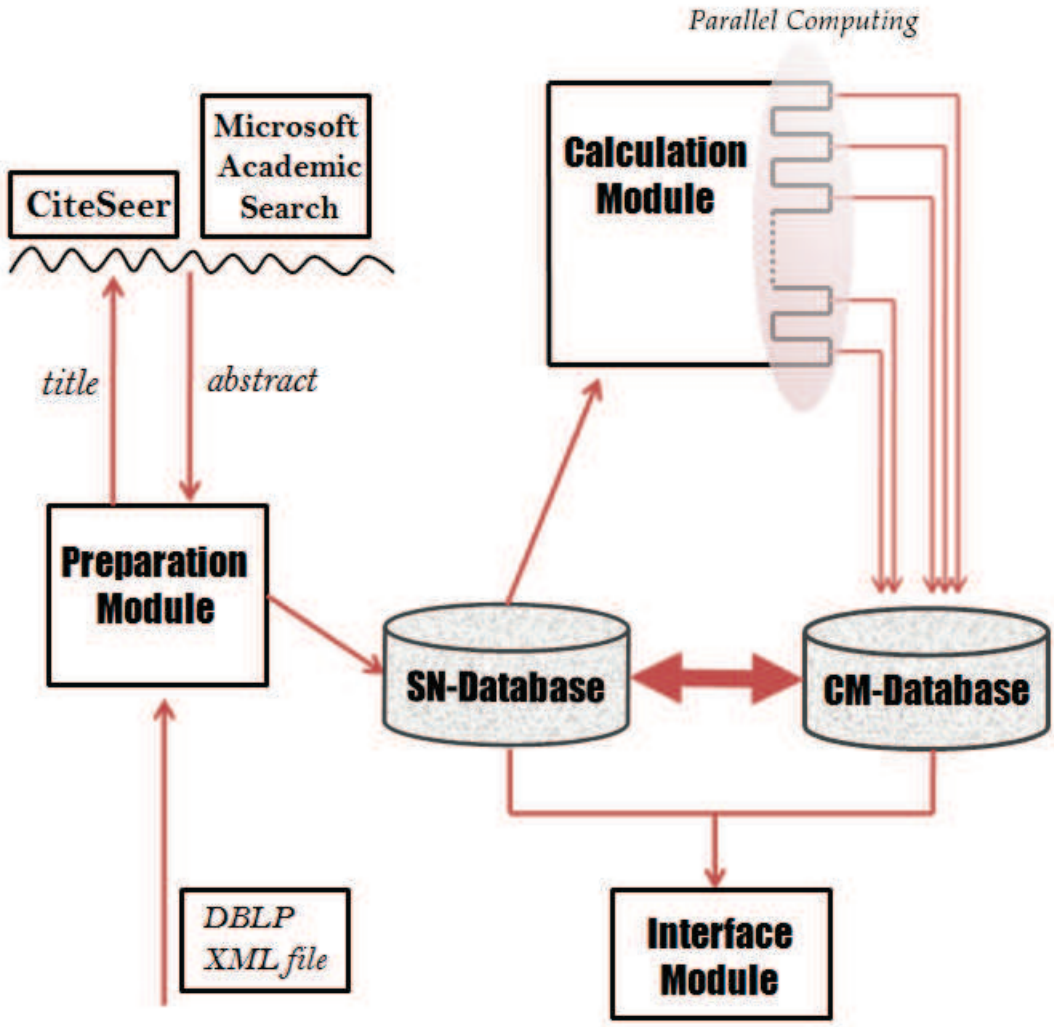


Figure 5.3: The Architecture of DB-Connect

the following attributes:

- <title>: determines the title of the paper
- <author>: determines the author(s) of the paper
- <year>: determines the publication year of the paper
- <journal>: determines the title of the journal
- <volume>: determines the volume of the journal

Also we locate *<inproceedings>* tags for conference publications. The following attributes for conference papers are collected:

- <title>: determines the title of the paper
- <author>: determines the author(s) of the paper
- <year>: determines the publication year of the paper
- <booktitle>: determines the title of the conference

Iterating over the publications (journal or conference), we construct an instance of Author/Proceeding whenever a new name is seen in <author>/<booktitle>/<journal> tag.

The next step is to generate the instances of Topic. DBLP only provides the title for publications. However, titles do not always clearly reveal the topic of the publications. Sometimes the keywords in the title of a publication do not directly point to the topic of the content. On the other hand, the abstract usually contains the gist of the publication. Fortunately there are some online services which provide the abstract of scientific publications (e.g CiteSeer, Microsoft Academic Search). Therefore, we try to collect as many abstracts as we can for the existing publications in DBLP. Not all the publications in DBLP are indexed in such online services. For this purpose we look up the title of the publications in 1) CiteSeer 2) Microsoft Academic Search. We are able to find the abstract for almost 25% of all the publications.

We choose the most frequent bi-grams in the titles of publications and the collected abstracts, as the topics. Even though finding topics based on the frequent words works well in general, there is an issue in our case. We do not know the exact number of topics. Furthermore, tuning the frequency threshold is challenging. It is not obvious how frequent must a bi-gram be to be considered as a topic. Generally, for frequency-based parameters there is no absolute value since they depend on the dataset. This means the higher the threshold is, the less the topic-candidates are. However, we assume the number of important topics in computer sciences do not exceed 2000. Furthermore, we list the valid bi-grams in a descending order of their frequency and select the top 2000 of them as topics. Unfortunately, our tests revealed that even considering this number of topics, leaves some authors (almost

25% of the authors) without any assigned topic. In other words, there are some authors with a few publications (in most cases just one) which do not have any of those selected bi-grams in the title or abstract of their publication(s). In the other hand, decreasing the threshold (i.e increasing the number of topic-candidates) with the hope of covering more keywords and eventually more publications is not a good choice since it adds too much noise to the topic set. For simplicity, we ignore those authors in our experiments. However, a possible solution to this problem is to assign the topics of the related proceedings for those authors. In other words, if author a has a publication in proceeding p but it is not assigned with any topic via his publication, we assign the related topics of p to a.

Once the topics and their relationship with other entities are found, all the entities and the relationships are stored in an Entity-Relationship (ER) database called SN-Database. The database schema is derived directly from the data model where each entity has its own table. Also each many-to-many relationship is implemented with a table, containing foreign keys to the two entities involved in that relationship. Figure 5.4 shows the database schema of SN-Database in DB-Connect.

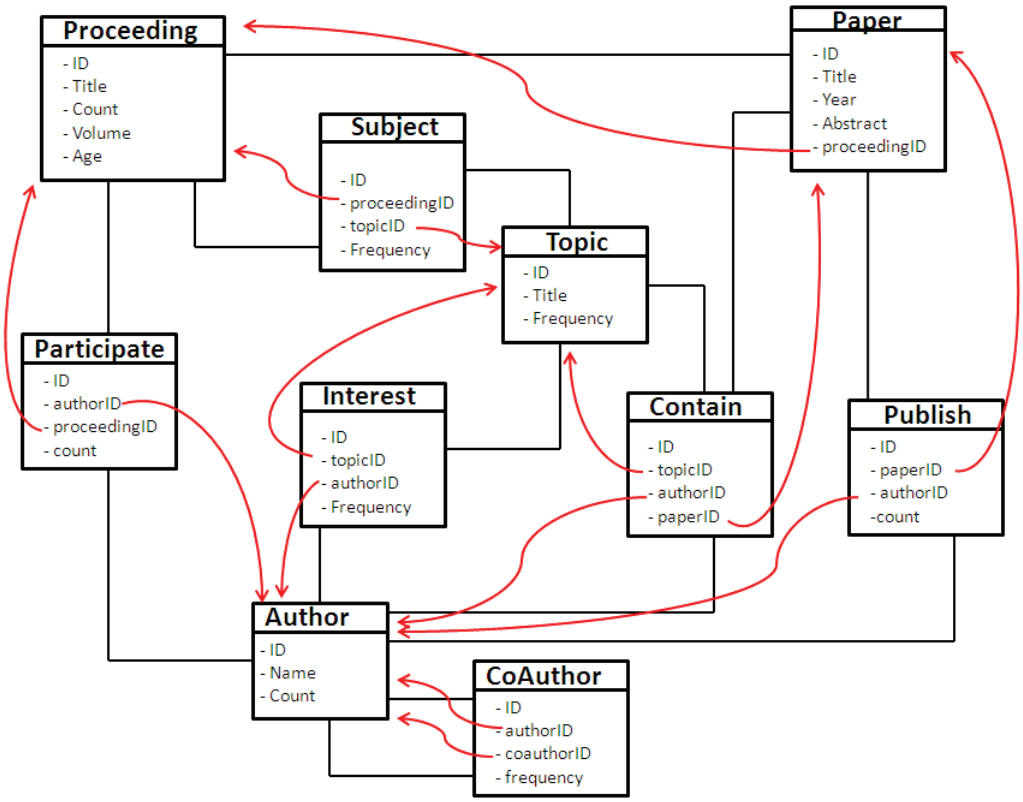


Figure 5.4: SN-Database Schema. Red arrows show foreign-keys

5.3.2 Calculation Module

Loading Author, Proceeding, and Topic from SN-Database, this module constructs the adjacency matrices of the graph, corresponding to the social network of these three entity types. More specifically, there are three partitions in the graph (tripartite graph) and they are all connected to each other. Figure 5.1 shows the graph of this network. We form the adjacency matrices of the bipartite graphs as follows:

