ABSTRACT

Title of dissertation: AUTHORITY FLOW-BASED RANKING IN HETEROGENEOUS NETWORKS: PREDICTION, PERSONALIZATION, AND LEARNING TO RANK

Hassan Sayyadi-Harikandehei, Doctor of Philosophy, 2014

Dissertation directed by: Professor Louiqa Raschid
Smith School of Business

Many real-world datasets, including biological networks, the Web, and social media, can be effectively modeled as networks or graphs, in which nodes represent entities of interest and links mimic the interactions or relationships among them. Such networks often contain multiple entity or relationship types, which are referred to as heterogeneous networks. Networks also evolve due to the existence of temporal features that characterize the entities or to the temporal relationships among them. Finding important/authoritative entities in real-world networks is a long-standing and well-defined challenge. In this dissertation, I focus on two variants of the problem. The first is the prediction of the ranking of scientific publications in a future state of a citation network. I introduce a new measure labeled the *future PageRank score*. I develop FutureRank, a prediction algorithm for *predicting* the future PageRank scores from the historical network structure, and evaluate the FutureRank algorithm on multiple bibliographic dataset.

Next, I focus on personalized ranking in social media. I extend a social media dataset to include relationships (edge types) between authors, blog posts, categories (topics) of the posts, and events (collections of posts). I then apply personalized ranking algorithms over the historical posts and events that have been visited by a user and use the ranking to recommend additional posts. I evaluate the personalized recommendations through an experiment with real users, as well as an extensive study of synthetic users whose preferences are defined based on intuitive criteria.

Finally, I present an approach for learning to rank (algorithms) applied to heterogeneous networks. Existing methods for learning to rank are typically limited to content-based features, while many real world problems correspond to relational features. I develop a framework for learning to rank, which targets authority flow-based ranking models on heterogeneous networks. I propose algorithms for both pointwise and pairwise learning. However, this framework can easily utilize any loss function from a non-relational learning domain. Experiments show that even with a small amount of training data, both pointwise and pairwise algorithms perform successfully and converge very fast. In addition, these solutions are shown to be robust against noise.

Authority Flow-based Ranking in Heterogeneous Networks: Prediction, Personalization, and Learning to Rank

by

Hassan Sayyadi-Harikandehei

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Acknowledgments

This thesis would not have been possible without the help, encouragement, and support of many people.

First and foremost, I thank my advisor Louiqa Raschid for her guidance and invaluable advice in all aspects of my academic life. I also thank the other members of my dissertation committee, Hal Daume III, Amol Deshpande, William Rand, and Najib M. El-Sayed for their time and comments.

During my graduate years, I had the opportunity to collaborate with many bright people whom I am grateful to. I thank Vagelis Hristidis and Lise Getoor with whom I have done some of my graduate research and coauthored several papers. I thank my other friends and coauthors Adam Lee, Matthew Hurst, Alexey Maykov, and those whom I have not listed here.

Last but not the least, I am forever grateful to my family to whom I owe everything I have achieved in my life.

Table of Contents

List of Figures v

List of Tables vii

1 Introduction 1
- 1.1 Challenges of Ranking in Evolving Networks 3
- 1.2 Challenges of Personalized Ranking in Social Media 4
- 1.3 Challenges of Learning to Rank in Heterogeneous Networks 6
- 1.4 Contributions 7
- 1.5 Outline 9

2 Related Work 10
- 2.1 Authority Flow-based Ranking 10
- 2.2 Personalized Ranking 13
- 2.3 Learning to Rank 14
  - 2.3.1 Learning to Rank in Networks 16
- 2.4 Ranking Scientific Articles 17
- 2.5 Noise and Missing Data 19

3 FutureRank: Ranking Scientific Articles by Predicting their Future PageRank 21
- 3.1 Proposed Model 22
  - 3.1.1 Network Structure 23
  - 3.1.2 FutureRank 25
- 3.2 Evaluation 27
  - 3.2.1 Dataset 27
  - 3.2.2 Evaluation Setup 28
  - 3.2.3 Ranking: Evaluation and Approaches 29
  - 3.2.4 Evaluation Metrics 31
  - 3.2.5 Parameter Estimation 33
  - 3.2.6 Further Comparison of Proposed Algorithms 35
  - 3.2.7 Running Time and Convergence 43
- 3.3 Discussion 43

4 Personalized Ranking in Social Media 47
- 4.1 Enriching Social Media Dataset 50
  - 4.1.1 Schema Graph for Personalized PR (pPR) and ObjectRank (pOR) 51
  - 4.1.2 Data Graph 51
  - 4.1.3 Computing Document Similarity for BlogPosts 52
  - 4.1.4 Identification of Events 53
- 4.2 Personalized Authority Flow-based Ranking 55
  - 4.2.1 Authority Flow-based Ranking: The ObjectRank Algorithm . 55

4.2.2 Personalized Ranking Variants 57
4.3 Evaluation 58
4.3.1 Synthetic User Experiment 59
4.3.1.1 Comparison of Ranking Algorithm Variants 61
4.3.1.2 Parameter Sensitivity Analysis 64
4.3.2 Evaluation with Real Users 66
4.4 Discussion 69

5 Learning to Rank in Heterogeneous Networks 70
5.1 Learning Framework 71
5.2 Loss Function 78
5.2.1 Pointwise Learning 78
5.2.2 Pairwise Learning 79
5.3 Evaluation 80
5.3.1 Datasets 80
5.4 Case Study 1: Evaluate the Performance of Learning Variants 80
5.4.1 Evaluation 81
5.5 Case Study 2: Learning the Parameters of FutureRank 85
5.5.1 Evaluation 86

6 Conclusion 88

Bibliography 91

List of Figures

2.1 Enhanced social media schema graph 14

3.1 Average number of citations in the dataset (arXiv), which articles obtain based on the number of years prior from publication date . . . 22
3.2 An example of the scientific article network, which has two types of nodes, papers (rectangles) and authors (circles), and two types of edges: authorship edges, which are between authors and papers, and citation links, which are between papers. 24
3.3 Network decomposition: a single mode network of citations and a bipartite network of authorship 25
3.4 Hypothetical figure of entire dataset, in which horizontal lines show the timeline and links show the citations: (a) the full dataset; the horizontal line shows the timeline and links show the citations of articles by later articles; (b) the query data, which only contain links originating before 2001; and (c) the evaluation data, which only contain links originating after 2001. 28
3.5 The precision of FutureRank for different configurations of the parameters α, β, and γ on the arXiv dataset. At any point in the figure, the value of β is equal to $1 - \alpha - \gamma$. Similar analyses on the DBLP dataset resulted in a near-identical figure. 34
3.6 Precision-recall based on precision in the top 50 results. 36
3.7 Precision of the top k results of FutureRank compared to the top k results of future PageRank 37
3.8 Percentage of the papers from each category that are among the top k output of FutureRank. 39
3.9 Correlation between ranking on the validation data and the results of CiteRank and FutureRank 40
3.10 Score convergence. The vertical axis shows the difference between the score values in two consecutive steps, so zero difference reflects the convergence 45
3.11 Precision of top 50 results after each iteration 45

4.1 Enhanced Social Media Schema Graph 50
4.2 An example of KeyGraph 54
4.3 $F1$ Values of pIR and PageRank versus ObjectRank for Author and Category users (U=50, D=6, H=30), Event users (U=15, D=1, H=30), and Keyword users (U=10, D=1, H=30). 63
4.4 $F1$ Values for Event Users with $D = 1$ and $H = 30$ and $U = 15$. . . . 63
4.5 Average $F1$ Values of pOR for 50 Author Users ($U = 50$) with Varying Values of $D = 2, 4, 6, 8, 10$ and $H = 10, 20, 30, 40, 50$. 64
4.6 Average $F1$ Values of pOR for 50 Category Users ($U = 50$) with Varying Values of $D = 2, 4, 6, 8, 10$ and $H = 10, 20, 30, 40, 50$. 64

4.7 $F1$ Values of pOR for 10 Keyword Users with Varying Values of $H = 10, 20, 30, 40, 50$. 65
4.8 Results of Experiment with Real Users 66
4.9 The interface for human user experiment 67

5.1 Spinn3r Schema Graph 73
5.2 The workflow of a learning algorithm for ObjectRank 76
5.3 The workflow of a learning algorithm proposed by Chakrabarti et al. 77
5.4 Accuracy of pointwise learning and pairwise learning. The X axis shows the size of the training data per node type and the Y axis shows the F1 for the top 50 results. 82
5.5 Convergence of pointwise learning and pairwise learning. The X axis shows the number of learning iterations and the Y axis shows the F1 for the top 50 results. 83
5.6 Performance of pointwise learning and pairwise learning with up to 50% noise in the training data. The X axis shows the percentage of noise and the Y axis shows the F1 value for the top 50 results. 83
5.7 Performance for learning FutureRank in comparison with the exhaustive search algorithm. 87

List of Tables

3.1 Top 20 articles retrieved by FutureRank that were published before 2001 (arXiv dataset) 42
3.2 Top 20 authors retrieved by FutureRank and the number of their articles in the dataset 44

Chapter 1

Introduction

Many real-world problems can be effectively modeled as networks, in which nodes represent entities of interest and links mimic the interactions or relationships among them. Node attributes represent the properties of the entities and link attributes represent the properties of the interactions or relationships. For example, a Web graph has one node per Web page, which has properties such as the page URL, title, content, etc. A directed link between two nodes represents a hyperlink from the source Web page to the target Web page, which has an attribute that stores the anchor text of the hyperlink. In a bibliographic network, there is one node per paper and one node per author. Each paper has properties such as a title, abstract, publication date, category labels, keywords, publication venue, etc., and each author has properties such as a name, email, and organizational affiliation. There are bidirectional links between authors and papers, as well as directed citation links from papers to papers. An authorship link may have an integer attribute that indicates the position of the corresponding author in the author list. Similarly, a citation link may have an integer attribute that indicates the position of the cited paper in the reference list, or an integer attribute that shows how many times the cited paper is mentioned by the citing paper. Such heterogeneous networks, which have multiple types of nodes and links, are referred to as entity-relationship networks in the database literature.

Studying such rich data enables researchers to understand the structure of the network and the interactions between entities. However, network analyses are often limited to static snapshots of the network or a single type of link between entities. As seen in the examples above, real world networks often contain multiple types of nodes and links. These networks also evolve due to the temporal behavior of the entities and their relationships. For example, an author's affiliation may change, or she may collaborate with new people. A paper may accrue new citations in the future. Such changes will change the network structure. Thus, limiting the analysis to a static snapshot of single-mode networks results in the loss of a wealth of information that could be used to develop a better understanding of the underlying application domain.

Finding authoritative entities in networks is a common task that has been used in a wide variety of applications. For example, in a Web graph, a hyperlink from one page to another usually implies an "endorsement" or "recommendation." This is a fundamental assumption in authority flow-based ranking algorithm. PageRank [57] and HITS [28] are two seminal authority flow-based ranking algorithms.

PageRank iteratively computes the score of a page based on the scores of its neighbors. HITS separates the role of each web page into a *hub* or *authority.* The hub score estimates the value of its links to other pages, and the authority score estimates the importance of the page. PageRank and HITS have the following two limitations: First, they were developed for homogeneous or single-mode networks. Second, they cannot personalize authority flow. Balmin et al. [4] introduced ObjectRank to overcome these limitations. ObjectRank computes authority scores for entities in heterogeneous networks, which contain multiple types of entities and relationships. It also can personalize authority flow using a *weight assignment vector*, which associates a personalized authority flow weight with each edge type. The weight of each edge determines the importance of the corresponding relationship. ObjectRank is discussed in more detail in Section 4.2.1.

In this dissertation, I address two domain-specific ranking problems namely, ranking predictions in evolving networks and personalized ranking in social media. Both problems require the explicit specification of parameters/weights for ObjectRank. These two solutions to the two domain specific problems motivates the third and more general challenge of learning to rank in heterogeneous networks.

1.1 Challenges of Ranking in Evolving Networks

Answering queries in a bibliographic dataset poses some special challenges. Users want results (papers) that score high based on relevancy, impact, and timeliness. Information retrieval techniques are often used to retrieve the publications that are relevant to user queries. There is also a long history of research in bibliometrics, which measures the impact of a publication. A popular approach to measure the impact of a scientific article is counting the number of citations, which measures the popularity of the articles. Alternatively, the PageRank score of an article in a citation network can measure its authority.

Timeliness is a special challenge. Given two papers of equal impact and relevance, a user may prefer the more recent paper. This is because the more recent paper may provide insights that extend results from the dated paper and may even provide a (citation) link to the more dated paper. Since citations occur in strict chronological order, the dated paper would not be able to link to the more recent paper. In addition, the number of citations to a paper is a function of time. Since the recent paper has had less time to accrue the same number of citations as the more dated paper, the more recent paper may be a better choice. I use the term "impact" to refer to authority based on past citations and "usefulness" to refer to authority based on future citations.

Suppose we consider the future PageRank score computed *based only on the citations that will be accrued in the future* to determine usefulness. While this definition of usefulness makes sense, obviously, it is also problematic, since it is based on information that is not available at query time. It creates the need to make predictions about future citations.