1 M_{AT}**:** The adjacency matrix of the bipartite graph between Author and Topic where rows indicate instances of Author and columns stand for instances of Topic.

$$M_{AT} = \begin{array}{c|cccc} & t_1 & t_2 & ... & t_{s_T} \\ \hline a_1 & w_{11}^{AT} & w_{12}^{AT} & ... & w_{1s_T}^{AT} \\ a_2 & w_{21}^{AT} & w_{22}^{AT} & ... & w_{2s_T}^{AT} \\ ... & ... & ... & ... & ... \\ a_{s_A} & w_{s_A1}^{AT} & w_{s_A2}^{AT} & ... & w_{s_As_T}^{AT} \end{array}$$

where s_A is the number of authors, s_T is the number of topics, and w_{ij}^{AT} is the normalized number of different publications by author a_i which include topic t_j (i.e $\sum_{j=1}^{s_T} w_{ij}^{AT} = 1$ where $i = 1, 2, ..., s_A$).

2 M_{AP}**:** The adjacency matrix of the bipartite graph between Author and Proceeding where rows indicate instances of Author and columns stand for instances of Proceeding.

$$M_{AP} = \begin{array}{c|cccc} & p_1 & p_2 & ... & p_{s_P} \\ \hline a_1 & w_{11}^{AP} & w_{12}^{AP} & ... & w_{1s_P}^{AP} \\ a_2 & w_{21}^{AP} & w_{22}^{AP} & ... & w_{2s_P}^{AP} \\ ... & ... & ... & ... & ... \\ a_{s_A} & w_{s_A1}^{AP} & w_{s_A2}^{AP} & ... & w_{s_As_P}^{AP} \end{array}$$

where s_A is the number of authors, s_P is the number of proceedings (journals + conferences), and w_{ij}^{AP} is the normalized number of times author a_i has participated in proceeding p_j (i.e $\sum_{j=1}^{s_P} w_{ij}^{AP} = 1$ where $i = 1, 2, ..., s_A$).

3 M_{TP}**:** The adjacency matrix of the bipartite graph between Topic and Proceeding where rows indicate instances of Topic and columns stand for instances of Proceeding.

$$M_{TP} = \begin{array}{c|cccc} & p_1 & p_2 & ... & p_{s_P} \\ \hline t_1 & w_{11}^{TP} & w_{12}^{TP} & ... & w_{1s_P}^{TP} \\ t_2 & w_{21}^{TP} & w_{22}^{TP} & ... & w_{2s_P}^{TP} \\ ... & ... & ... & ... & ... \\ t_{s_T} & w_{s_T1}^{TP} & w_{s_T2}^{TP} & ... & w_{s_Ts_P}^{TP} \end{array}$$

where s_T is the number of topics, s_P is the number of proceedings (journals + conferences), and w_{ij}^{TP} is the normalized number of times topic t_i has been a subject of proceeding p_j (i.e $\sum_{j=1}^{s_P} w_{ij}^{TP} = 1$ where $i = 1, 2, ..., s_T$).

Next, we apply the ranking approach, discussed in Chapter 4, for each node to find the related instances of each class in the network with respect to That node. The top related instances for each node, are stored in CM-Database.

Note that the ranking for each individual can be done separately. Furthermore, the calculation module is multi-threaded to make the ranking task more efficient. The more the number of threads, the faster the whole module finishes ranking. The theoretical limit for number of threads would be the number of entities in the network (i.e in our dataset more than a million). Therefore, to compute the rankings for the whole dataset, we use services from a *High Performance Computation* (HPC) facility called *WestGrid*.

Western Canada Research Grid: WestGrid

WestGrid is one of the seven partner consortia that make up Compute Canada, a national platform that integrates HPC resources across the country to create a dynamic computational resource. Compute Canada brings together HPC, data resources and tools, and academic research facilities around the country.

DB-Connect stores and loads its data into a MySQL database. The only server on WestGrid providing this service is *Bugaboo*. Running the *Scientific Linux* operating system, Bugaboo is a Dell blade cluster which is comprised of 10 chassis, each containing 16 8-core blades, for a total of $1,280$ cores. Each compute node contains two sockets, with each socket containing an *Intel Xeon* $E5430$ quad-core processor, running at 2.66 GHz. Each blade has 16 GB of memory that can be shared among the 8 cores on that node. In addition to the login server (i.e head node), there is a file server called *Bugaboo-fs*. The MySQL database is located on this file server. For more information about Bugaboo please visit $http://www.westgrid.ca/support/quickstart/bugaboo$.

To use this facility, one have to submit the task as a job to the head node. Therefore, after setting up the database on Bugaboo-fs, we provide a script describing the detail specification of the job and submit it to the head node. There is an automatic scheduler running on the head node which manages all the submitted jobs. Generally, the more resources a job requires, the later the job is qualified to be run (i.e it has to wait in the queue for a longer time).

Currently, DB-Connect is running on the cluster using thirty-two (32) cores with the maximum memory usage of four GigaBytes (4GB). Detail information on the job submission protocol can be found on $http://www.westgrid.ca/support/running\_job$. DB-Connect has been designed in a way to keep track of the entities which are ranked. Therefore if any interruption occurs during the calculation, DB-Connect is able to continue the computation from where it left.

5.3.3 Interface

This module is the set of PHP pages which access the result of Calculation Module. Combining the basic information of entities from SN-Database and the ranking results from CM-Database, this module provides a profile page for each individual in the network. (i.e every instance of Author, Topic, and Proceeding has its own page. In addition to the basic information for that entity, the page gives the top related instances of other entities in the network. So for example in the page of a conference, there are top related researchers and top related topics with respect to that conference. There are also top similar other conferences or journals to that particular conference. Figure 5.5 shows some screenshots from the interface of DB-Connect.

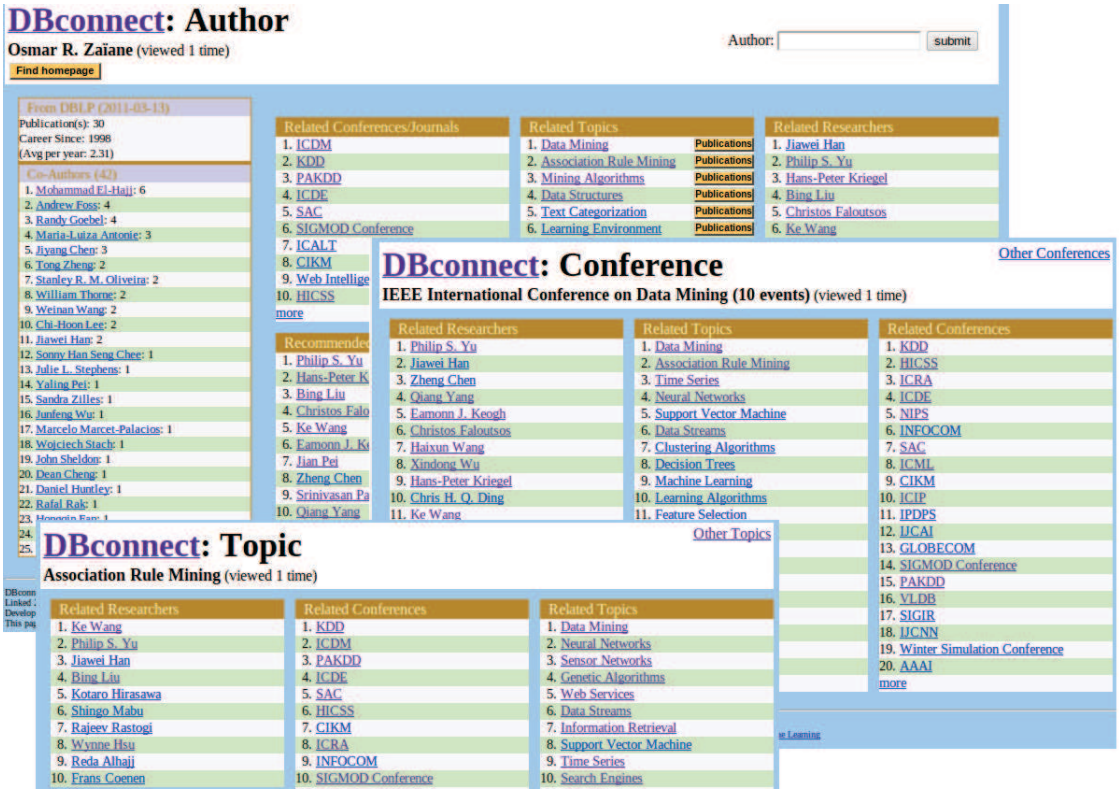


Figure 5.5: The interface of DB-Connect

In this chapter, we explained academic social networks as one possible application for our ranking approach. We discussed how we construct the social network for our case study. Also, we introduced DB-Connect as a framework for applying our approach to the academic social network. In the next chapter we report the results from DB-Connect. We compare the results from DB-Connect to similar available systems for a selective set of entities. In addition, we implement a blind comparison between our ranking approach and the method introduced by Zaiane et al. [57] based on their results on the same dataset. We also show that the idea of parallelizing the computation works in practice.