CiteRank [68] is a key approach to predict the number of future citations of scientific publications. It models the citation process, in which researchers start their search from a recent paper and follow a chain of citations until satisfied. The probability of following a citation link is proportional to the publication time of the cited article; i.e. more recent articles are favored. CiteRank uses this random-walk model to predict the number of future citations. It then uses the predicted number of citations to rank the articles.

Chapter 3 presents FutureRank [60], an algorithm for predicting the number of future citations and the future PageRank score. FutureRank is an authority flow-based algorithm. It adopts mutual reinforcement between authors and papers, and utilizes a personalization vector to favor the more recently published papers. By using a heterogeneous network it also can benefit from adding more types of nodes and links into the network, such as the author affiliations and venues. FutureRank performs parameter estimation through an exhaustive search on the parameter space. The need for parameter-estimation algorithm will become more critical, as we add more types of nodes and links. This is one of the motivations that lead to my research in learning to rank in heterogeneous networks.

1.2 Challenges of Personalized Ranking in Social Media

Social media and the social interactions of users on the Internet have become a vital source of breaking news, while reflecting the expertise of crowds. The importance of such data has been acknowledged by Web search engines, which now index blog sites. Such sources are of particular importance in evolving situations such as disasters or other unplanned events. Social media is particularly important when the most knowledgeable experts may not be known a priori, a diversity of information is needed, information evolves over time, or the quality of information varies. These factors increase the value of social media and social interactions as valuable sources of information. On the other hand, crowd-sourcing can also create a massive stream of irrelevant and low-quality information. For instance, users of social networking sites such as LinkedIn or Twitte may receive notifications linked to thousands of blogs, messages, forums, etc., from other users and groups.

User behavior in social media is different from search behavior on the Web. As discussed in [50], users submit ad hoc keyword queries to Web search engines. In contrast, social media users may follow posts about a particular topic, or they may follow their favorite author or an interesting event. Hence, the topics, authors, and events that a user has liked in the past provide valuable information.

Personalization can enhance relevance and impact in social media recommendations. The challenges of personalized social media ranking are as follows:

1. The lack of a rich entity-relationship graph for a social media dataset that exploits the relationships between authors, blog posts, categories (topics) of the posts, and events (collections of posts).

2. The need to reflect user browsing needs and to personalize the ranking output based on their past history of retrieval.

As mentioned earlier, a major approach to personalization is based on authority flow-based ranking [4, 23, 57]. There are two variants for personalization as follows:

1. The first method uses a base set of entities of interest to the user. For example, [57] suggests using a user's favorite pages as the base set in PageRank, whereas [23] proposes computing a set of PageRank topic vectors, with one for each representative topic.

2. The second personalization method is exemplified by ObjectRank [4]. It adjusts the importance of various types of semantic connections. ObjectRank models nodes as entity types and groups edges by their edge type or semantic

type. It can personalize the ranking in heterogeneous networks by a *weight assignment vector*, which associates a personalized authority flow weight with each edge type. The weights determine the importance of each edge type in the ranking.

Chapter 4 presents a personalized ranking algorithm based on the second approach, ObjectRank. This algorithm is then used to effectively rank blog postings in a personalized way, by analyzing their content to discover key topics and exploiting their explicit categorization and author information. A limitation of this solution is that the parameters are set manually, instead of being learned from user feedback. This, too, motivates the next challenge.

1.3 Challenges of Learning to Rank in Heterogeneous Networks

Learning-to-Rank is a type of supervised machine learning problem in which the goal is to automatically construct a ranking model from given training data. For example, in information retrieval, query-document pairs are usually represented by numerical vectors called feature vectors, and a loss function associates a “cost” or “error” between a given ranking and the ideal ranking. The ranking model is then constructed by minimizing the error associated with the output of the model and the given training data.

Based on the type of the training data, learning-to-rank algorithms fall into three categories:

- **Pointwise**: The input(s) for this learner is one (or more) objects and their scores. This approach reduces ranking to regression or classification on individual documents [39, 52].

- **Pairwise**: The input(s) for this learner is one (or more) pairs of objects and the partial order for each pair. This approach no longer assumes absolute relevance. It reduces ranking to classification on pairs of documents. A classifier will classify each pair of objects to 1 if the first object is ranked lower and -1 if it is ranked higher [8, 14, 17, 18, 24, 66, 73, 74].

- **Listwise**: The input for this learner is a ranked list of objects. This approach directly operates on ranked lists and directly optimizes information retrieval evaluation measures or any other listwise loss functions [10, 70].

Existing learning-to-rank solutions focus on content-based features for ranking, and are not applicable to networks or authority flow-based models. Learning to rank objects based on both content information and relationships has not been extensively explored. As a result, many network-ranking algorithms, including FutureRank and the personalized ranking algorithm, discussed in Chapter 4, lack a framework for learning the parameters.

Chapter 5 presents an effective framework for estimating the parameters of authority flow-based ranking in heterogeneous networks. I first introduce a pointwise solution for learning from absolute scores. Asking users to provide absolute score may not always be easy, or sometimes, even possible. However, comparing two objects and expressing the preference is much easier. I present a pairwise solution for learning from partial preferences.

1.4 Contributions

- **Ranking in Evolving Networks:** I introduce a new measure I refer to as the future PageRank score. The future PageRank score is the PageRank score computed based only on the citations that will be accrued in the future. I then present FutureRank, a prediction algorithm for predicting the future PageRank score from the historical network structure. In addition to making use of the citation network, FutureRank uses the authorship network and the publication time of the article to predict future citations. Finally, I compare FutureRank with existing approaches and show that FutureRank is accurate and useful for finding and ranking publications.

- **Personalized Ranking:** I extend a social media dataset to exploit the associations between authors, blog posts, categories (topics) of the posts and events (collections of posts). Next, I apply personalized random-walk ranking algorithms. The personalization approaches are then evaluated through an experiment with real users, as well as an extensive study of a range of synthetic users whose preferences are defined based on intuitive criteria. The results shows that the accuracy of my personalized recommendations ranges from good to very good for a majority of users, and outperforms reasonable baseline approaches.

- **Learning to Rank in Heterogeneous Networks:** I present a framework to build an effective solution for estimating the parameters of an authority flow-based model. It includes pointwise and pairwise learning algorithms:
  - **Pointwise**: the training data include the absolute score of the training objects.
  - **Pairwise**: the training data include the preference for pairs of training objects.

  Finally, I evaluate the performance of these learning algorithms for learning the parameters of FutureRank and Personalized Ranking on multiple datasets. The results demonstrate that my framework successfully learns the parameters of FutureRank as well as the personalized parameters for personalized ranking. In addition, the results show that both pointwise and pairwise models can successfully learn from as few as 10 training samples. Both models are shown to be very robust against artificially-introduced noise in the dataset. The other advantage of my framework is that it allows learning-to-rank algorithms from a non-relational domain to be utilized for learning-to-rank in heterogeneous networks. Furthermore, it is simple to implement and converges after a few iterations with high accuracy.

1.5 Outline

In the next chapter, the related work is reviewed. Chapter 3 presents the FutureRank algorithm for ranking scientific articles, and Chapter 4 presents a personalized ranking algorithm for blog posts. Chapter 5 presents a framework for learning the parameters of network-ranking models that can learn from different types of user feedback. Finally, I conclude in Chapter 6 and discuss future work.

Chapter 2

Related Work

This chapter reviews related research in the area of ranking objects in networks. I first give an overview of the preliminary work in ranking objects in networks. I then review personalized ranking in social media. Next, I present an overview of related research in the area of learning to rank. Finally, related studies of one application of ranking in heterogeneous networks–ranking scientific articles–will be discussed.

2.1 Authority Flow-based Ranking

Finding authoritative entities in networks is a common task that has been used in a wide variety of applications. For example, in a Web graph, a hyperlink from one page to another usually implies an "endorsement" or "recommendation." This is a fundamental assumption in authority flow-based ranking algorithms. In 1998, two algorithms, PageRank [57] and HITS [28], launched the field of rankings based on the link structure. Both algorithms propose to rank web pages based on the link structure of the Web graph. PageRank was introduced in [57] to capture the intuition that important pages have a large number of important pages pointing to them. Let A denote the transition matrix which represents the authority transfer rate from the source page of a hyperlink to the target page:

$$A[i,j] = \begin{cases} \frac{1}{outdegree(j)} & \text{if there is an edge from } j \text{ to } i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Let P be the PageRank vector to be computed over the web pages. The *person-*

alization vector P^0 (also called *teleportation vector*) can be used to bias PageRank to prefer certain pages. In standard PageRank, P^0 is a uniform distribution to indicate the equal probability of randomly jumping to any page. The PageRank vector P is recursively defined as follows:

$$P = \alpha A \cdot P + (1 - \alpha)P^0 \tag{2.2}$$

The other link-analysis-ranking algorithm, HITS [28], considers that each web page has two roles: hub and authority. In the HITS algorithm, the first step is to retrieve the set of results for the search query. The computation is then performed only on this result set, not across all Web pages. The hub score estimates the value of its links to other pages, and the authority score estimates the importance of the page. $x^{<p>}$, the non-negative authority score assigned for page p, and $y^{<p>}$, the non-negative hub score assigned for page p, are then calculated as follows:

$$\begin{aligned} x^{<p>} &= \sum_{q:(q,p)\in E} y^{<q>} \\ y^{<p>} &= \sum_{q:(p,q)\in E} x^{<q>} \end{aligned}$$

In contrast to PageRank, which measures the global importance of a Web page independent of the query, HITS ranks pages according to the query topic, which can provide more relevant authority and hub pages. However, it is easily spammed. Inefficiency at query time is the other disadvantage of HITS. Collecting the relevant pages for each query, expanding it, and performing eigenvector computation are all expensive operations.

Balmin et al. [4] introduced ObjectRank for ranking entities in entity-relation graphs. ObjectRank models the entity sets and semantic connections among them as a schema graph, where the authority-transfer assignment is dependent on the type of connections. In ObjectRank, the transition matrix A depends on the authority transfer specified for each edge type. For a given graph $G = (V, E)$, edge (j, i) has type $t(j, i)$ belonging to a set of types numbered $1, ..., T$. Each type t has an associated importance represented by β_t. Thus, edge (j, i) has weight $\beta_{t(j,i)}$. The transition matrix A is defined as follows:

$$A[i, j] = \beta_{t(j,i)} a(j, i) \tag{2.3}$$

$$a_{j,i} = \begin{cases} \frac{1}{OutDeg(j,t(j,i))} & \text{if exists } (j, i) \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

where $OutDeg(j, t(j, i))$ is the number of outgoing edges from object j of type $t(j, i)$. Let P^0 represent the *personalization* vector. The score vector P is recursively defined as follows:

$$P = \alpha A.P + (1 - \alpha)P^0 \tag{2.5}$$

While authority flow-based ranking algorithm ObjectRank has been very successful for a wide range of problems, studies such as [35, 48, 49] show that for some other problems other methods perform better. Minkov et al. [49] used n-grams of edge labels as features for re-ranking the results of the authority flow-based model. Minkov et al. [48] proposed a method that upweights authority flow paths that are more likely to reach relevant entities in the training data. Lao et al. [35] showed that for a range of problems called *typed proximity queries* in networks, a weighted combination of path-constrained authority flows perform better than the generic type of authority flow models. Instead of assigning a weight per edge type, they assign a weight per edge type sequence: proximity is defined by a weighted combination of simple path experts, each corresponding to following a particular sequence of edge types.

2.2 Personalized Ranking

Most of the existing search engines focus only on answering user queries, although personalization will be more and more important as the amount of information available on the Web increases. Some commercial search engines provide personalization by accommodating the topics of interest, prior search history, or other descriptions of users' preferences.

Personalized Web Search: There is a large corpus of work on personalizing the results of Web searches. Solutions include using the user's closest Web community, relevance feedback, and navigation history [64, 65]. Recently, research has been conducted on leveraging the tagging of pages in social networks to rank the pages [25, 5, 26]. These methods generally create a tripartite graph of users, tags, and pages, and apply adaptations of the PageRank algorithm. However, these approaches cannot be directly applied to domains such as blogs and social media, for which explicit links (hyperlinks or tag-to-user-to-page) are not available.

Kritikopoulos et al. [32] consider the Web community of a specific user to personalize Web search results. Past interactions of the user with the search engine are used to improve future search results. For each Web community, its neighborhoods, including the documents linked to or from and documents in the community, are determined. Query answers are ordered to reflect the number of times these community neighborhoods have been visited. [64, 65] proposed refining search results based on users' history, navigational history, browsing history, or query history. Multiple data mining techniques can be applied to extract usage patterns from Web logs [13, 63].

Personalization of Authority Flow-based Ranking: A major approach to personalization is based on *authority flow-based ranking* [23, 57, 4], which uses two methods for personalization. The first uses a base set of entities of known interest to the user; e.g., PageRank [57] and ObjectRank [4] suggest using the user's favorite pages as the base set in the personalization vector, whereas [23] proposes computing a set of PageRank topic vectors, one for each representative topic.