Chapter 6

Experimental Results

In the previous chapter, we explained that we develop DB-Connect to apply our ranking approach on the academic social network of DBLP. In this chapter we report the results from DB-Connect. We also compare the results from DB-Connect to similar available systems for a set of selected entities. In addition, we implement a blind comparison between our ranking approach and the method introduced in [57], based on their results on the same dataset.

6.1 Ranking Results for Authors

Figure 6.1 shows a screenshot from DB-Connect. This page shows an author profile which belongs to Jiawei Han, one of the famous researchers in data mining and databases. Note that Jiawei Han is selected arbitrarily just to describe different parts of author profile. On the top left corner, it indicates this object is an *Author*. Clicking on any of the hyper-linked names will generate a page for that selected instance whether it is an author, a topic, or a conference/journal. The box on the left contains some information about the given author such as the number of his publications, the duration of his academic career, the total number of different researchers who have co-authored with the given author, and the top co-authors based on the number of publications they have co-authored together.

Applying our ranking approach with the given author set as the starting point, we find the top related entities of each type with respect to it. There are six tables for each given author[1] which we explain in detail. Items in each table are ordered by their descending relevance scores to the given author.

[1] In some cases the last two tables might be empty because the rankings are one-way

Figure 6.1: DB-Connect screenshot for Jiawei Han

6.1.1 Related Conferences/Journals

This table sorts the top related proceedings for the given author. That means the high score proceedings from the random walk on *Author-Proceeding*, are combined with the high score proceedings obtained from the random walk on *Author-Topic-Proceeding* using the weighted union operator. Note that this is not necessarily the list of the proceedings that the author has already published in them. In fact, we expect this list to show those proceedings with the following criteria:

- Proceedings which contain most of the given author's publications
- Proceedings which have close topics to the given author's interests
- Proceedings which are attended by researchers who are close to the given author
- Important and significant proceedings

6.1.2 Related Topics

This table depicts the top related topics with respect to the given author. To make this list, we merge the high score topics from the random walk on *Author-Topic* with the high score

topics from the random walk on *Author-Proceeding-Topic*. We expect this list to show the topics with the following criteria:

- Common topics in the given author's publications
- Topics which are close to the topics of the related proceedings to the given author
- Topics which are mostly used by researchers who are close to the give author
- Frequently used topics

The author may or may not have a publication in either of these topics. If the author has at least one publication on a topic, there is a button in front of that topic in the table, which directs the user to another page showing the list of publications by the author on that topic. For example, Figure 6.2 shows a part of the publication list of on *Data Streams* by *Jiawei Han*.

DBconnect: Publications

Jiawei Han on Data Streams (17 found)

1. "Addressing Concept-Evolution in Concept-Drifting Data Streams." (Mohammad M. Masud, Qing Chen, Latifur Khan, Charu C. Aggarwal, Jing Gao, Jiawei Han, Bhavani M. Thuraisingham) at ICDM, 2010
2. "Classification and Novel Class Detection in Data Streams with Active Mining." (Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham) at PAKDD (2), 2010
3. "Classification and Novel Class Detection of Data Streams in a Dynamic Feature Space." (Mohammad M. Masud, Qing Chen, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham) at ECML/PKDD (2), 2010
4. "A Multi-partition Multi-chunk Ensemble Technique to Classify Concept-Drifting Data Streams." (Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham) at PAKDD, 2009
5. "Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams." (Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham) at ECML/PKDD (2), 2009
6. "A Practical Approach to Classify Evolving Data Streams: Training with Limited Amount of Labeled Data." (Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham) at ICDM, 2008
7. "Stream Sequential Pattern Mining with Precise Error Bounds." (Luiz F. Mendes, Bolin Ding, Jiawei Han) at ICDM, 2008
8. "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions." (Jing Gao, Wei Fan, Jiawei Han, Philip S. Yu) at SDM, 2007
9. "Finding frequent items in data streams using hierarchical information." (Xiaoyu Wang, Hongyan Liu, Jiawei Han) at SMC, 2007
10. "On Appropriate Assumptions to Mine Data Streams: Analysis and Practice." (Jing Gao, Wei Fan, Jiawei Han) at ICDM, 2007
11. "Error-Adaptive and Time-Aware Maintenance of Frequency Counts over Data Streams." (Hongyan Liu, Ying Lu, Jiawei Han, Jun He) at WAIM, 2006
12. "A Framework for Projected Clustering of High Dimensional Data Streams." (Charu C. Aggarwal, Jiawei Han, Jianyong

Figure 6.2: DB-Connect screenshot for Publications by Jiawei Han on Data Streams

6.1.3 Related Researchers

This table shows the list of top related researchers with respect to the given author. This list is generated using the results of two previous tables. That means, we compute the top related authors using both "Related Conferences/Journals" and "Related Topics". Merging the high score authors from the random walk on *Proceeding-Author* and *Topic-Author* with the weighted union operator, we come up with the final list of top related authors. Note that these authors may or may not be co-author with the given author. To be more precise, we expect this list to show the researchers with the following criteria:

- Researchers who are mostly working on topics close to the related topics of the given author
- Researchers who are mostly publishing in the proceedings related to the given author
- Researchers who are well-known in their field

6.1.4 Recommended Collaborators

This table is the list of potential collaborators for the given author. Removing co-authors from "Related Researchers", we come up with the list of related researchers who have not collaborated with the author yet. Since these researchers are closely related to the author but they have not co-authored yet, we suggest them as the potential collaborators for the author.

6.1.5 Recommended To

This table shows the researchers who have the given author in their "Recommended Collaborators". In fact the relevance between two instances (e.g two researchers) is not a two-way relationship. For example, even though *Jiawei Han* has been recommended to *Raghu Ramakrishnan* as a potential collaborator, *Raghu Ramakrishnan* is not recommended to him. Note that this table might be empty since the given author is not necessarily recommended as a potential collaborator to the other researchers.

6.1.6 Symmetric Recommendations

This table depicts the list of researchers who have a two-way relevance relation with the given author. It means they are recommended to the given author while they also have him/her on their "Recommended Collaborators" list. For example, *Christos Faloutsos* is

recommended to *Jiawei Han*, at the same time *Jiawei Han* is recommended to him. This puts *Christos Faloutsos* in the "Symmetric Recommendations" in *Jiawei Han*'s profile. Note that if the given author is not recommended to anybody (i.e "Recommended To" is empty), this table does not have any items.

6.2 Ranking Results for Topics

Figure 6.3 depicts a screenshot of DB-Connect corresponding to a topic profile. The given topic is "Retrieval Systems". Note that this topic is selected arbitrarily just to present the different parts of topic profile. On the top left corner, it indicates this object is a *Topic*. Each item on the page (whether it is an author, a topic, or a proceeding) is a hyper-link to the corresponding profile page for that entity.

Figure 6.3: DB-Connect screenshot for Retrieval Systems

Setting the starting point of our ranking approach to the given topic, we find the top related entities of each type with respect to that topic. There are three tables corresponding to the three entity types which are ranked for the given topic. Items in each table sorted based on descending order of their relevance scores to the given topic.

6.2.1 Related Researchers

This table lists the top related researchers with respect to the given topic. More specifically, the high score authors from the random walk on *Topic-Author* are combined with the high score authors from the random walk on *Topic-Proceeding-Author* using the weighted union operator. We expect this list to show the researchers who are mostly working on this topic and are close to the other researchers in this topic. Also they more likely publish in conferences/journals which are related to the given topic.

6.2.2 Related Conferences/Journals

This table sorts the top related proceedings for the given topic. The high score proceedings from the random walk on *Topic-Proceeding* are combined with the high score proceedings from the random walk on *Topic-Author-Proceeding* using the weighted union operator. We expect this list to show the proceedings which contain many publications in this topic. Also if many researchers, interested to the given topic, publish in a proceeding, we expect that proceeding to show up in this list.

6.2.3 Related Topics

This table depicts the top related topics with respect to the given topic. This list is generated based on "Related Researchers" and "Related Conferences/Journals". In fact, we merge top score topics from the random walk on *Author-Topic* and *Proceeding-Topic* with the weighted union operator to come up with the most related topics with respect to the given topic. We expect this list to show the topics which are mostly used with the given topic by the significant researchers in this area. Also the topics which often appear with the given topic in the lead conferences/journals in the field.