The second personalization method adjusts the importance of various types of semantic connections. As mentioned earlier, ObjectRank [4] groups edges by type and assigns an authority-transfer rate to each type of edge. ObjectRank allows the authority-transfer rates to be personalized for each user. The *weight assignment vector* determines the importance of each type of relationship in the ranking. For example, in Figure 2.1, 4 edge types associate *BlogPost* to *BlogPost*, etc. Eight weights are associated to these edges, one for each direction of an edge type. Varadarajan et al. [67] present techniques to learn the weights using relevance feedback.

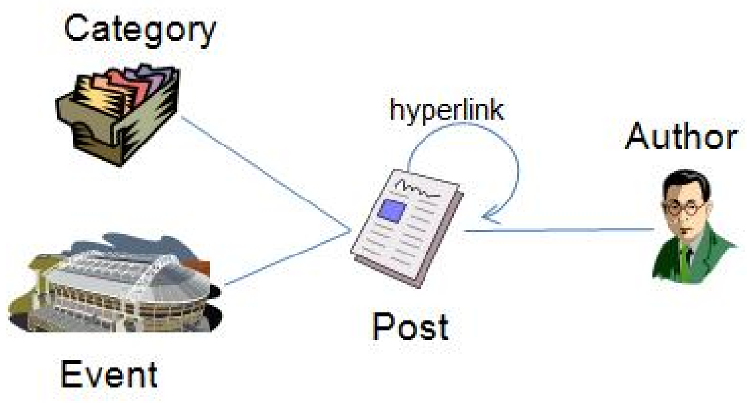


Figure 2.1: Enhanced social media schema graph

2.3 Learning to Rank

The well-known network-ranking algorithms PageRank [57], HITS [28], and many extensions, including ObjectRank [4], Co-Rank [75], and FutureRank [60], are each defined for a specific domain or network. Applying to new domains requires extensive tuning and adaptation. In addition, there is no universal framework for learning the parameters of such algorithms in a new domain. For example, ObjectRank is generic enough to be used for any networks with many types of nodes and relationships. However, the authority-flow rates, which have been shown to dramatically affect the quality of the results, have to be set manually by a domain expert or given by the user. Due to the complexity of network-ranking algorithms, in which the rank of an object is correlated to the rank of its neighbors, traditional learning algorithms are not applicable. As a result, many solutions for ranking in networks, including FutureRank and the personalized ranking algorithm that will be presented in this dissertation, lack a framework for learning the parameters.

Recently, a new generation of so-called *Learning to Rank* or *Structured Prediction* models, has been introduced. Unlike traditional ranking models, these models learn the ranking function from training data using one of the following approaches:

- **Pointwise**: The input for this learner is a single object, and the output is the score of the object or the class label. This approach reduces ranking to regression or classification on single documents [39, 52].

- **Pairwise**: The input for this learner is a pair of objects, and the output is the partial order preference. This approach no longer assumes absolute relevance. It reduces ranking to classification on document pairs. A classifier will classify each pair of objects to 1 if the first object is ranked lower and -1 if it is ranked higher [8, 14, 17, 18, 24, 66, 73, 74].

- **Listwise**: The input for this learner is the collection of objects, and the output is the ranked object list. This approach directly operates on ranked lists and directly optimizes IR evaluation measures or any other listwise loss functions [10, 70].

These approaches are introduced for non-relational data and cannot be directly used for networks. In this dissertation I present a framework for learning to rank that can utilize these algorithms for ranking in heterogeneous networks.

2.3.1 Learning to Rank in Networks

Lao et al. [36] proposed a method for learning a weighted combination of path-constrained random walks, that is able to discover and leverage complex path features of relational retrieval data. They also create a node for each word in the documents and extend the network with these nodes. Finally, they use a combination of path-constrained random walkers in the extended network. Their solution is more a retrieval algorithm for typed proximity queries in the network in which they learn linear weighting schemes over all relation paths of bounded length l.

For a given transition matrix A, in practice, the PageRank or ObjectRank score vector is frequently computed via power iteration, by initializing $p = p^0$ and iterating $P = A.P$ until convergence. Chakrabarti et al. [11] truncate the recursion and suggest $P = A^H.P^0$ for modest values of H (10-50). Therefore, the problem of learning the parameters reduces to optimizing some loss function for $P = A^H.P^0$. They solve this problem for a specific loss function. However, for any different choice of loss function, they need to solve a new optimization problem. In addition, Chakrabarti et al. [11] use a very specific authority transfer that normalizes transferred authority scores. The authority transfer from an node i to node j is specified as follow:

$$C(i,j) = \frac{\beta(t(i,j))}{\sum_{j'} \beta(t(i,j'))}$$

However, this formula means that a node does not fairly distribute its authority according to the neighbor types–e.g., if a paper has 10 outgoing citations and 1 author, and assuming all types have equal weight, the author will get 1/11 of authority in this case. This is different from ObjectRank, which gives 1/2 of authority to the author node, and the 10 papers would share the other 1/2. ObjectRank's approach is more intuitive. The formulation of transition matrix by Chakrabarti et al. allows them to scale all β_ts by a factor and $C(i,j)$ remains unchanged, which they use later in the optimization process.

2.4 Ranking Scientific Articles

Ranking scientific articles is an important and challenging problem with a long tradition of study. One of the important early steps in this area was the work of Garfield [19] in 1970s. He proposed a measure for ranking journals and called it the *ImpactFactor*, which is calculated based on a three-year period. For example, the impact factor of year i for a journal j is calculated as follows:

$A =$ the number of times articles published in journal j in years $i-1$ and $i-2$ were cited in indexed journals during year i.

$B =$ the number of articles, reviews, proceedings or notes published in journal j years $i-1$ and $i-2$

$ImpactFactor(j,i) = A/B$.

Thus, the impact factor is an approximation of the average number of citations within a year given to the set of articles in a journal published during the two preceding years. He also applied a similar idea of counting citations to evaluate scientists [20].

Based on this work, many different versions of impact factors have been proposed [21, 29, 30, 31, 37, 38, 53, 54, 58]. However, all of these approaches count citations. The problem with counting citations is that it measures only the popularity of articles and not the authority, which is measured by scores similar to PageRank.

After the introduction of a revolutionary ranking model called PageRank [57] and its application in different domains [6, 15, 22, 27, 47, 51, 55], many researchers explored using PageRank on citation networks for ranking scientific articles [7, 12, 45]. The results confirmed that *ImpactFactor* finds the popularity while the PageRank score reflects the authority.

In addition to analysis of the citation network, researchers have considered making use of the co-authorship network as well. For example, Liu et al. [42] applied PageRank to the co-authorship network to rank scientists. Zhou et al. [75] used the idea of mutual reinforcement between hubs and authorities [28]. They made use of three networks: the citation network, the co-authorship network, and the authors' social network. In the authors' social network, two authors are linked if they published a paper together or attended the same conference. Co-Rank contains two independent random walks in the citation network and authors' social network, as well as a random walk on the authorship network. If at any given moment the random walk is on the author side, then it can either make m intra-class steps or k inter-class steps. Similarly, if it is on the document side, then it can either make n intra-class steps or k inter-class steps:

$$M = \begin{bmatrix} (1-\lambda)(\widetilde{A}^T)^m & \lambda DA^T(AD^TDA^T)^k \\ \lambda AD^T(DA^TAD^T)^k & (1-\lambda)(\widetilde{D}^T)^n \end{bmatrix}$$

in which A denotes the adjacency matrix of authors' social network, D denotes the adjacency matrix of document citations, and AD denotes the adjacency matrix of authorship network from authors to documents (DA just shows the reverse direction of AD). Next, they concatenated the two vectors of ranks (vector a for articles and vector d for documents) into a vector v such that $v = [a^T, d^T]^T$. Finally, by solving $v = Mv^T$, the final scores are computed. The model provides a co-ranking of articles and authors, and is evaluated based on the author ranking. Nie et al. also consider articles as Web objects and collect Web information for the object [56]. They rank Web objects in terms of their popularity and relevance to the user query. ObjectRank [4] computes an authority value for entities in the entity-relationship graph for the DBLP database. The graph contains articles, authors, and venues.

However, the focus of ObjectRank is on personalized ranking. It personalizes the ranking by assigning a weight to each link, which determines the importance of the corresponding association in the ranking.

A problem with the above approaches is that they rank articles by their prior popularity or authority. Consequently, recently published articles will always receive lower scores in such models. Walker et al. [68] introduced CiteRank, which uses the publication time of the articles in a random walk to predict the number of future citations. They then use the predicted values to rank the articles. CiteRank models the citation process by which researchers start their search from a recent paper or reviews and follow a chain of citations until satisfied. The probability of jumping to an article is proportional to its publication time, which is computed as follows:

$$\rho_i = e^{-age_i/T_{dir}}$$

in which age_i is the age of the i-th article. The CiteRank traffic of the paper is then defined as follows:

$$\overrightarrow{T} = I.\overrightarrow{\rho} + (1-\alpha)W.\overrightarrow{\rho} + (1-\alpha)^2W^2.\overrightarrow{\rho} + ...$$

which shows the probability of encountering an article via all possible paths (W is the adjacency matrix of the citation network). Their experiment shows that the best correlation between the predicted citation by CiteRank and the number of future citations is around 0.68.

2.5 Noise and Missing Data

Like other machine learning models, ranking models can suffer from noisy data or missing data. Network datasets are often incomplete, which results in missing nodes and missing edges in the corresponding networks. There are many ways to handle missing data in the non-network domain [33, 69]. The most common means of dealing with missing data is deletion, i.e., when all instances with a missing feature are deleted. However, a better solution is imputation, where missing data are replaced with substitute values. A few of the well-known imputation approaches are as follows:

- **Mean imputation:** In this approach any missing feature value is replaced with the mean of that feature over all other instances. This has the benefit of not changing the sample mean for that feature.
- **Regression imputation:** A regression model is estimated to predict observed values of a feature based on other features, and that model is then used to impute values in cases where that feature value is missing [33].
- **Multiple imputation:** Missing values for any feature are predicted using a stochastic regression model (based on the existing values from other features), where the predicted values are drawn from the posterior distribution. The predicted values, called imputes, are substituted for the missing feature values, resulting in a full data set called an imputed data set. However, this process is performed multiple times producing multiple imputed data sets. Each imputed data set is analyzed separately and the results are averaged [69].

In networks, missing data results in missing edges. The best solution for such scenario is to use link prediction algorithms to predict the missing edges [3]. I will discuss this issue in more detail in Chapter 5. I will explain how missing data in networks can affect the results of an authority flow-based model and suggest some possible solutions.

Chapter 3

FutureRank: Ranking Scientific Articles by Predicting their Future PageRank

A simple and popular approach to measure the quality of a scientific article is by counting the number of citations. Unfortunately, newly published articles do not have many citations. It is important to differentiate the popularity in the past from *usefulness*, which is defined as the number of citations in the future. In other words, usefulness is based on future popularity and authority. Future popularity is well defined (but unobserved). I define future authority by introducing a measure I refer to as the future PageRank score. The future PageRank score is the PageRank score computed *based only on the citations that will be accrued in the future.*

While this definition of usefulness obviously makes sense, it is also problematic in that it is based on information that is not available at the query time. Hence, information about future citations has to be predicted.

In summary, my work makes the following contributions:

- I introduce a new measure I refer to as the future PageRank score. The future PageRank score is the PageRank score based only on citations that will be accrued in the future.

- I present FutureRank, a prediction algorithm for predicting the future PageRank score. In addition to making use of the citation network, FutureRank uses the authorship network and the publication time of the article to predict future citations.

- My experiments compare FutureRank with existing approaches by using precision-recall values and Spearman's rank correlation.

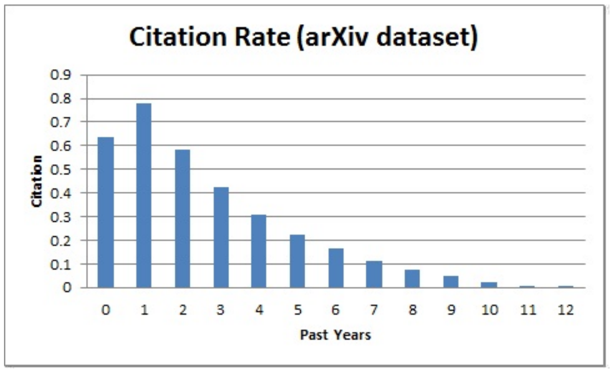


Figure 3.1: Average number of citations in the dataset (arXiv), which articles obtain based on the number of years prior from publication date

The rest of this chapter is organized as follows: Section 3.1 describes FuturneRank, a model for ranking scientific articles. Section 3.2 presents experimental results and evaluations. Finally, I conclude in section 3.3.

3.1 Proposed Model

In this section, I describe an algorithm for ranking scientific articles and authors. The goal is to rank papers based on predicted future citations, as this will help researchers find good articles more easily. To do this, we need a measure to evaluate the usefulness of the articles. As mentioned earlier, one traditional measure is the popularity or number of citations, but a better measure would be the PageRank score–i.e., the estimated authority of the article.

Figure 3.1 shows the average number of citations of an article in the following years for the arXiv dataset, which is a collection of high-energy physics publications [1]. The figure shows that the number of citations an article receives in each following year decreases exponentially; most citations are received after one year. Consequently, any algorithm that does not incorporate publication date into the ranking will not be able to capture this effect.

In addition, useful articles are often written by well-known researchers. Therefore, another source of valuable information is the authorship network, from which author reputation and contribution can be extracted.

The assumptions for ranking scientific articles based on the usefulness are:

1. Useful articles are cited by many important articles.
2. Useful articles are written by researchers with high reputations, and researchers have high reputation if they write good research articles. This illustrates the mutual reinforcement between articles and authors.
3. Recently published articles are more useful. In other words, they are more likely to accrue citations in the future.
4. Recently cited articles are more useful.

One might assume that these assumptions would penalize fundamental papers that are not recent, yet are still cited by new papers. However, even these types of papers will achieve good ranks using these assumptions. While these papers are not recent, they still will be cited by recently published papers. Hence, assumptions 1, 3, and 4 support such papers. In an authority-transfer model, the citing papers will propagate their score to the referenced papers, so the older papers will have high scores because of their recent citations.

3.1.1 Network Structure

Figure 3.2 shows an example of a bibliographic network. The network has two types of nodes, papers and authors, with two types of links: *authorship links*, which are undirected links between papers and their authors, and *citation links*, which are directed links from a paper to each paper that is cites. This network can be split into two subnetworks. The first is the citation network, which only contains paper nodes and citation links. One may use PageRank on this subnetwork to simulate

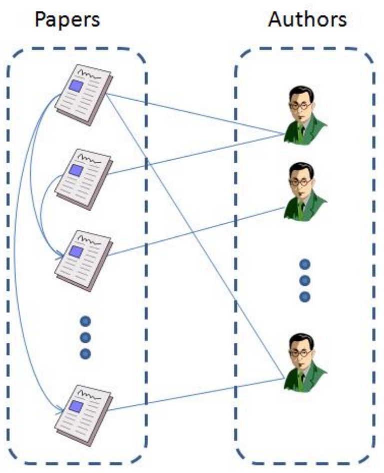


Figure 3.2: An example of the scientific article network, which has two types of nodes, papers (rectangles) and authors (circles), and two types of edges: authorship edges, which are between authors and papers, and citation links, which are between papers.

the authority transfer from articles to their references. The second subnetwork is the authorship network, which contains both paper and author nodes, but only contains authorship links. It is a bipartite network, in which articles are *authorities* and authors are *hubs*. This network can simulate the mutual reinforcement between articles and their authors using a HITS-style propagation algorithm. Figure 3.3 shows the mapping. An adjacency matrix is used to represent the network, and ranking scores are stored in vectors. If P is the set of papers and A is the set of authors, then matrix M^C is a $|P| \times |P|$ citation matrix:

$$M^C_{j,i} = \begin{cases} \frac{1}{|citations(p_i)|} & \text{if } p_i \text{ cites } p_j; \\ 0 & \text{otherwise;} \end{cases}$$

For any paper p_i that does not cite any article in the dataset, $M^C_{i,j}$ is set to 1, for all js. In other words, there will be virtual links from dangling nodes to every other node. Next, the authorship matrices M^{AP} and M^{PA} are defined as $|P| \times |A|$

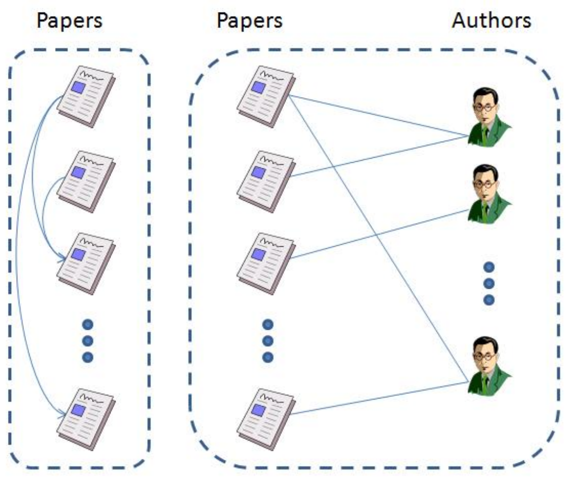


Figure 3.3: Network decomposition: a single mode network of citations and a bipartite network of authorship

and $|A| \times |P|$ matrices:

$$M^{AP}_{j,i} = \begin{cases} \frac{1}{|papers(a_i)|} & \text{if } a_i \text{ is the author of } p_j; \\ 0 & \text{otherwise;} \end{cases}$$

$$M^{PA}_{i,j} = \begin{cases} \frac{1}{|authors(p_i)|} & \text{if } a_j \text{ is the author of } p_i; \\ 0 & \text{otherwise;} \end{cases}$$

3.1.2 FutureRank

Since two subnetworks share nodes, the scores of the objects in each network cannot be computed separately. Instead, I propose a new ranking algorithm, which I refer to as FutureRank. It operates on both networks, passing information back and forth between networks. The ranking algorithm is an iterative algorithm that runs one step of PageRank and one step of HITS, then combines the results. It then repeats these steps until convergence. Let R^P denote the vector of paper scores, and

R^A denote the vector of author ranks. Author scores are then computed as follows:

$$R^A = M^{PA}.R^P$$

This computes the hub scores in the authorship network. Articles transfer their authority score to their authors, and authors collect the authority score of all of their publications. However, article scores are computed as follows:

$$\begin{aligned} R^P &= \alpha M^C.R^P \\ &+ \beta M^{AP}.R^A \\ &+ \gamma R^{Time}. \end{aligned}$$

This formula is a linear combination of three factors:

- $M^C * R^P$ is the authority scores coming from the citation network,
- $M^{AP^T} * R^A$ is the authority scores coming from the authorship network,
- and R^{Time} is a "personalized" PageRank vector. In the original PageRank model, the personalized vector is a pre-computed score vector to favor the user's preferences. The default value for all nodes is $\frac{1}{n}$, where n is the number of nodes in the network. In FutureRank, the values in the personalized vector are pre-computed based on the publication time of the papers. FutureRank favors the papers that has been published recently:

  $$R_i^{Time} = e^{-\rho*(T_{current}-T_i)}$$

  where $T_{current}$ is the current time or the query time, and T_i is the publication time of p_i. $T_{current} - T_i$ shows the age of p_i.

The initial value of R_i^P is $\frac{1}{|P|}$ and similarly the initial value of R_i^A is $\frac{1}{|A|}$. This initialization keeps the sum of paper ranks equal to 1, as well as the sum of author ranks. This property will also hold after each iteration, since the computation performs an authority propagation and the sum of the weights, $\alpha + \beta + \gamma + (1 - \alpha - \beta - \gamma)$, is equal to one.

3.2 Evaluation

This section demonstrates the performance of FutureRank (several variants) on a collection of scientific papers. Several performance criteria are used to compare FutureRank with state-of-the-art algorithms. I conclude with a discussion of running time and convergence.

3.2.1 Dataset

The evaluation is performed on two real datasets of scientific articles, the arXiv (hep-th) dataset [1] and the DBLP dataset. The arXiv dataset contains articles published on high-energy physics from 1993 to 2003. It contains approximately 28,000 articles, 15,000 authors, and 350,000 citations. The DBLP dataset contains computer science articles published from 1993 to 2003. It contains approximately 21,000 articles, 16,000 authors, and 100,000 citations. I extracted authors from the description file for each article. Two authors are considered identical only if their full names match. This is a strong matching criterion and misses many partial matches. More sophisticated author name resolution strategies have the potential to improve performance. Citations to articles outside the dataset were removed.

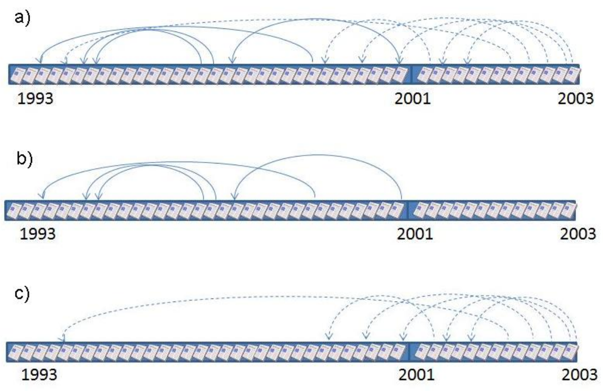


Figure 3.4: Hypothetical figure of entire dataset, in which horizontal lines show the timeline and links show the citations: (a) the full dataset; the horizontal line shows the timeline and links show the citations of articles by later articles; (b) the query data, which only contain links originating before 2001; and (c) the evaluation data, which only contain links originating after 2001.

3.2.2 Evaluation Setup

The dataset is split into two sets: The first set, historical data, contains all papers published before 2001, and the second set, future data, contains papers published in or after 2001. The first set is then used as training data for predictions, and the second set is used as the gold standard for the evaluations.

To construct the evaluation, one may view the system from the standpoint of a user in 2001 who is searching for research papers. At this point, the only available information is the information in the first set. As the definition of usefulness, the most useful paper for that user is the paper that will obtain the highest *future PageRank*, e.g., the highest number of citations after 2001. The *future PageRank* is computed as the PageRank scores of articles based only on citations in and after 2001. To do this, a new network of all papers from 1993 to 2003 is created. However, the edges are restricted to citations that originate from articles published in and after 2001.

To illustrate the setup, figure 3.4(a) shows a hypothetical figure of a full dataset. The horizontal line in the figure shows the timeline, and links show the citations. Dashed links show future citations, which are available only in the evaluation data. The historical and evaluation sets are also shown separately in Figure 3.4(b) and 3.4(c), respectively.

Gold-standard scores are computed as the PageRank score on the evaluation network. It is called *future PageRank* because it is a PageRank score that is computed on the future network. The goal of FutureRank is to predict future PageRank by using historical data.

Recall the personalized vector R^{Time} that was introduced in section 3.1.2. To find the best value of ρ in R^{Time} for the arXiv dataset, I found the best exponential curve that fits Figure 3.1. Ignoring the data for *#years*$= 0$, the best curve is:

$$c * e^{-0.62*x}$$

This value is exactly the same as that reported by [68] (for the arXiv dataset), in which $T_{dir} = 1.6 = \frac{1}{0.62}$ is the best value. The authors of [68] ran CiteRank for all possible values of T_{dir} and found $T_{dir} = 1.6$ years as the best value. While both obtained the same value, I found the best value by curve fitting and *without* using the evaluation data. In CiteRank, the authors ran the model for all possible values and needed test data to find the values that provided the best precision. I performed the same curve fitting for the DBLP dataset and obtained $\rho = 0.20$ as the best value for the DBLP dataset.

3.2.3 Ranking: Evaluation and Approaches

I use two approaches for evaluating the FutureRank algorithm:

- Precision and recall

- Spearman's rank correlation between the ranking output of FutureRank and future PageRank.

The following variations of FutureRank are considered in this evaluation:

- **FutureRank**: The FutureRank model uses all available information (authorship, citation networks, and publication time):

$$\begin{aligned} R^P &= \alpha * M^C * R^C \\ &+ \beta * M^{A^T} * R^A \\ &+ \gamma * R^{Time} \\ &+ (1 - \alpha + \beta + \gamma) * [1/n] \end{aligned}$$

$$R^A = M^A * R^P$$

- **FutureRank(CT)**: a variant of FutureRank that only uses citation network and publication time, but does not use the authorship network ($\beta = 0$).

$$\begin{aligned} R^P &= \alpha * M^C * R^C \\ &+ \gamma * R^{Time} \\ &+ (1 - \alpha - \gamma) * [1/n] \end{aligned}$$

$$R^A = M^A * R^P$$

  The information used in this model is similar to the information CiteRank [68] uses. Neither uses the authorship network.

- **FutureRank(CA)**: a variant of FutureRank that only uses citation and authorship networks ($\gamma = 0$).

$$\begin{aligned} R^P &= \alpha * M^C * R^C \\ &+ \beta * M^{A^T} * R^A \\ &+ (1 - \alpha - \beta) * [1/n] \end{aligned}$$

$$R^A = M^A * R^P.$$

This model does not use the publication time of the articles, which is similar to CoRank [75].

- **PageRank**: This is the traditional PageRank model for $\alpha = 0.9$ and a random jump with probability of 0.1 ($\alpha = .9, \beta = 0$, and $\gamma = 0$).

$$\begin{aligned} R^P &= \alpha * M^C * R^C \\ &+ (1 - \alpha) * [1/n] \end{aligned}$$

$$R^A = M^A * R^P$$

3.2.4 Evaluation Metrics

There are several metrics which are commonly used to judge how well an ranking algorithm performs [46]:

- **Set-based metrics:** Precision, recall, and F measure are set-based measures that are often used to compare two ranked lists. They are computed using unordered sets of top k documents. Precision is the fraction of top k retrieved documents that are relevant, while recall is the fraction of top k relevant documents that are retrieved. In simple terms, high precision means that an algorithm returned substantially more relevant results than irrelevant results, while high recall means that an algorithm returned most of the relevant results. A single measure that trades off precision versus recall is the F-measure; it is the weighted harmonic mean of precision and recall. In recent years, other measures have become more common. A popular measure among the TREC community is Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels. Among evaluation measures, MAP has been shown to have especially good discrimination and stability.