6.3 Ranking Results for Conferences/Journals

Figure 6.4 shows a screenshot from DB-Connect corresponding to a conference profile. The given conference is "Neural Information Processing System"(NIPS). Note that this conference is selected arbitrarily just to present different parts of conference/journal profile. On the top left corner, it indicates this object is a *Conference*.

Setting the starting point of our ranking approach to the given conference, we find the top related entities of each type with respect to it. Similar to the topic profile, there are three boxes corresponding to the three entity types which are ranked for the given conference.

Figure 6.4: DB-Connect screenshot for Neural Information Processing System (NIPS)

Items in each table sorted based on descending order of their relevance scores to the given conference.

6.3.1 Related Researchers

This table sorts the top related researchers with respect to the given conference. More specifically, the high score researchers from the random walk on *Proceeding-Author* are merged with the high score researchers from the random walk on *Proceeding-Topic-Author* using the weighted union operator. We expect this list to show the researchers who often publish in the given conference and researchers who are interested in topics close to the given conference's topics.

6.3.2 Related Topics

This table depicts the top related topics with respect to the given conference. The high score topics from the random walk on *Proceeding-Topic* and *Proceeding-Author-Topic* are combined using the weighted union operator. We expect this list to contain the topics which are mostly used within the given conference and are popular among researchers who publish in the given conference.

6.3.3 Related Conferences/Journals

This table lists the top related proceedings for the given conference. This list is generated based on "Related Researchers" and "Related Topics". In fact, we find high score proceedings once from the random walk on *Author-Proceeding*, and another time from the random walk on *Topic-Proceeding*. Merging these top score proceedings with the weighted union operator, we come up with the top related proceedings with respect to the given conference. We expect to see the proceedings which are popular among the related researchers to the given conference and are about the related topics to the given conference.

6.4 Discussion on the Quality of the Ranking Results

In this section, we select one instance of each entity type in our social network (i.e. a researcher, a topic, and a conference) to argue its ranking results. We select *Philip S. Yu* among the researchers because he is a well known researcher in data mining. Consequently the reader is more likely familiar with his related topics of interests, conferences, and researchers which makes the analysis of the ranking results straightforward. Also we select *Data Mining* among the topics because this dissertation mostly lies in this topic, and *KDD* among the conferences/journal because it is one the most prestigious conferences in data mining.

6.4.1 DB-Connect: Ranking of different entities for Philip S. Yu

Figure 6.5 shows *Philip S. Yu*'s profile in DB-Connect. Having 201 publications in 24 conferences, Mr.Yu is one of the most prolific researchers in our dataset. Here, we investigate the ranking results of each entity type with respect to Mr.Yu.

Conferences/Journals

A quick look into our dataset discloses that Mr.Yu has published in all the selected conferences in this list. In fact they contain most of his publications (i.e $\frac{174}{201}$ which is more than 86%). All the conferences in the list are mainly about data and knowledge mining and/or engineering. Based on Mr.Yu's home page[2] his research interests include data mining, privacy preserving publishing and mining, data streams, database systems, Internet applications and technologies. *ICDE, ICDM, KDD, CIKM, SIGMOD, VLDB, ICDCS, SAC, WWW,*

[2]http://www.cs.uic.edu/PSYu

Figure 6.5: DB-Connect screenshot: Philip S. Yu

INFORCOM are clearly lead conferences in these fields and they are listed as top ranked conferences with regard to Mr.Yu.

Topics

Having the "publication button" in front of topics, the first glance at this list reveals that Mr.Yu has publications in most of the top ranked topics (i.e *Data Streams, Data Mining, Association Rule Mining, Time Series, Decision Trees, Clustering Algorithms, Search Engines, Data Structures*). A deeper look into the dataset, discloses that $\frac{77}{201} \simeq 40\%$ of his publications contain these keywords in their titles and/or abstracts. In addition, these topics are major topics of interest for the related conferences such as *ICDE, ICDM, KDD,* etc. While Mr.Yu does not have a publication on *Sensor Networks* in our dataset[3], looking at the whole list of his publications discloses that he actually has several publications on this topic. Also, checking the top ranked conferences, they all include a number of publications on this topic (i.e 978 in total). However, Mr.Yu does not have a publication on *Neural*

[3]This results are based on the "small dataset" which only contains a portion of the real data

Networks, yet this topic is ranked high for him. To figure out why it appeared in the top rank list, we again check our dataset. First, we find that there are a number of publications, in the top related conferences to Mr.Yu, on *Neural Networks* (in *ICDCS, ICDM, KDD, CIKM, VLDB, SAC, WWW, INFOCOM*). More importantly, being located in 4563 publications, *Neural Networks* is the most frequent topic in our dataset. The large number of publications associated with this topic (i.e its general popularity), is the main reason of its appearance in the list of top ranked topics. Considering the fact that Mr.Yu has a direct relationship with 109 topics out of 400 (total number of topics in the small dataset) which means having publications on more than 27% of topics, there is a high probability for the random walker, surfing back and forth on the bipartite graph of *Author-Topic*, to eventually visit this topic. In other words, the random walker computes a high relevance score for this topic with respect to Mr.Yu. Also, this topic is reachable through *Author-Proceeding-Topic*. Altogether, *Neural Networks* shows up in the top related topics for Mr.Yu.

Authors

Investigating the quality of this list is more difficult. One might think researchers who have co-authored many publications together, are the top related ones to each other. However, we should note that our definition of relevance is a little different. We assume a researcher is more likely related to the given author if his/her topics of interests are close to the top related topics with respect to the given researcher, and his/her publications are published in proceedings close to the top related proceedings to the given researcher. Also, between researcher a_1 and a_2 who have similar situations with respect to the given author, the one with more links (the more well-known researcher) is more likely to be ranked top for the given author. Having this in mind, it is easier to analyze the results. For example the top ranked author for Mr.Yu is *Jiawei Han*. Even though there are other researchers who have co-authored more times with Mr.Yu (e.g *Haixun Wang*), *Jiawei Han* has more topics and proceedings in common with him.

Let us review the profile of *Philip S. Yu* in the existing online systems for academic social networks. Note that these systems only have profiles for researchers. However, DBConnect provides a profile for all the entity types (i.e Authors, Topics, Conference).

6.4.2 Microsoft Academic Search: Philip S. Yu

Figure 6.6 depicts a screenshot from Microsoft Academic Search showing Mr.Yu's profile. In addition to showing the basic information about the given author (i.e number of publica-

tions, number of citations, list of publications, etc.), Microsoft Academic Search provides a navigation tool in the academic social network. For example for the given author, it lists all the conferences/journals that the researcher has published in them. It also provides the list of most frequent keywords used in the researcher's publications such as *Data Mining, Data Stream*, and *Association Rule*.

Figure 6.6: Microsoft Academic Search screenshot on May 5^{th}, 2011: Philip S. Yu

Although Microsoft Academic Search sorts the entities of each type based on the citation number, there is no notion of ranking entities based on each other. In other words, not only the user can not rank different entity types for each other (e.g ranking conferences for an author), the user is not able to rank the entities of the same type based on one specific instance (e.g ranking topics for a specific topic). However, Microsoft Academic Search detects some topics of interests for researchers. In Figure 6.6, *Data Mining, Databases,*

Distributed and Parallel Computing are lists as interests of Mr.Yu. The Microsoft Research team does not disclose how they obtain these topics. However, we believe a hierarchy has been built by experts which contains broad topics and the common keywords in each. Then based on the frequent keywords in a given researcher's publications, they guess the topics of interests for the researcher.

6.4.3 ArnetMiner: Philip S. Yu

Figure 6.7 depicts Mr.Yu's profile from ArnetMiner. ArnetMiner gathers all types of information about an entity from the web to make a profile for the entity. In this case, the basic information about Mr.Yu such as: position, affiliation, address, phone, fax, email, homepage is provided. Applying the topical model explained in [48], ArnetMiner detects the topical aspects of the given researcher. The "Research Interest" section shows these topics: *Data Mining, Data Streams, Data Mining Techniques*. Finally, ArnetMiner provides a list of entities of each type which it believes they are related to the given researcher. The box at the right side of the page shows related researchers, topics of interests, and conferences.

- Others[Researchers]: *Kun-Lung Wu, Charu C. Aggarwal, Ming-Syan Chen, Haixun Wang, Joel Wolf*
- Expertise: *Data Mining, XML Data, Data Mining / Query Processing, Real-Time Systems / Automated Software Test Data, Software Engineering / Business Maintenance Model, Parallel Algorithms / Wormhole Networks*
- Conferences: *IEEE Trans. Knowl. Data Eng, ICDE, ICDM, KDD, SDM, ICDCS, CIKM*

Note that the related entities in these lists are purely obtained from the constructed graph for the social network. In the other words, there is no notion of ranking in here. ArnetMiner only crawls the web to find entities and the relationships among them to build the corresponding Entity-Relation (ER) model of the social network. Therefore, the related researchers for Mr.Yu are simply his co-authors. The expertise is the frequent keywords used in his publications (i.e there is at least a publication by the given researcher using that keyword). Finally, the related conferences are the ones which the researcher has published in them.