- **Rank-based metrics:** Unlike the precision and recall, the Spearman's rank correlation coefficient ρ considers the order of the results as follows:

  $$\rho = 1 - \frac{6 \sum (r_i - g_i)^2}{n(n^2 - 1)}$$

  where r_i is the predicted rank of document i, and g_i is the rank of document i in the gold standard. It uses a quadratic loss function (on the ranking position) normalized by the size of the input lists. It measures the strength of the linear relationship between two ranked lists. The coefficient can range between -1 and 1. The sign of the Spearman's correlation indicates the direction of the association between the two given ranked lists.

  Kendall's τ is another measure for the evaluation of ranked retrieval results as follows:

  $$\tau = \frac{\#concordant\_pairs - \#discordant\_pairs}{\frac{n(n-1)}{2}}$$

  where $concordant\_pairs$ represents the pairs that are in the correct order and $discordant\_pairs$ represents the pairs that are in the wrong order. In most cases, Kendall's τ and Spearman's ρ values are very similar. However, they represent different interpretations. Spearman's ρ accounts for the variability across two ranked list, whereas Kendall's τ represents a probability, i.e., the difference between the probability that the observed data are in the same order versus the probability that the observed data are not in the same order.

The output of FutureRank is a ranked list that has to be compared with the gold standard ranking. I pick the top $k = 50$ papers returned by future PageRank as the (relevant) gold-standard set. The precision of each algorithm is then computed as follows:

$$Precision = \frac{|FutureRank_{top50} \cap Future\ PageRank_{top50}|}{50}$$

Since I picked the same k for the number of retrieved results and the number of relevant documents, precision and recall are always equal. Hence, I only the precision for varying ks as well as the precision-recall curve for top $k = 50$ gold standard results. In addition, I will report the Spearman's ρ for the entire output lists (not just the top k) to compare the output of FutureRank with the related work. The reason for choosing Spearman's ρ was that most other methods reported this value.

A nice property of both precision and Spearman's ρ is that they only consider the rank of documents (not the absolute scores that led to the ranking). Score changes that do not lead to reordering have no effect on the value of the precision or Spearman's correlation coefficient. The rationale for using ranking positions for evaluation is that the goal is the rank and not the score. In the context of FutureRank, most papers will obtain only a few citations in their life time; hence, we determine precision over the top k results.

3.2.5 Parameter Estimation

I first investigate the performance sensitivity of FutureRank to its parameters (citation [α], author [β] and publication time information [γ]). Figure 3.5 shows the precision of FutureRank for different values of α, β, and γ for the arXiv dataset (I

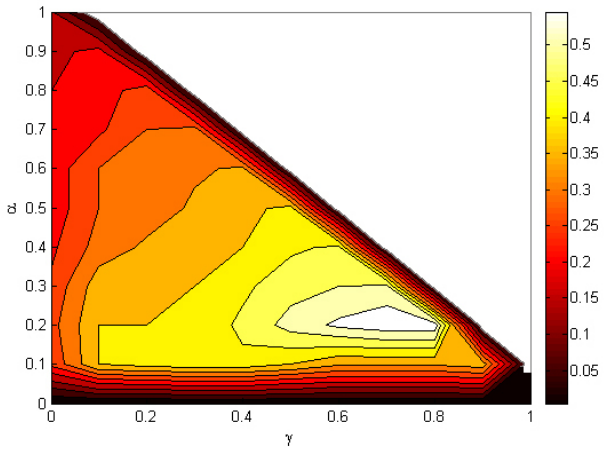


Figure 3.5: The precision of FutureRank for different configurations of the parameters α, β, and γ on the arXiv dataset. At any point in the figure, the value of β is equal to $1-\alpha-\gamma$. Similar analyses on the DBLP dataset resulted in a near-identical figure.

obtained a very similar graph for the DBLP dataset as well). The x-axis shows the value of γ and the vertical axis shows the value of α. Since $\alpha+\beta+\gamma$ is always equal to 1 at any point in the heatmap, the value of β is $1-\alpha-\gamma$ (the top right triangle of the map is empty, because the sum of α, β, and γ cannot be more than 1). The lighter the color in the heatmap, the higher the precision.

Figure 3.5 shows all possible configurations of FutureRank. For example, the accuracies shown on each edge of the heatmap triangle show the combination of using only two types of information. All values on the horizontal edge are obtained for $\gamma = 0$, which means that the horizontal edge shows all possible configurations of FutureRank(CA),while the hypotenuse shows all possible configurations of FutureRank(CT). Each corner also shows the precision of FutureRank, which only uses one type of information (one of α, β, and γ is equal to 1 and the other two are zero). The nice observation is that the space has a single optimal region, rather than a more complex collection of optimal configurations.

The highest precision of FutureRank is obtained at $\alpha = 0.19$, $\beta = 0.02$ and $\gamma = 0.79$. However, one should be careful when interpreting these values. This combination of values does not mean that the effect of time R^Time is three times more important than the citation. This happens because the scores of the different types of objects have different scales. While all scores are between 0 and 1, the sum of scores in each vector R^P, R^A, and R^{time} must sum to 1 individually. Hence, the range of scores in the different vectors can be very different. For example, the sum of author scores should be 1–as well as the sum of papers scores–but there are about twice as many papers as authors. This means that the average score for papers will be less than the average score for authors.

3.2.6 Further Comparison of Proposed Algorithms

This section shows the effectiveness of FutureRank in more detail. Figure 3.6 shows the precision-recall curves of four models:

1. **FutureRank**: The best curve is obtained by $\alpha = 0.19$, $\beta = 0.02$, and $\gamma = 0.79$.
2. **FutureRank(CT)**: The best curve is obtained by $\alpha = 0.2$,$\beta = 0$, and $\gamma = 0.8$.
3. **FutureRank(CA)**: The best curve is obtained by $\alpha = 0.2$, $\beta = 0.8$, and $\gamma = 0$.
4. **PageRank**

As shown in Figure 3.6, the top 25% of the FutureRank output are correct, which means that the precision is 100%. While this figure shows that FutureRank has a better performance on the arXiv dataset, note that for both arXiv dataset and DBLP dataset, FutureRank significantly outperforms the other baseline algorithms PageRank and FutureRank(CA). FutureRank(CA) uses the same features that CoRank [75] uses for ranking authors.

Figure 3.7 shows the precision of FutureRank for varying k.

Consider the following four categories:

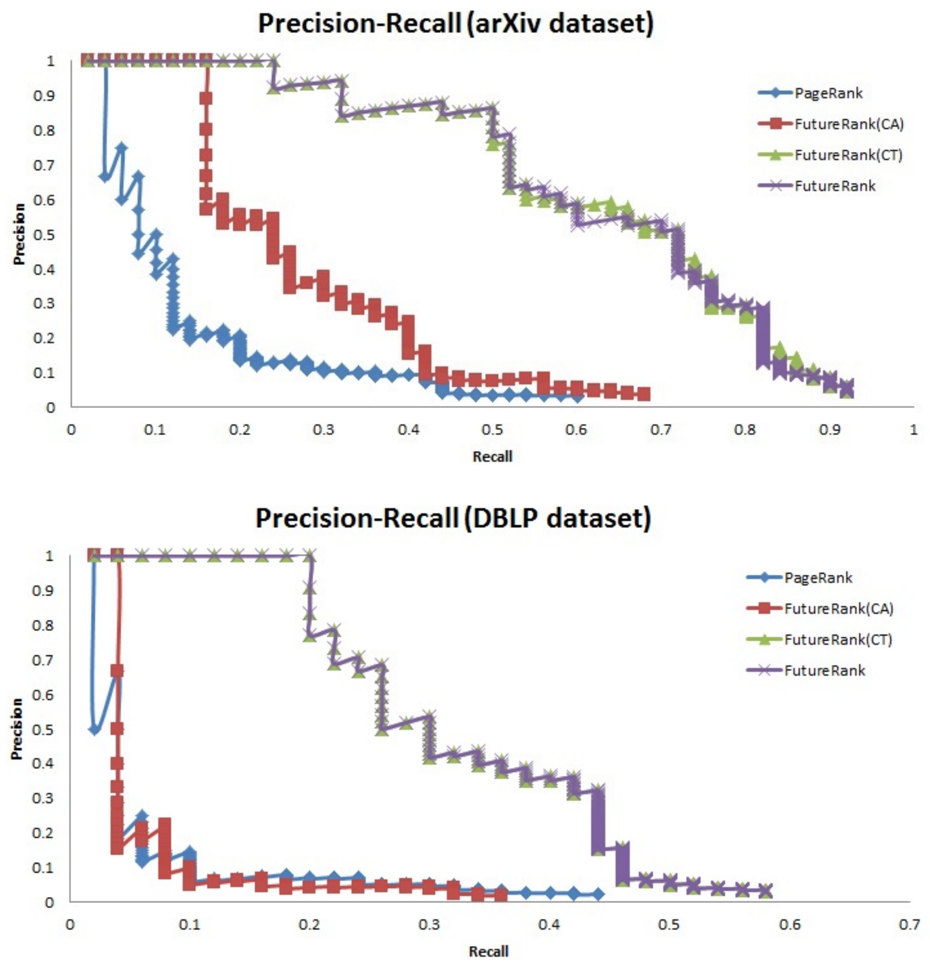
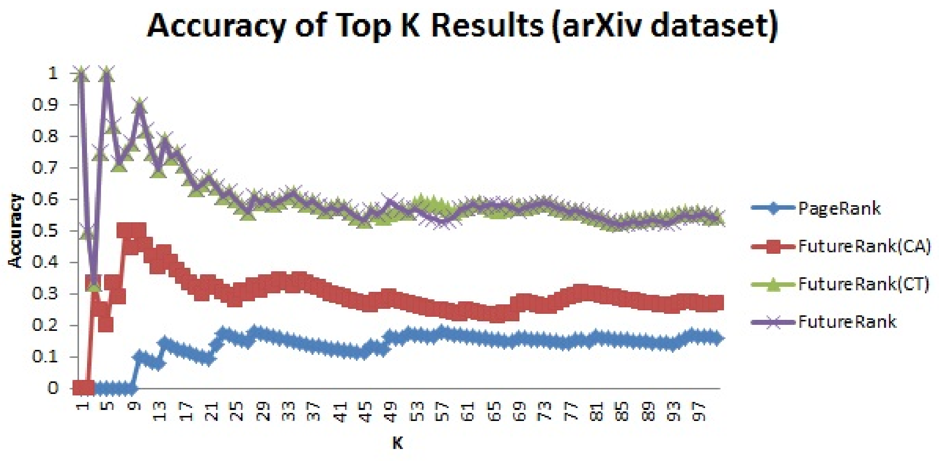


Figure 3.6: Precision-recall based on precision in the top 50 results.

- **High→High (True Positive)**: This category represents the papers that are among the top k results based on the number of citations they acquired in the past. They are also among the top k results based on the number of future citations. The papers in this category are true positives.

- **Low→High (True Positive)**: This category represents the papers that are not among the top k results based on the number of citations that they acquired in the past, but they are among the top k results based on the number of future citations. The papers in this category are true positives.

- **High→Low (False Positive)**: This category represents the papers that are



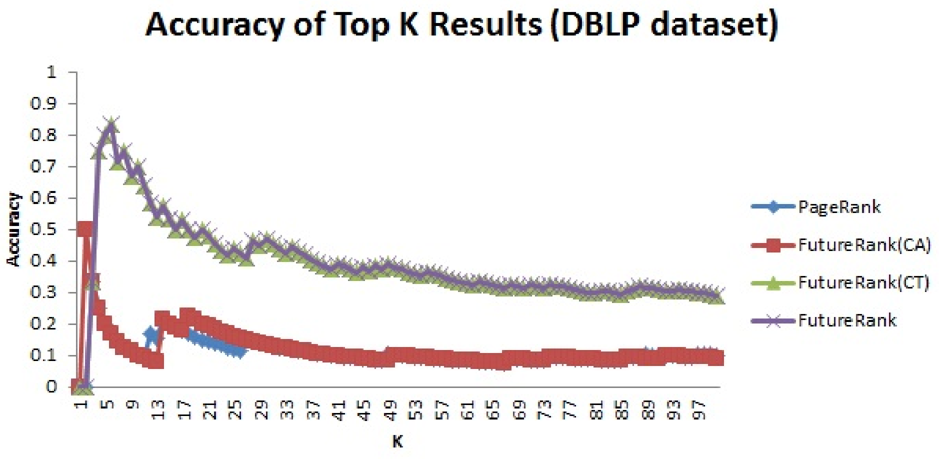


Figure 3.7: Precision of the top k results of FutureRank compared to the top k results of future PageRank

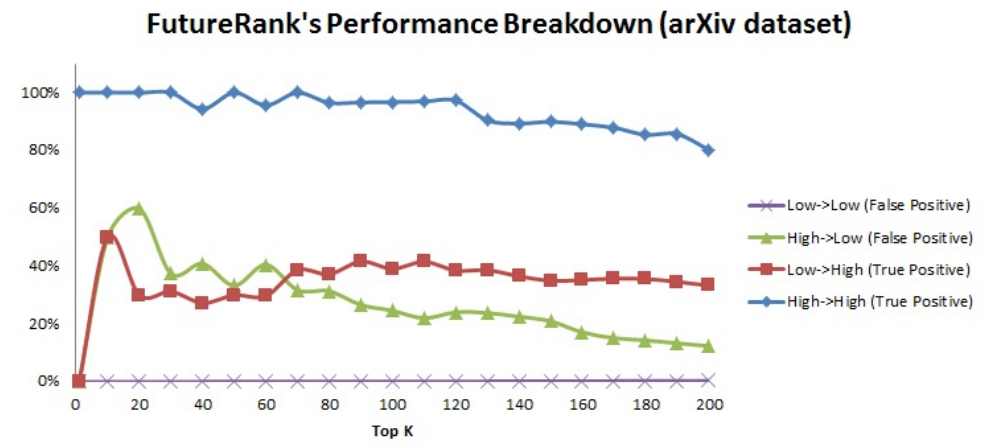
among the top k results based on the number of citations they acquired in the past, but they are not among the top k results based on the number of future citations. The papers in this category are false positives.

- **Low→Low (False Positive)**: This category represents the papers that are neither among the top k results based on the number of citations they acquired in the past, nor among the top k results based on the number of future citations. The papers in this category are false positives.

Figure 3.8 shows the percentage of the papers from each category that were among the top k output of FutureRank. This figure shows that FutureRank successfully predicted almost all of the papers from the High→High category and more than 40% of the papers from the Low→High category for the arXiv dataset. FutureRank also makes accurate prediction for the category Low→Low. Finally, the least accurate predictions are for the category High→Low. This is as expected as FutureRank has no knowledge to identify such papers. FutureRank performs better on the arXiv dataset compared to the DBLP dataset.

The authors of CiteRank [68] do not report precision/recall numbers, but they computed the correlation coefficient between the number of citations in the evaluation set (in the future) and the citation traffic estimation by CiteRank. They run their experiments on the arXive dataset. The best correlation between the predicted citation by CiteRank and the number of future citations for the arXiv dataset was around 0.68, while the correlation between the FutureRank score and the future PageRank score was 0.83. This is encouraging, but these correlation values are not comparable since they are correlations of two different measures.

Hence, instead of comparing the correlations between scores, I chose the Spearman's rank correlation. The CiteRank article showed the highest Spearman's rank correlation between the CiteRank ranking and the ranking by the number of citations in the future was 0.57, while the highest correlation between FutureRank and



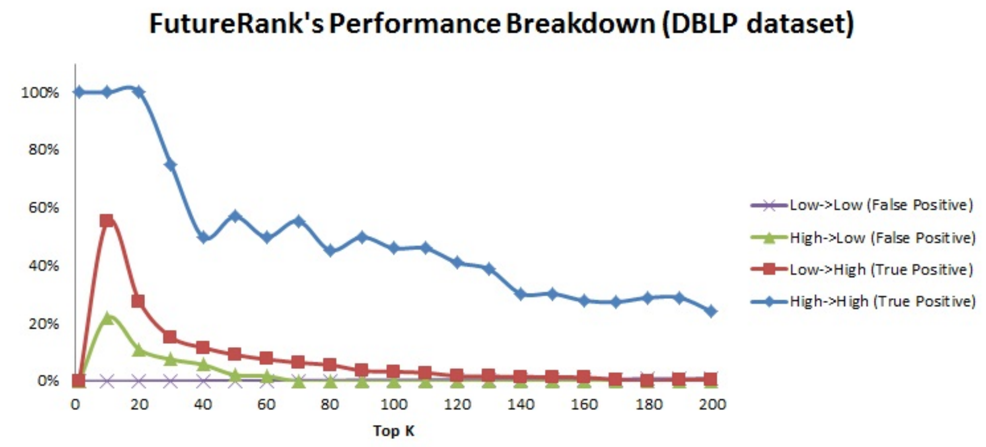


Figure 3.8: Percentage of the papers from each category that are among the top k output of FutureRank.

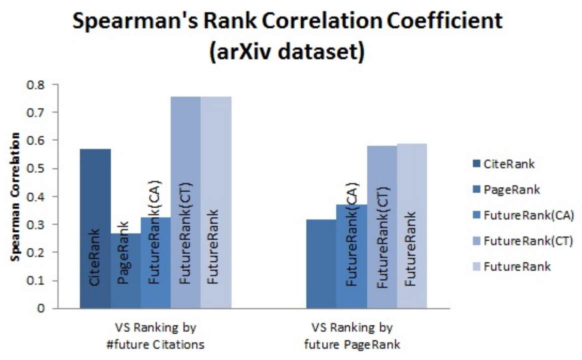


Figure 3.9: Correlation between ranking on the validation data and the results of CiteRank and FutureRank

the ranking by the number of citations in the future was 0.74 (the publication date is used as a tie-breaker if two articles have the same number of citations). This is a significant improvement. I also computed the correlation between the FutureRank ranking and the ranking by the future PageRank, which is a more desirable measure. The obtained correlation is 0.59, which is still more than the correlation obtained by CiteRank for the simpler measure of the number of future citations.

The parameters of the models shown in Figure 3.9 are as follows:

1. **FutureRank**: The best correlation is obtained by $\alpha = 0.4$, $\beta = 0.1$, and $\gamma = 0.5$

2. **FutureRank(CT)**: The best correlation is obtained by $\alpha = 0.5$, $\beta = 0$, and $\gamma = 0.5$.

3. **FutureRank(CA)**: The best correlation is obtained by $\alpha = 0.65$, $\beta = 0.35$, and $\gamma = 0$.

For the arXiv dataset, the highest correlation between FutureRank(CA) and the number of future citations is 0.34, which is consistent with its precision-recall curve. This shows the importance of publication time, which is ignored by methods such as CoRank. Furthermore, the highest correlation between the number of citations and

FutureRank(CT), which uses the same information as CiteRank, is 0.62. This also shows that in terms of precision and recall, FutureRank will significantly outperform CiteRank, although the precision-recall curve of CiteRank was not available for comparison. Figure 3.6 shows that the precision-recall curves of FutureRank(CT) and FutureRank are very similar and cross each other several times. Although the top results of both configurations are very similar, the correlations obtained for FutureRank are slightly better than those for FutureRank(CT).

The top 20 results for FutureRank within PageRank and FutureRank(CA) are shown in Table3.1. FutureRank, future PageRank (gold standard), the number of citations that papers obtained before 2001 (historical data), the number of citations that papers would obtain in and after 2001 (the future data that was not available to the FutureRank), and the publication dates are shown in Table 3.1. Since the top results for FutureRank and FutureRank(CT) are almost identical, results of FutureRank(CT) are not shown in Table 3.1.

In Table3.1, article 9906064, "An Alternative to Compactification," has only 414 citations before 2001. It received a good rank (3) by FutureRank. This paper was published in 1999, and obtained the 414 citations in less than two years. This suggests that the paper will receive many citations in the future and was a very useful paper at the query time (in 2001). Both the number of citations (617) the paper received after 2001 and the fact that it attained the second position in the ranking by future PageRank confirm the accuracy of FutureRank.

Table 3.1: Top 20 articles retrieved by FutureRank that were published before 2001 (arXiv dataset)

arXiv ID	**Title**	**Pub. Date**	**#Cit. bef. 2001**	**#Cit. aft. 2001**	**PageRank**	**FutureRank(CA)**	**FutureRank**	**Future PageRank**
9711200	The Large N Limit of Superconformal Field Theories and Supergravity	11/28/1997	1540	874	10	3	1	1
9802150	Anti De Sitter Space And Holography	2/23/1998	1137	638	28	6	2	4
9906064	An Alternative to Compactification	6/9/1999	414	617	92	129	3	2
9802109	Gauge Theory Correlators from Non-Critical String Theory	2/17/1998	1054	587	37	8	4	5
9908142	String Theory and Noncommutative Geometry	8/23/1999	471	673	131	26	5	3
9407087	Monopole Condensation	7/19/1994	1082	217	1	1	6	10
9610043	M Theory As A Matrix Model: A Conjecture	10/8/1996	922	277	14	7	7	8
9510017	Dirichlet-Branes and Ramond-Ramond Charges	10/5/1995	937	218	4	5	8	14
9711162	Noncommutative Geometry and Matrix Theory: Compactification on Tori	11/21/1997	449	339	106	91	9	7
9905111	Large N Field Theories	5/17/1999	352	455	174	80	10	6
9503124	String Theory Dynamics In Various Dimensions	3/21/1995	981	133	2	4	11	46
9408099	Monopoles	8/19/1994	856	150	6	2	12	25
9510135	Bound States Of Strings And *p*-Branes	10/19/1995	675	100	13	9	13	77
9510209	Heterotic and Type I String Dynamics from Eleven Dimensions	10/30/1995	546	242	40	18	14	9
9611050	TASI Lectures on D-Branes	11/11/1996	594	107	76	23	15	29
9409089	The World as a Hologram	9/20/1994	266	161	95	31	16	15
9711165	D-branes and the Noncommutative Torus	11/24/1997	297	159	211	108	17	33
9204099	The Black Hole in Three Dimensional Space Time	5/8/1992	291	89	69	37	18	62
9410167	Unity of Superstring Dualities	10/25/1994	672	76	5	12	19	94
9603142	Eleven-Dimensional Supergravity on a Manifold with Boundary	3/22/1996	314	180	167	70	20	17

Finally, the top 20 authors retrieved by FutureRank and the number of their publications are listed in Table 3.2.

3.2.7 Running Time and Convergence

The convergence of FutureRank is measured as if the score difference between two consecutive iterations is less than some threshold *minDifference*. The score difference of two consecutive iterations is computed as follows:

$$\begin{aligned} \mathit{Difference} &= \sum_{p_j \in P} (R_j^{Pi} - R_j^{Pi-1})^2 \\ &+ \sum_{a_j \in A} (R_j^{Ai} - R_j^{Ai-1})^2 \end{aligned}$$

While the convergence rate was different for different parameter sets, the model converges very fast in most cases. Figure 3.10 shows the difference between the scores computed in two consecutive steps. FutureRank and FutureRank(CT) have very similar behavior and, apparently, both of them converge much faster than FutureRank(CA).

Figure 3.11 shows the precision of the top 50 results. It shows that each model converges after three or four iterations, while the results reported for CiteRank were obtained after 20 iterations. Hence, in addition to better precision and correlations, FutureRank converges in many fewer iterations than CiteRank.

3.3 Discussion

This chapter presented the FutureRank algorithm, which is able to combine information about citations, authors, and publication time to effectively predict the future PageRank of a paper. While the impact of a paper at any time can be measured by the number of current citations, the number of the citations that a

Table 3.2: Top 20 authors retrieved by FutureRank and the number of their articles in the dataset

Rank	**Name**	**# of Publications**
1	Edward Witten	100
2	Ashoke Sen	89
3	A.A. Tseytlin	111
4	Zurab Kakushadze	63
5	Joseph Polchinski	47
6	Juan M. Maldacena	21
7	Donam Youm	55
8	Nathan Seiberg	45
9	Cumrun Vafa	78
10	John H. Schwarz	47
11	Michael R. Douglas	52
12	Andrew Strominger	65
13	Nathan Berkovits	59
14	P.K. Townsend	56
15	Sergei V. Ketov	51
16	Miao Li	52
17	C.N. Pope	129
18	Shinichi Nojiri	94
19	N. Seiberg	23
20	Ichiro Oda	37

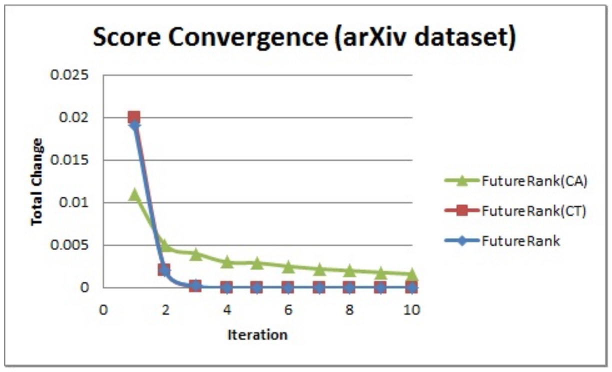


Figure 3.10: Score convergence. The vertical axis shows the difference between the score values in two consecutive steps, so zero difference reflects the convergence

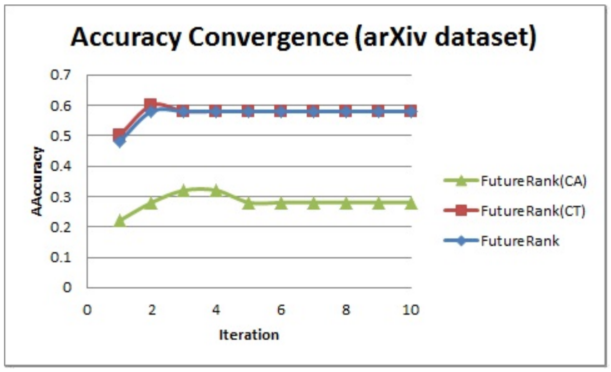


Figure 3.11: Precision of top 50 results after each iteration

paper will obtain in the future measures how useful the paper is at the query time. This makes the future PageRank a better measure for retrieving articles that will help researchers find the most relevant articles. Experimental results showed that FutureRank is a significant improvement over other recently proposed algorithms. Also, the precision-recall curve shows that FutureRank(CT), which only uses the publication time of the article and the current citations, significantly outperforms the traditional PageRank, which uses only the current citations. FutureRank obtained a much higher correlation score, in addition to faster model convergence. It also learned the decay-factor parameter from the data, while CiteRank had to perform an exhaustive search.