Figure 6.7: ArnetMiner screenshot on May 5^{th}, 2011: Philip S. Yu

DBLife: Philip S. Yu

Figure 6.8 depicts the screenshot from DBLife corresponding to Mr.Yu's profile. DBLife provides some general information about the given researchers such as: the top images from Google Images Search, the homepage, number of citations, the list of publications from DBLP. Similar to the other online services, DBLife lists other related entities for the given researcher on the right-side panel.

- Related People: *Jian Pei, Ling Liu, Shaoping Chen, Bing Liu*
- Related Topics: *data mining, knowledge discovery and data mining, knowledge discovery, mobile computing*

- Services: *KDD 2009 (PC), SIGMOD 2008 (PC), SIGMOD 2008 (Chair), KDD 2007 (PC)*
- Related Organizations: *University of Illinois at Urbana-Champaign, Microsoft Research, National University of Singapore, Carnegie Mellon University*
- Panels: *ICDE 2009. VLDB 2002*
- *SIGMOD 2010, KDD 2006*

Looking at the related entities, we observe that they are based on the found relationships on the web. In other words, there is no ranking done for the given researcher.

Figure 6.8: DBLife screenshot on May 5^{th}, 2011: Philip S. Yu

6.4.4 DB-Connect: Ranking of different entities for *Data Mining*

Figure 6.9 shows the profile in DB-Connect corresponding to *Data Mining*. Being located in $2,531$ publications, *Data Mining* is the 5^{th} top frequent topic in our dataset. Here, we separately investigate the ranking results of each entity type with respect to this topic.

DBconnect: Topic Other Topics

Data Mining (viewed 1 time)

Related Researchers	Related Conferences	Related Topics
1. Jiawei Han	1. KDD	1. Neural Networks
2. Philip S. Yu	2. ICDM	2. Sensor Networks
3. Hans-Peter Kriegel	3. HICSS	3. Genetic Algorithms
4. Bing Liu	4. SAC	4. Web Services
5. Christos Faloutsos	5. ICDE	5. Association Rule Mining
6. Eamonn J. Keogh	6. PAKDD	6. Data Streams
7. Gagan Agrawal	7. IPDPS	7. Information Retrieval
8. Xindong Wu	8. GECCO	8. Time Series
9. Charu C. Aggarwal	9. SIGMOD Conference	9. Search Engines
10. Ke Wang	10. ICRA	10. Support Vector Machine
11. Srinivasan Parthasarathy	11. INFOCOM	11. Machine Learning
12. Jian Pei	12. CIKM	12. Data Structures
13. William Perrizo	13. VLDB	13. Learning Algorithms
14. Heikki Mannila	14. ICIP	14. Social Networks
15. Wynne Hsu	15. NIPS	15. Mobile Robots

Figure 6.9: DB-Connect screenshot: Data Mining

Researchers

The first step to evaluate this list is to check these researcher's homepages. Not surprisingly, all the selected researchers in the list have *Data Mining* as their research interest. Nevertheless, this topic is a broad topic used by many researchers. Therefore, having *Data Mining* as a research interest for a top ranked researcher with respect to this topic, is necessary but not enough. In fact, there are $3,707$ researchers using the topic *Data Mining* in their publications in our dataset. However, only a small number of them have more than a few publications on this topic (e.g only 6% of these researchers have more than 5 publications about *Data Mining*). Knowing this fact and checking the frequency of usage of this topic only by the high ranked researchers, we observe a significant number. More specifically, almost 20% of the usage of *Data Mining* is just by the selected researchers as the top ranked for this topic. In addition to that, checking the ranking results for these researchers, *Data Mining* is ranked first or second most related topic for all of them. We can also verify the results by considering the *Topic-Proceeding-Author* path. The top ranked researchers to *Data Mining* publish in 75 conferences. Not only all these conferences have publications

on *Data Mining*, more than 90% of the usage of topic *Data Mining* is in these conferences (i.e $2,301$ out of $2,531$).

Using *Microsoft Academic Search*, we realize the top ranked researchers for *Data Mining* are the most cited researchers in this field. For example *Jiawei Han, Philip S. Yu, Hans-Peter Kriegel, Bing Liu, Christos Faloutsos, Eamonn J. Keogh, Xindong Wu, Charu C. Aggarwal, Jian Pei* are among the top 80 most cited researchers in *Data Mining*. Also, based on the same source, these researchers have high *H-index* values which show their importance in their field.

Conferences/Journals

Checking the list of conferences for *Data Mining*, we observe prestigious conferences in this topic such as:

- *KDD*: Knowledge Discovery and Data Mining
- *ICDM*: IEEE International Conference on Data Mining
- *SAC*: ACM Symposium on Applied Computing
- *ICDE*: International Conference on Data Engineering
- *PAKDD*: Pacific Asia Conference on Knowledge Discovery and Data Mining
- *SIGMOD Conference*: ACM SIGMOD Conf on Management of Data
- *VLDB*: Very Larg Data Bases

Looking into our dataset, *Data Mining* is used $2,531$ times in 117 conferences/journals. However, this topic is used $1,723$ times in the top 20 related conferences. This means only the top 20 ranked conferences for *Data Mining* contain almost 70% of its usage over all the other conferences. At the same time, there are conferences in the list which do not look related enough to *Data Mining* at the first glance. For example one might ask why *Hawaii International Conference on System Sciences* (HICSS) is ranked high for *Data Mining*. Checking the conferences having publications in *Data Mining*, we observe that *HICSS* is the 6^{th} top conference out of 117 (after *KDD, ICDM, PAKDD, ICDE, SAC*) containing the most number of publications on this topic. Also *HICSS* is the second most central conference in our dataset. Having $6,797$ publications, *HICSS* has the most number of publications after *ICRA* (with $9,193$ publication). The centrality among other conferences increases the chance of *HICSS* to be ranked high for *Data Mining*.

Topics

Verifiying this list requires more dedication. Most of the selected topics in this list are known as related fields to *Data Mining* such as: *Sensor Networks, Data Streams, Information Retrieval, Time Series, Search Engines* or they are sub-categories of this broad topic such as: *Association Rule Mining, Data Structures, Social Networks, Learning Algorithms*. However, there are some topics such as *Neural Networks, Genetic Algorithms, Web Services* that at the first glance, do not look related to *Data Mining*. We seek the reason in our dataset. Based on the dataset these topics share a considerable number of conferences and researchers with *Data Mining*. For example, *Neural Networks* and *Data Mining* have been appeared in 100 conferences together. At the same time they have been used together by 341 researchers. Note that, a deeper look in the dataset reveals that *Data Mining* does not share as many conferences and researchers with other topics as it does with *Neural Networks*. The same idea holds for *Genetic Algorithms* and other such topics. Table 6.1 illustrates the number of times that each topic is used together with *Data Mining* in conferences and by different researchers. Note that the same statistics for other topics is significantly less than the high ranked topics. In fact the more conferences and researchers a topic shares with the given topic, the more likely it is to be ranked high for the given topic.

Data Mining	Number of Shared Researchers	Number of Shared Conferences
Neural Networks	341	100
Sensor Networks	190	86
Genetic Algorithms	206	95
Web Services	162	83
Associated Rule Mining	522	66
Data Streams	315	85

Table 6.1: Mutual usage of *Data Mining* with other high ranked topics with respect to this topic: 1) by researchers 2) in conferences

6.4.5 DB-Connect: Ranking of different entities for *KDD*

Figure 6.10 shows the ranking results for *KDD* in DB-Connect. Same as other entities, *KDD* has a profile in DB-Connect where the it represents the top ranked researchers, topic, and other conferences with respect to this conference. Here, we investigate the quality of the ranking results separately for each entity type.

DBconnect: Conference Other Conferences

Knowledge Discovery and Data Mining (14 events) (viewed 1 time)

Related Researchers	Related Topics	Related Conferences
1. Jiawei Han	1. Data Mining	1. ICDM
2. Philip S. Yu	2. Association Rule Mining	2. HICSS
3. Christos Faloutsos	3. Neural Networks	3. ICDE
4. Jian Pei	4. Time Series	4. ICRA
5. Bing Liu	5. Support Vector Machine	5. INFOCOM
6. Heikki Mannila	6. Machine Learning	6. NIPS
7. Charu C. Aggarwal	7. Social Networks	7. CIKM
8. Eamonn J. Keogh	8. Data Streams	8. SAC
9. Ravi Kumar	9. Feature Selection	9. ICIP
10. Jieping Ye	10. Decision Trees	10. ICML
11. Martin Ester	11. Learning Algorithms	11. IPDPS
12. Ke Wang	12. Search Engines	12. SIGMOD Conference
13. Wynne Hsu	13. Sensor Networks	13. VLDB
14. Vipin Kumar	14. Information Retrieval	14. IJCAI
15. Srinivasan Parthasarathy	15. Clustering Algorithms	15. PAKDD

Figure 6.10: DB-Connect screenshot: Knowledge Discovery and Data Mining (KDD)

Researchers

The first fact about these high ranked researchers is that they all have published considerable number of papers in *KDD*. In fact they are among the most publishers in this conference. $2,601$ researchers have participated $4,311$ times in *KDD*. That means the normal participation rate for the researchers is about 1.65. However, the top 15 high ranked researchers have participated 314 times in this conference. That means the participation rate for this group (i.e 20.93) is 12 times more than the average participation rate in *KDD*. In addition, the top 15 researchers in the list, work in the same fields as the topics of *KDD*. More specifically, *KDD* is related to 242 topics in our dataset. The top 15 researchers cover 191 of them. In other words, almost 80% of *KDD*'s topics is covered by these researchers.