The ranking model introduced in this chapter was designed for a particular application: predicting future PageRank. In Chapter 5, I present a learning-to-rank algorithm that can learn ranking models such as FutureRank. It can also be trained on user feedback and is powerful enough to learn any authority flow-based ranking function in heterogeneous networks.

Chapter 4

Personalized Ranking in Social Media

A major approach to personalization is based on authority flow-based ranking [23, 57, 4]. There are two approaches to personalization in authority flow-based ranking. The first method uses a base set of entities of known interest to the user; e.g., [57] suggests using a user's favorite pages as the base set in PageRank, whereas [23] proposes computing a set of PageRank topic vectors, one for each representative topic. The second personalization method, such as ObjectRank [4], adjusts the importance of various types of semantic connections. ObjectRank models nodes as entity types and groups edges by their edge type or semantic type. It can personalize ranking in heterogeneous networks by a *weight assignment vector*, which associates a personalized authority-flow weight with each edge type. The weight of each edge determines the importance of the corresponding relationship.

This chapter presents a personalized ranking algorithm for a blog dataset, which is based on the second approach, ObjectRank. This chapter has several objectives. First, a blog dataset must be enhanced with additional nodes, e.g., authors, events, etc., and corresponding edges in order to apply ObjectRank. This leads to a form of entity-relationship graph, which conforms to an entity-relationship schema. For example, an entity-relationship schema to facilitate authority flow ranking includes four entity types–*BlogPost*, *Author*, *Category* (explicit topic of a post), and *Event* (discovered collection of posts)–as well as the corresponding edge types, as shown in Figure 4.1. As mentioned above, while some entity types are easily identified in social media, e.g., author and category, other entity types such as *Event* must be generated, as will be discussed.

The second objective is to develop a suite of authority flow-based personalized ranking techniques. The list of personalization techniques developed in this study are as follows:

- pPR: Personalized PageRank (pPR) on a restricted graph that has one entity type *BlogPost* and a reflexive edge from *BlogPost* to *BlogPost*, where an edge denotes the text similarity of two BlogPosts. Personalization for pPR is represented by a personalized *base set* of *BlogPost* entries, where the base set is defined as the set of nodes of a graph from which the authority flow originates.

- pOR: ObjectRank (OR) [4] is an authority flow-based ranking technique for entity-relationship data graphs. A key difference between ObjectRank and PageRank is that in ObjectRank, different edge types (e.g., BlogPost-to-Author) carry different amounts of authority flow. In contrast to the single-node-type graph of pPR, the richer entity-relationship schema graph of Figure 4.1 is used in pOR. OR is associated with a Weight Authority Vector, which associates a personalized authority flow weight with each edge type. pOR uses a *default value of equal authority flow* for each edge type.

- pOR$^+$: The weight for each edge type is customized based on user profile. For instance, users who typically pick blogs based on the author have a high BlogPost-to-Author weight. Clearly, pOR$^+$ offers more flexibility and customization than pPR and pOR, and, as will be shown, achieves the best personalized ranking.

- pIR: This is an extension of the Apache Lucene [44] text-search engine, in which text-similarity algorithms are employed. pIR is a baseline for comparison. pIR achieves personalization using a user-specified set of BlogPosts, the personalized base set. The favorite words for each user are extracted from the base set documents.

The next objective is to study the accuracy of personalized recommendations, for which a subset of the Spinn3r dataset [9]. A human-subject study, as well as an experiment with various types of *synthetic users*, is performed to measure the effectiveness of these algorithms. Each synthetic user has a profile represented by a personalized ranking of his daily favorite *BlogPost* entries. In particular, synthetic users who follow the posts of a set of bloggers (*Author Users*), the posts of a set of categories (*Category Users*), the posts related to a set of keywords (*Keyword Users*), or the posts related to a set of events (*Event Users*) are considered.

Results show that pOR can provide accurate personalization for the majority of real and synthetic users. The F_1-score, which combines the precision and recall measures, is used to compare the effectiveness of the various ranking techniques. The performance of pOR$^+$ for different types of synthetic users is examined, as well as, the robustness of pOR and pOR$^+$ ranking across many parameters that configure the synthetic users.

In recognition that real-user behavior may be more sophisticated than that of synthetic users, an evaluation with real users is performed. In this experiment, both pIR and pPR are used as the baseline algorithms and are compared with pOR, which shows superior performance. Despite the difficulty in constructing a profile for real users, result shows that pOR (even with non-personalized weights) can provide high-quality personalized recommendations for real users.

In summary, this chapter makes the following contributions:

- The social media dataset is enhanced with additional nodes and edges in order to apply ObjectRank. A new event discovery algorithm that is efficient for social media data is introduced. Further, a massively parallel document similarity technique, using the MapReduce paradigm, is exploited to measure the similarity between BlogPosts (Section 4.1).

- A suite of novel and baseline ranking techniques is presented for the personalized recommendation of BlogPosts (Section 4.2).

- Effectiveness of the proposed ranking techniques is evaluated using both real and synthetic users. Results demonstrate that pPR and pOR can significantly outperform the baseline pIR for real users. Further, pOR outperforms pPR, and pOR$^+$ outperforms all variants[1] (Section 4.3).

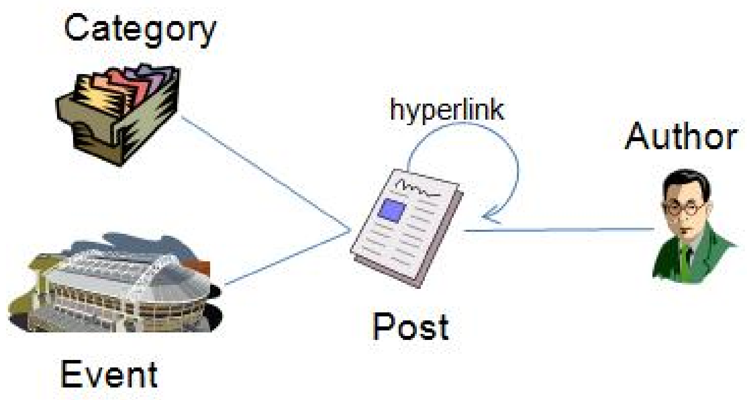


Figure 4.1: Enhanced Social Media Schema Graph

4.1 Enriching Social Media Dataset

The dataset provided by Spinn3r.com contains 44 million blog posts made between August 1 and October 1 2008 [9]. The post includes both text and meta-data such as the blog's homepage, timestamp, etc. Data are formatted in XML and further arranged into tiers approximating search-engine ranking. I first discuss enriching the social media schema graph and dataset preparation and then briefly summarize document-similarity computation and event identification.

[1]Due to the complexity of finding the best personalized weights for real users, pOR$^+$ was not considered in the real-user study. This will be addressed in Chapter 5.

4.1.1 Schema Graph for Personalized PR (pPR) and ObjectRank (pOR)

A shortcoming of social media is the lack of a rich hypergraph to determine the importance of pages. In this study, following the example of projects on ranking collections of documents that are unconnected by hyperlinks, the pairwise document similarity between two *BlogPost*s is computed first and inserted as a bidirectional link between two documents, as seen in Figure 4.1; the label *doc-sim-weight* reflects the document-similarity value for each link.

Next, an entity-relationship schema with four entity types, *BlogPost*, *Author*, *Category* (topic of the post), and *Event*, is utilized to reflect authority flow. Figure 4.1 shows the schema. The concepts of *BlogPost*, *Author*, and *Category* are intuitive, and these nodes are easily identified in social media datasets. The concept of an *Event* and the process used to identify event nodes will be discussed in the next section. Also, links represent the associations between *BlogPost* and *Category*, *BlogPost* and *Author*, and *BlogPost* and *Event*, with the reflexive edge from *BlogPost* to *BlogPost* representing document similarity.

4.1.2 Data Graph

A 31-day subset of the data (August 2008) is used to create data graphs for both training and testing. After removing non-English posts and posts without Author or Category information, the subset contains approximately 800,000 posts.

To create a data graph that is appropriate for personalized ranking, more filtering is applied so that the distribution of posts per author, posts per category, or categories per post reflect a non-sparse and normalized distribution. Furthermore, a sufficient amount of data is needed for training and testing. For example, for a *synthetic user* who is following a particular author, the author nodes in the dataset will be restricted to authors who have more than H posts. Note that many posts are not labeled with category labels. In addition, category labels can be inconsistent and sparse, since category labels are chosen arbitrarily by authors. Hence, some category labels are only used by one author or a few authors.

The following procedure is used to create a data graph for personalization:

1. Identify frequent categories as categories that contain more than 50 posts.
2. Identify frequent authors as authors who have at least 10 posts from the frequent categories.
3. Select the posts written by the frequent authors that are associated with the frequent categories.

The final dataset comprises a data graph of

- 248908 nodes, including:
  - 137047 *BlogPosts*,
  - 2210 *frequent Authors*,
  - and 109651 *frequent Categories*,
- and 1528514 edges, including:
  - 137047 Author_BlogPost edges,
  - 794005 Category_BlogPost edges,
  - 15270 Event_BlogPost edges,
  - and 582192 BlogPost_BlogPost edges.

4.1.3 Computing Document Similarity for BlogPosts

Document similarity has been used for many Web search applications, including ranking and crawling [34, 59]. Computing pairwise document similarity between all pairs of *BlogPost*s is an expensive step in enhancing the entity-relationship data graph. In this work, the cosine similarity between pairs of documents is used as *doc-sim-weight* value of the link between the corresponding documents. The implementation uses the MapReduce/Hadoop cloud computing framework [16, 41]. The MapReduce paradigm performs a parallel computation over a large number of records to obtain partial results, which are then aggregated. Based on functional programming, MapReduce provides an abstraction, in which the programmer defines a mapper and a reducer. Hadoop [41] provides a distributed computing platform to evaluate a Map/Reduce program.

Recall that the evaluation dataset included more than 100000 posts. Including pairwise document similarity edges between each pair of posts would make the graph very large, and result in a very expensive ranking computation without necessarily improving the accuracy of the personalized recommendation. Hence, for each post, *BlogPost_BlogPost* edges are limited to the 5 most similar posts.

4.1.4 Identification of Events

Social media activity is often triggered or clustered around an event or topic; loosely, an event or topic is a specific occurrence at a specific time and/or in a specific place. When posts are clustered around an event, users may also be interested in following those posts related to the event or topic of interest.

Many successful topic detection and tracking (TDT) approaches have been studied in the literature [2, 40, 71]. TDT approaches have typically used clustering algorithms, in which documents are treated as database records and words as features. Next, variations of TF/IDF are used to compute feature values, and cosine similarity is used as a similarity (or distance) measure. The next generation of event detection approaches recognizes that a core group of words will be common to the documents related to an event. In fact, the existence of such words for each

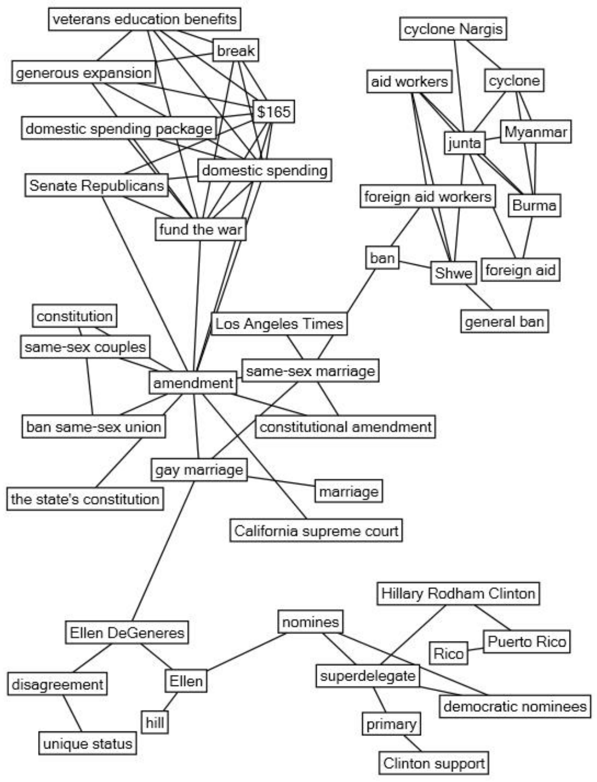


Figure 4.2: An example of KeyGraph

topic makes the content of related documents similar to each other, as measured by variations of TF/IDF. Hence, such keywords are important features for topic or event detection. More importantly, they can scale to handle large noisy social media data collections.