Another interesting fact is that this list is not just the list of researchers who publish many papers in *KDD*. For example, there are researchers who are among the top participants of this conference, yet not ranked high for it. Having 20 publications in *KDD*, *Padhraic Smyth* is one of the top participants in this conference but he is not ranked high for *KDD*. Querying our dataset, we observe that even though *Padhraic Smyth* has many publications in *KDD*, he is not working as much on the hot topics of this conference. In fact, his top frequent topics are *Mixure Models, Time Series, EM Algorithms, Error Rate, Probabilistic Models, Hidden Markov Models*, etc. Also, based on the official homepage of *Padhraic Smyth*[4] his main research interests are "modeling of real-world data sets involving text, time series, images, multivariate data from areas as diverse as text mining, Web data analysis,

[4] www.ics.uci.edu/ smyth/grad.html

medical image analysis, bioinformatics, atmospheric science, remote sensing, and cognitive science", which are not hot topics in *KDD*.

Topics

A quick look at the list of high ranked topics, reveals that most of them exactly match the hot topics in *KDD*. Based on the webpage of this conference[5], we realize that topics such as *Data Mining, Association Rule Mining, Time Series, Social Networks, Data Streams, Feature Selection, Clustering Algorithms*, are popular topics of *KDD*. In the other hand, checking our data distribution confirms this list as well. Not only all the high ranked topics for *KDD* are used in the publications of this conference, they are the most frequent used topics in *KDD*. More precisely, 242 topics were located $3,546$ times in the publications in *KDD*. However, the top 15 high ranked topics cover $1,531$ usage in total. In other words, more than 43% of publications in *KDD* are about these top 15 high ranked topics. Despite, there are a few topics which are not directly of interest of *KDD*. For example, even though *Neural Networks* is not a hot topic in *KDD*, it is ranked high for this conference. The reason for this phenomenon is similar to what we explained for the appearance of *Neural Networks* in the high ranked topics for *Philip S. Yu* in Section 6.4.1. Being located in $4,563$ publications, *Neural Networks* is the most frequent topic in our dataset. The centrality of this topic makes it to be reachable thourgh many paths in the graph. Therefore it recieves a high score in the random walk on the bipartite graph of *Topic-Proceeding* and *Topic-Author*. To put it simply, although *Neural Networks* is not a hot topic in *KDD* and close conferences to *KDD*, they all include a number of publications in this topic. In fact, *Neural Networks* appeared in more than 80% of the conferences in our dataset. This constant usage of *Neural Networks* adds up and makes it relevant to many entities in the network.

Conferences/Journals

Last but not least, we analyze the high ranked conferences with respect to *KDD*. As a matter of fact, most of the well-known conferences in data mining and knowledge discovery are found in the list of related conferences such as:

- *ICDM*: IEEE International Conference on Data Mining
- *ICDE*: International Conference on Data Engineering
- *PAKDD*: Pacific Asia Conference on Knowledge Discovery and Data Mining

[5] www.kdd.org

- *SIGMOD Conference*: ACM SIGMOD Conf on Management of Data
- *VLDB*: Very Large Data Bases
- *CIKM*: International Conference on Information and Knowledge Management
- *SAC*: ACM Symposium on Applied Computing
- *HICSS*: Hawaii International Conference on System Sciences

A deeper look into the list of high ranked conferences discloses that there are also a number of well-known conferences in the area of machine learning and related topics. Indeed, there is no clear border between data mining and machine learning. Therefore, *NIPS, ICIP, ICML, IJCAI* are related to *KDD*. In the other hand, there is a few items in the list (e.g *ICRA*: International Conference on Robotics and Automation) which are not clear why they are ranked high for *KDD*. Looing into our dataset, it turned out *ICRA* shares numerous topics with *KDD* (i.e 202). Even though some top frequent topics in *KDD* are not as frequent in *ICRA*, this conference contains considerable number of publications in other frequent topics in *KDD* such as *Machine Learning, Support Vector Machine, Learning Algorithms, Hidden Markov Models, Learning Methods, Probabilistic Models*.

6.5 Parallelization

Describing our methodology in Chapter 5, we explained that our ranking approach does not need the ranking results of other entities to be able to rank with respect to a particular instance of the network. In other words, the ranking computation could be done separately for each entity. That means, it is possible to parallelize the ranking task to reduce the total computation time. To prove this idea, we perform a simple experiment. Setting the goal to compute the ranking results for 8 researchers in the big dataset (i.e more than 1.3 million nodes), we run the program where the number of threads equals $1, 2, 4, 8$. The machine to perform this task has a 64 bit Linux-Ubuntu operating system with $2 \times$ quad-core *AMD* Opteron 2.3 GHz. In each case, we average the computation time over 10 runs. Table 6.2 summarizes the performance results. Clearly the total computation time reduces by increasing the number of threads. More precisely, the total time is inversely proportional to the number of threads. We have to note that the computation time for each instance might vary based on its centrality. Furthermore, in the last case (i.e 8 threads) where each thread is dealing with one instance, the total time is the maximum computation time for an entity

over all the instances. That explains why the time is not exactly divided into half when we multiply the number of threads. All in all, as long as the appropriate resources are available, we can add more threads to reduce the total ranking time.

Number of Threads	Total Time (seconds)
1	1984
2	991
4	564
8	325

Table 6.2: Parallel Ranking: The effect of using threads on the ranking time. The task is to rank 8 researchers (i.e finding top ranked Topics, Conference, Researchers for each of them). As the number of threads become larger the total computation time decreases.

6.6 Comparing the outcome from Big Dataset with Small Dataset

Introducing our dataset in Chapter 5, we explained that we built a smaller dataset to run our experiments faster. However, we also applied our approach on the real-size dataset (i.e Big Dataset) to rank the entities of different type for each other. Analyzing the ranking results for a number of entities from Big Dataset and comparing them with the results from Small Dataset for the same entities shows a slight difference between them. For example Figure 6.11 illustrates the conference ranking results from both datasets for *Philip S. Yu*. Comparing the top related conferences, 7 out of 10 are the exactly the same with a slight change of ordering. Note that *SDM, CoRR, EDBT* replaced *SAC, WWW, INFOCOM* in this new list. Note that *SDM, CoRR, EDBT* were not in Small Dataset at all. Also, *SAC, WWW, INFOCOM* are still among the top 20 ranked conferences for *Philip S. Yu* in Big Dataset.

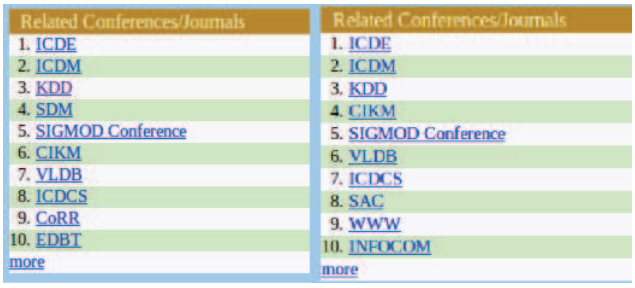


Figure 6.11: Conference comparison between the two data sets for Philip S. Yu: The left ranking is from the big dataset and the right one from the small dataset

Similarly, looking at the list of top related topics in Figure 6.12 shows a slight differ-

ence. *Clustering Algorithms, Search Engines, Data Structures* have been replaced by *Web Services, Query Processing, Database Systems*. An interesting fact is in Small Dataset, Mr.Yu did not have any publication on *Sensor Networks* yet it was ranked high for him. In the list of related topics from Big Dataset, *Sensor Networks* appears again but this time there are some publications on this topic by Mr.Yu.

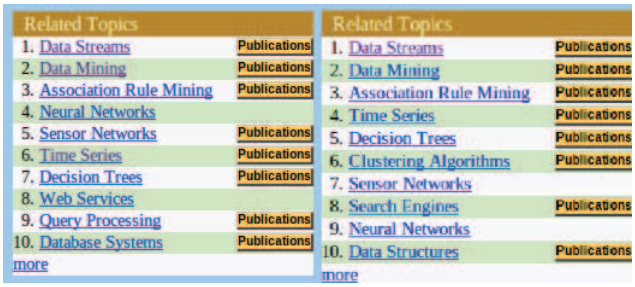


Figure 6.12: Topic comparison between the two data sets for Philip S. Yu: The left ranking is from the big dataset and the right one from the small dataset

Finally we compare the list of top related researchers from Big Dataset with the list of top related researchers from Small Dataset (i.e Figure 6.13) with respect to Mr.Yu. Most of top ranked researchers (i.e 7 out of 10) are fixed with a slight change in the ordering. *Beng Chin Ooi, Jian Pei, Kian-Lee Tan* in the results from Small Dataset are replaced by *Surajit Chaudhuri, Gerhard Weikum, Rakesh Agrawal* in the current list.

Figure 6.13: Researcher comparison between the two data sets for Philip S. Yu: The left ranking is from the big dataset and the right one from the small dataset

6.7 Blind Test

As a part of the evaluation of our approach, we compare the ranking results from DB-Connect to a similar work. Zaiane et al. [57] proposed a solution based on random walks to rank entities in the same social network. In fact, the authors developed a prototype system

called DB-Connect to apply their method. Note that the framework developed in this study only shares its name and some parts of the interface with the prototype system. In the old DB-Connect, the contribution is modeling the relational databases with an extended version of bipartite graphs (i.e adding surrogate nodes). This only covers the relationship between two entity types (i.e Author and Conference). To cover the third entity type (i.e Topic) they choose a random direction (Author $->$ Conference $->$ Topic or Author $->$ Topic $->$ Conference) for the random walk. However, having the same interface in both systems and working with the same dataset, enables us to implement a blind comparison between the ranking results.

Obviously, to judge between two ranking results with respect to a specific entity, one has to know the given entity and its relevant entities. Furthermore, we selected ten well-known researchers in the database and data mining area and developed a web interface which allows the user to choose between the two ranking results, ours and the old version. While the rankings are anonymous to the user (i.e the user does not know which output is from which system), we invited those researchers to vote between the two ranking results for all ten cases. Unfortunately, we did not receive enough votes. We believe this is mostly because those selected top researchers are busy and did not give it the time sensitive importance it deserves. In the other hand, ranking entities based on each other is certainly "subjective". Therefore, integrating different votes for a specific instance is even more troublesome. All in all, we decided to ask locally available researchers to choose between the two ranking results for themselves. In other words, we show each researcher, the ranking results with respect to him/her. Clearly, the researcher is the most authoritative person to argue the ranking result with respect to himself/herself.

Table 6.3 summarizes the outcome of this poll. In each case, the researcher is asked to choose, separately for each entity type, between the two ranking results without knowing which one is which. Evidently, the winner is the new DB-Connect which applies the proposed approach in Chapter 4 to rank the entities. Note that there are statistical measures such as *Krippendorff's Alpha* which is a statistical measure of the agreement for values of a variable or *Cronbach's Alpha* which is a coefficient of reliability. It is recommended to use these coefficients to measure the statistical importance of such tests. Unfortunately we were not able to collect enough data points to use these measures.

	Proceeding Ranking	**Topic Ranking**	**Researcher Ranking**
Dale Schuurmans	NEW	SAME	NEW
Mario Nascimento	NEW	NEW	NEW
Joerg Sander	OLD	SAME	NEW
Michael Bowling	NEW	NEW	NEW
Davood Rafiei	OLD	NEW	SAME
Csaba Szepesvari	NEW	SAME	OLD
Rich Sutton	NEW	NEW	OLD
Russ Greiner	NEW	NEW	NEW
Osmar Zaiane	NEW	NEW	NEW
total votes for DBConnect (NEW)	7	9	7
total votes for Prototype DB-Connect (OLD)	2	3	3

Table 6.3: The outcome of the poll between two ranking results from researchers

Chapter 7

Conclusion

7.1 Summary

Entities and the interactions/relationships among them compose a *Social Network*. This definition empowers a wide range of studies (e.g epidemiology, sociology, anthropology, social psychology, economics, biology, communication studies, information science, sociolinguistics, organizational studies, geography, marketing, etc) to employ social networks for modeling and further for analysing their corresponding environment. The study of structural and behavioural properties of social networks is called *Social Network Analysis* (SNA). Representing social networks by graphs where nodes are the entities and edges are the relationships, SNA discloses useful information about social networks such as: the most influential entities in the network, the hubs in the network, the groups/communities in the network, central nodes in the groups. SNA is also used to study the evolution of social networks (e.g detecting the formation, expansion, and changes in the communities within a social network). In addition, SNA may be used to rank entities in social networks. In many applications, there is the need to rank the entities in the corresponding social network. This ranking could be as simple as sorting of entities based on one specific attribute. However, in many application domains, the ranking task may be more sophisticated. For example assume that the manager of an organization wants to promote the employees based on their interactions with the clients. The more clients an employee serves and the more positive feedback the employee gets from the clients, the more bonus the employee receives. This scenario requires a ranking of employees based on their interactions with the clients. In other words, the ranking is based on the properties of the social network of the employees and clients.

Ranking can be even more complicated. Assume that a company offers variety of services/products and targets a large number of clients regularly (e.g banks, chain supermar-

kets), the manager may want to know which clients usually use a specific service or buy a specific product to make some business decisions. This means ranking of entities (i.e clients) based on one specific entity (i.e service/product) in the social network. However, ranking the entities based on each other in social networks with different entity types and different relationships, *Heterogeneous Multiple Relation Social Networks*, is not trivial because analysing and understanding the properties of such social networks is problematic.

In this dissertation, we addressed the problem of ranking entities in heterogeneous multiple relation social networks. Modeling such social networks with k-partite graphs, we propose a ranking approach based on random walks which finds the top n related instances of each entity type with respect to any node in the network. The main idea of our approach is to break the k-partite graph into several bipartite graphs, apply a random walk to generate relevance scores for the nodes with respect to the target entity, and finally to incorporate the obtained relevance scores to detect the top n related instances.

Developing a framework called *DB-Connect*, we tested our theory in practice. DB-Connect employs our ranking approach on a real heterogeneous multiple relation social network extracted from *DBLP*. Analysing the ranking from DB-Connect suggests that our results are promising. Comparing our results with other similar systems supports this conclusion.

Afterall, DBConnect may be useful in the following:

- To find the best matching reviewer for a paper submitted to a specific conference/journal.
- To suggest a venue to submit a paper based on its content and authors.
- To suggest popular topics when authors preparing for a conference.
- To find new collaborators for researchers.
- To find related conferences, topics and researchers when there is not much history for a datapoint. For example for a new researcher who has only a few publications.

7.2 Contributions

To address the thesis statement we have done the following:

1. Modeling heterogeneous multiple relation social networks with k-partite graphs, we propose an algorithm to rank all the nodes in the graph (i.e entities in the social network) based on one specific instance of the network. More specifically, using

random walks on bipartite graphs, we assign relevance scores to the nodes of the graph with respect to the target instance. Therefore we are able to sort the entities based on their scores. We prove that the proposed ranking approach is theoretically scalable for larger values of k. Developing a framework called *DB-Connect*, we apply our ranking approach to a real heterogeneous multiple relation social network (i.e tripartite graph) and argue the accuracy of the outcome.

2. The proposed ranking approach only uses the information available in the social network. In other words, it computes the ranking for each instance separately without using the results of other entities. Furthermore, ranking of different entities can be done in parallel. Designing DB-Connect to be multi-threaded, we show the parallelized ranking in practice.

3. Performing our ranking approach on a reasonable portion of the real dataset and comparing the results with the outcome of the real-size dataset, we show that there is a negligible difference between the two results. This suggests that for large datasets, where there is not enough computation resources available, one may partition the dataset into smaller pieces and apply our ranking approach, yet achieve pleasant results. Note that the partitioning should be done in a reasonable manner (e.g clustering approaches)

7.3 Challenges Left to Explore

7.3.1 Advanced Topical Model

We believe that there are more effective approaches to find the topical aspects of publications. We only used the frequency of words in the titles and abstracts of publications to determine 1) the instances of topics 2) the relationship between researchers and conference/journals with topics. The better we model the topical aspects of publications, the more exact our social network model is. For example the following ideas can be considered in a more sophisticated topical model:

- Hierarchy of topics. Having a hierarchy from broad topics to the narrow one, each publication lies within a broad topic (e.g *Data Mining*), yet is assigned with one or more specific keyword(s) (e.g Association Rule Mining, Web Mining, etc.). This would be particularly handy to detect expert researchers or specialized conferences

in those narrow topics because otherwise they are outnumbered by the large number of researchers and conferences dealing with the broad topics.

- Overlapping topics. We assumed if a publication contains one of the two frequent overlapping bi-grams (e.g *World Wide, Wide Web*), it is related to the tri-gram topic (i.e *World Wide Web*). This assumption does not always hold. Assume *Continuous Speech* and *Speech Recognition* are both frequent over all. If a publication contains *Speech Recognition*, it is not necessarily related to *Continuous Speech Recognition*.

- Topics by experts. Currently, we find the instances of topics based on the information from publications. One may ask experts to define topics for the domain. This eliminates the noisy topics. However, finding the relationship between publications and the pre-defined topics is not trivial anymore.

7.3.2 Ranking with Clustering

Applying our ranking approach to a reasonable portion of the dataset (i.e Small Dataset), we observed that the results are close to the outcome from the real-size dataset (i.e Big Dataset). This suggests that using the appropriate clustering algorithm on the large size datasets may be an effective solution to decrease the computational and time complexity of the ranking task.

7.3.3 Dynamic Tracking of the Social Network

Proposing our ranking approach, we considered a static graph corresponding to a static social network. This assumption is not based on the real world. In fact, the academic social network is always changing. There are always new publications, researchers, and even conferences/journals which have to be added to the network. In the current version of DB-Connect, after adding new nodes to the network, we have to compute the ranking for all the entities from scratch. Indeed, this task is time-consuming. One may think of heuristics to find the rankings incrementally or at least limit the number of entities in the network for which a ranking has to be recomputed.

Bibliography

[1] Small world experiment. $http://en.wikipedia.org/wiki/Small_world_experiment$.

[2] Levent Bolelli, Seyda Ertekin, Ding Zhou, and C. Lee Giles. Finding topic trends in digital libraries. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '09, pages 69–72, New York, NY, USA, 2009. ACM.

[3] Philip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[4] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555 – 564, 2007.

[5] Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55 – 71, 2005.

[6] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136 – 145, 2008.

[7] Ulrik Brandes and Daniel Fleischer. Centrality measures based on current flow. In Volker Diekert and Bruno Durand, editors, *STACS 2005*, volume 3404 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin / Heidelberg, 2005.

[8] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[9] Yuhan Cai, Xin Luna Dong, Alon Halevy, Jing Michelle Liu, and Jayant Madhavan. Personal information management with semex. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 921–923, New York, NY, USA, 2005. ACM.

[10] Antoni Calvo-Armengol and Yves Zenou. Social networks and crime decisions: The role of social structure in facilitating delinquent behaviour. CEPR Discussion Papers 3966, 2003.

[11] Kaushik Chakrabarti, Venkatesh Ganti, Jiawei Han, and Dong Xin. Ranking objects by exploiting relationships: Computing top-k over aggregation. sigmod. In *SIGMOD Conference*, pages 371–382, 2006.

[12] Jiyang Chen. *Community Mining, Discovering Communities in Social Networks*. PhD thesis, University of Alberta, 2010.

[13] Pedro DeRose, Warren Shen, Fei Chen 0002, Yoonkyong Lee, Douglas Burdick, An-Hai Doan, and Raghu Ramakrishnan. Dblife: A community information management platform for the database research community (demo). In *CIDR*, pages 169–172. www.crdrdb.org, 2007.

[14] Anhai Doan, Raghu Ramakrishnan, Fei Chen, and Pedro Derose. Community information management, 2006.

[15] Xin Dong and Alon Y. Halevy. A platform for personal information management and integration. In *Conference on Innovative Data Systems Research*, pages 119–130, 2005.

[16] Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 118–127, New York, NY, USA, 2004. ACM.

[17] Francois Fouss, Alain Pirotte, Jean michel Renders, and Marco Saerens. A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering, 2004.

[18] Franois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19:355–369, 2007.

[19] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215 – 239, 1978-1979.

[20] Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Trans. Inf. Syst.*, 21:64–93, January 2003.

[21] E. Garfield. Is citation analysis a legitimate evaluation tool? *Scientometrics*, 1:359–375, 1979. 10.1007/BF02019306.

[22] G.H. Golub, C.F. van Loan. *Matrix computations*. John Hopkins University Press, 3^e edition, 1996.

[23] Liang Gou, Xiaolong (Luke) Zhang, Hung-Hsuan Chen, Jung-Hyun Kim, and C. Lee Giles. Social network document ranking. In *Proceedings of the 10th annual joint conference on Digital libraries*, JCDL '10, pages 313–322, New York, NY, USA, 2010. ACM.

[24] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[25] Guoming He, Haijun Feng, Cuiping Li, and Hong Chen. Parallel simrank computation on large graphs with iterative aggregation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 543–552, New York, NY, USA, 2010. ACM.

[26] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[27] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, September 1999.

[28] Yehuda Koren, Stephen C. North, and Chris Volinsky. Measuring and extracting proximity in networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 245–255, New York, NY, USA, 2006. ACM.

[29] John Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann, 2001.

[30] Justin Fagnan Osmar R. Zaiane Mansoureh Takaffoli, Farzad Sangi. Modec - modeling and detecting evolutions of communities. In *Fifth International AAAI Conference on Weblogs and Social Media*, ICWSM '11, 2011.

[31] Qiaozhu Mei, Jian Guo, and Dragomir Radev. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1009–1018, New York, NY, USA, 2010. ACM.

[32] L. Meyers, M. Newman, and B. Pourbohloul. Predicting epidemics on directed contact networks. *Journal of Theoretical Biology*, 240(3):400–418, 2006.

[33] M.E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54, 2005.

[34] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji rong Wen, and Wei ying Ma. Web object retrieval. In *World Wide Web Conference Series*, pages 81–90, 2007.

[35] Zaiqing Nie, Ji rong Wen, and Wei ying Ma. Object-level vertical search. In *Conference on Innovative Data Systems Research*, pages 235–246, 2007.

[36] Zaiqing Nie, Fei Wu, Jirong Wen, and Weiying Ma. Extracting objects from the web. In *International Conference on Data Engineering*, 2006.

[37] Zaiqing Nie, Yuanzhi Zhang, Ji rong Wen, and Wei ying Ma. Object-level ranking: bringing order to web objects. In *World Wide Web Conference Series*, pages 567–574, 2005.

[38] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[39] Jordi Palau, Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. Collaboration analysis in recommender systems using social networks. In *Proceedings of the 8th International Workshop on Cooperative Information Agents*, CIA '04, pages 137–151, 2004.

[40] Augusto Pucci, Marco Gori, and Marco Maggini. A random-walk based scoring algorithm applied to recommender engines. In Olfa Nasraoui, Myra Spiliopoulou, Jaideep Srivastava, Bamshad Mobasher, and Brij Masand, editors, *Advances in Web Mining and Web Usage Analysis*, volume 4811 of *Lecture Notes in Computer Science*, pages 127–146. Springer Berlin / Heidelberg, 2007.

[41] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *In Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI)*, San Francisco, CA, 2004. Morgan Kaufmann.

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[43] Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. Modeling multi-step relevance propagation for expert finding. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 1133–1142, New York, NY, USA, 2008. ACM.

[44] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *The Tength ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[45] Gilbert Strang. *Introduction to Linear Algebra*. Wesley Cambridge Press, 2003.

[46] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Relevance search and anomaly detection in bipartite graphs. *SIGKDD Explor. Newsl.*, 7:48–55, December 2005.

[47] Pontus Svenson, Per Svensson, Hugo Tullberg, Foi Ledningssystem, Foi Ledningssystem, and Foi Ledningssystem. Social network analysis and information fusion for antiterrorism. In *In Proc. CIMI, paper S3.1*, pages 91–89410, 2006.

[48] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 990–998, New York, NY, USA, 2008. ACM.

[49] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 613–622, Washington, DC, USA, 2006. IEEE Computer Society.

[50] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Communities and technologies*, pages 81–96, 2003.

[51] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[52] Debra Aho Williamson. Social network marketing: Ad spending and usage. $http : //www.emarketer.com/Report.aspx?code = emarketer_2000478$.

[53] Jennifer Xu and Hsinchun Chen. Criminal network analysis and visualization. *Commun. ACM*, 48:100–107, June 2005.

[54] C.C. Yang and T.D. Ng. Terrorism and crime related weblog social network: Link, content analysis and information visualization. In *Intelligence and Security Informatics, 2007 IEEE*, pages 55 –58, may 2007.

[55] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20:796–808, 2008.

[56] Osmar Zaiane, Jiyang Chen, and Randy Goebel. Mining research communities in bibliographical data. In Haizheng Zhang, Myra Spiliopoulou, Bamshad Mobasher, C. Giles, Andrew McCallum, Olfa Nasraoui, Jaideep Srivastava, and John Yen, editors, *Advances in Web Mining and Web Usage Analysis*, volume 5439 of *Lecture Notes in Computer Science*, pages 59–76. Springer Berlin / Heidelberg, 2009.

[57] Osmar R. Zaiane, Jiyang Chen, and Randy Goebel. Dbconnect: mining research community on dblp data. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 74–81, New York, NY, USA, 2007. ACM.

[58] Hugo Zaragoza, Henning Rode, Peter Mika, Jordi Atserias, Massimiliano Ciaramita, and Giuseppe Attardi. Ranking very many typed entities on wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 1015–1018, New York, NY, USA, 2007. ACM.

[59] Jun Zhu, Zaiqing Nie, Jirong Wen, Bo Zhang, and Weiying Ma. 2d conditional random fields for web information extraction. In *International Conference on Machine Learning*, 2005.

[60] Jun Zhu, Zaiqing Nie, Jirong Wen, Bo Zhang, and Weiying Ma. Simultaneous record detection and attribute labeling in web data extraction. In *Knowledge Discovery and Data Mining*, 2006.