KeyGraph, a graph analytical approach that was designed for large-scale social media datasets, was used to extract the events from the evaluation dataset. The extracted events/topics were then used to enhance the social media schema graph. Details of the approach are in [61, 62]. The algorithm includes the following steps:

1. Construct a word co-occurrence graph, labeled a KeyGraph (see Figure 4.2).

2. Perform community structure analysis of the KeyGraph to decompose into multiple communities.

3. Create an event corresponding to each community.

4. Cluster documents using the KeyGraph communities.

KeyGraph identified multiple communities in the KeyGraph of Figure 4.2. I briefly describe an example of an event identified in the experimental dataset: the *cyclone in Myanmar in 2008.* The event is identified by the following constellation of keywords, cyclone, Myanmar, foreign aid workers, Burma, ban, etc. Fourteen key events identified in this dataset were evaluated using 3 human evaluators (via Amazon's Mechanical Turk). When averaged across all events and all evaluations, 72% of the posts were identified as *very relevant* or *relevant* to the assigned event. Further, 25% of the posts were also judged to be somewhat relevant by evaluators, and only 3% of posts were judged to be not relevant. Thus, enhancement of the blog dataset using events is expected to yield a high-quality enhanced dataset.

4.2 Personalized Authority Flow-based Ranking

Personalization of Authority Flow-based Ranking: Two approaches to personalization are used in *authority flow-based ranking.* The first method considers a set of preferred objects or topics. Personalization with a base set involves selecting user specific objects as the source of the authority in the data graph. [57] suggests using a user's favorite pages as the base set in PageRank, whereas [23] proposes computing a set of PageRank topic vectors, with one for each representative topic.

4.2.1 Authority Flow-based Ranking: The ObjectRank Algorithm

The second personalization method adjusts the importance of various types of semantic connections. ObjectRank [4] personalizes ranking in Entity-Relationship graphs; it models nodes as entity types and groups edges by their edge type or semantic type. Then, the authority flow is personalized by a *weight assignment* *vector*. The weights determine the importance of each type of association in the ranking. For example, in Figure 4.1, there are 4 edge types associating *BlogPost* to *BlogPost*, to *Author*, etc. Bidirectional edges have two associated weights, one for each direction. Varadarajan et al. [67] present techniques to learn the weights using relevance feedback.

The transition matrix A_{OR} of ObjectRank depends on the authority transfer weights specified on the *schema graph*; however, A_{OR} is defined at the level of the data graph. To demonstrate the relationship of the ObjectRank transition matrix and the PageRank transition matrix, without loss of generality, one may assume that objects of the same type are grouped together. Consider an authority transfer schema graph with t types. The *weight assignment vector* = $\{\alpha_{1,1}, \alpha_{1,2}, ..., \alpha_{1,t}, \alpha_{2,1}, \alpha_{2,2}, ..., \alpha_{2,t}, ..., \alpha_{t,1}, \alpha_{t,2}, ..., \alpha_{t,t}\}$ represents the authority transfer weights. A_{OR} contains $t \times t$ submatrices. Each submatrix entry of the transition matrix A_{OR} is multiplied by the authority transfer weight for the corresponding semantic edge type. A_{OR} can be expressed as follows:

$$A_{OR} = \begin{pmatrix} \alpha_{1,1}A_{1,1} & \alpha_{1,2}A_{1,2} & \cdots & \alpha_{1,t}A_{1,t} \\ \alpha_{2,1}A_{2,1} & \alpha_{2,2}A_{2,2} & \cdots & \alpha_{2,t}A_{2,t} \\ \vdots & \vdots & & \vdots \\ \alpha_{t,1}A_{t,1} & \alpha_{t,2}A_{t,2} & \cdots & \alpha_{t,t}A_{t,t} \end{pmatrix} \tag{4.1}$$

The submatrix $A_{p,q}$ contains authority transfer probabilities from objects of type p to objects of type q. Let $e^T(v_i, v_j)$ be the semantic type of edge (v_i, v_j) in the data graph. Let $\alpha(e^T(v_i, v_j))$ denote the weight assignment for $e^T(v_i, v_j)$. $OutDeg(v_i, e^T(v_i, v_j))$ is the number of outgoing edges from page v_i, of type $e^T(v_i, v_j)$. The submatrix $A_{p,q}$ is defined as follows:

$$A_{p,q}[i,j] = \begin{cases} \frac{1}{OutDeg(v_i, e^T(v_i, v_j))} & \text{if exists } (v_1, v_j) \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Let P represent a personalized baseset at the level of the data graph. Let A_{OR}^T denote the transposition of A_{OR}. The ObjectRank vector R is recursively defined as follows in Equation (5.3):

$$R = \epsilon A_{OR}^T \cdot R + (1 - \epsilon)P \tag{4.3}$$

4.2.2 Personalized Ranking Variants

In this study, the following personalization models are evaluated on the enriched social media graph of Figure 4.1.

- pPR: This is a personalized PageRank (pPR) that is applied to a restricted graph that has one entity type, *BlogPost*, and a reflexive edge from *BlogPost* to *BlogPost* with a document similarity edge weight. There will also be a personalized base set of *BlogPost* entries. The details of computing document similarity and the choice of queries and personalized base set for evaluation will be discussed later. This variant pPR will be the baseline for the evaluation of personalized ranking.

- pOR: Personalized ObjectRank (OR) will be evaluated on the entity-relationship schema graph of Figure 4.1. For this variant, the edge weights are set to *default weights*, i.e., equal weights for all outgoing types of edges from each type of node. Thus, this variant of pOR determines the impact of only enriching the social media schema graph, but not using personalized weights. pOR will also use a personalized base set of *BlogPost* entries.

- pOR$^+$: Personalized ObjectRank, where personalization consists of both a personalized base set of *BlogPost*s as well as a personalized set of weights for each edge type.

- pIR: This is an extension of the Apache Lucene text search engine [44]. Personalization is implemented by using all the keywords in the personalized baseset of each BlogPost entry to create a document query [72]. This will also serve as a baseline.

In this study, edge weights are set manually based on the type of synthetic user. Handpicked weights are reported in the evaluation section. pOR$^+$ reflects the (maximum) expected benefit of an enriched social media schema graph, authority flow-based ranking, and complete knowledge of the profile of the synthetic user. Since the synthetic users had simple profiles, e.g., they followed posts about some event, it was straightforward to manually determine the best weights to be used in pOR$^+$. In contrast, real users are difficult to model. This problem will be addressed in the next chapter.

4.3 Evaluation

This section presents the results of an experimental evaluation of the accuracy of personalized authority flow-based ranking using the enhanced Spinn3r dataset. The first set of experiments demonstrates the accuracy of pOR for several classes of synthetic users. The sensitivity of pOR to the types of synthetic users is also examined.

The second set of experiments provides a comparative analysis of personalized PageRank (pPR), pOR and pOR$^+$, in which the personalized weights have been manually chosen to reflect each synthetic user's profile. pPR represents a *baseline*, since the social media dataset is not completely enhanced with additional node or edge types. Similarly, pOR$^+$ represents the maximum benefit of enhancing the dataset and using authority flow-based ranking. The robustness of the pOR and pOR$^+$ ranking across many parameters that configure synthetic users is evaluated as well.

Finally, in recognition that real-user behavior may be more sophisticated than that of synthetic users, a third set of experiments using an evaluation with real users is conducted as well. In this experiment, both pIR and pPR as used as the baseline. The performance of baseline algorithms is then compared with that of pOR. Recall that choosing the best personalized weights for the complicated profiles of real users is too difficult to be done manually. Hence, pOR$^+$ is not evaluated in this experiment. Despite the difficulty of constructing a profile for real users, results show that even pOR with no personalized weights can provide high-quality personalized recommendations for real users.

4.3.1 Synthetic User Experiment

Four classes of synthetic users are considered in this experiment:

- Author Users: These users are interested in posts by specific authors; both the *personalized training base set* and *testing set* are from posts by a selected author.

- Category Users: These users are interested in posts labeled with specific categories; both the *training* and *testing* posts are posts that are labeled with a specific category.

- Keyword Users: These users are interested in posts that are most relevant to some set of keywords. All *BlogPost*s were indexed using the Apache Lucene [44] text search engine, and the Top K=100 posts for each keyword are retrieved to serve as the ground truth.

- Event Users: These users are interested in posts about a specific event; both the *personalized training base set* and *testing set* are posts that are labeled with a specific event, as discussed in a previous section.

I consider several parameters to test the sensitivity and robustness of the personalization variants. For each synthetic-user experiment, I vary three parameters, as follows:

1. The first parameter, D, is the number of distinct queries for a synthetic user. For example, for $D = 2$, an Author User will follow the posts of 2 authors or a Category User will follow posts labeled with 2 Category keywords.

2. The second parameter, H, reflects the cardinality of the *personalized training base set*. The set of ground truth *BlogPost*s will be sorted in chronological order and partitioned into two parts; the first H posts are for the *personalized training base set*.

3. The third parameter, U, indicates the number of synthetic users generated for each experiment. Results are reported as an average over U users.

For each experiment, the *personalized training base set* is provided to pOR. Default weights are equal values for all outgoing edges from each type of node ((Author_BlogPost = 1.0, Category_BlogPost = 1.0, Event_BlogPost = 1.0, BlogPost_Author = 0.25, BlogPost_Category = 0.25, BlogPost_BlogPost = 0.25, BlogPost_Event = .25). After training, pOR then returns a set of personalized recommendations, *pRec*. The cardinality of the *pRec* set is chosen to be the cardinality of the ground truth *testing set*.

Note that for Author, Event, and Category Users, one can only determine whether a recommended post in *pRec* is contained in the ground truth and is relevant to the user, or if it is not contained in the ground truth and thus is not relevant to the user. For Keyword Users, one could utilize the ranking provided by the search engine. However, for uniform reporting of experimental results, a binary decision of relevant or not relevant is made to determine whether the recommended post in *pRec* is contained in the ground truth.

To measure the accuracy of the personalized ranking, I report on the $F1$ measure for *pRec*, averaged over the U users. $F1$ is the harmonic mean of precision and recall; it has a best-case score of 1.0 and a worst-case score of 0. Recall that the cardinality of *pRec* is identical to the cardinality of the ground truth *testing set*. Hence, in the experiments, the value of $F1$ for *pRec* is equal to both the precision and the recall.

4.3.1.1 Comparison of Ranking Algorithm Variants

I compare the performance of the ranking variants for all type of users, as follows:

- pIR: The posts in the user's base set are concatenated into a single user profile document P. Next, the candidate-recommended posts are ranked by their similarity to P. The similarity is the cosine similarity that is returned by Lucene [44]. This variant serves as one of the basline algorithms in this experiment.

- pPR: This uses the schema graph of Figure 4.1, restricted to one node type, *BlogPost*, and one reflexive document similarity link. This variant serves as the second baseline algorithm.

- pOR-NE: This uses the schema graph of Figure 4.1 without the *Event* node and without the edge type *Event-BlogPost*; here, the purpose is to compare both pPR and pOR when neither can benefit from the *Event* node for Event Users.

- pOR: This uses the schema graph of Figure 4.1, including the *Event* node, and the edge type *Event-BlogPost*. Personalized weights were chosen using the following simple heuristic: A weight of 1.0 was chosen for the authoritative edge types, e.g., the Author_BlogPost edge type or the Category_BlogPost edge type. Other edge types where the authority flow is not known, e.g., the BlogPost_BlogPost edge type or the backlinks, such as the BlogPost_Category edge type, were given a weight of 0.25 to reflect the lower authority flow. The weights are as follows: (Author_BlogPost = 1.0, Category_BlogPost = 1.0, Event_BlogPost = 1.0, BlogPost_Author = 0.25, BlogPost_Category = 0.25, BlogPost_BlogPost = 0.25, BlogPost_Event = .25)

- pOR$^+$: This is similar to pOR, with the additional benefit that weights are adjusted manually to be more appropriate for the profile of an Event User. An Event User was observed in experiments to be a good reflection of human user behavior. The weights are as follows: (Author_BlogPost = 1.0, Category_BlogPost = 1.0, Event_BlogPost = 1.0, BlogPost_Author = 0.2, BlogPost_Category = 0.2, BlogPost_BlogPost = 0.2, BlogPost_Event = .4). Note that adjusting the weights manually is not ideal. The best approach is to learn the weights from user feedback, as will be addressed in the next chapter.

Figure 4.3 shows the superiority of pOR, in comparison to the authority-flow baseline of pPR and full-text-search baseline of pIR, for Author Users, Category Users, Keyword Users, and Event Users. The most significant improvement observed was for Author Users. Surprisingly, pIR does not perform well for Keyword users. This is because the user profile document P is too big, so the weight of the user's profile keywords is not high.

Figure 4.4 reports on the $F1$ values for the four variants for Event users. This figure shows that the baseline algorithm pIR has an extremely low $F1$ value. pPR and pOR-NE have very similar low $F1$ values. However, pOR shows a significant

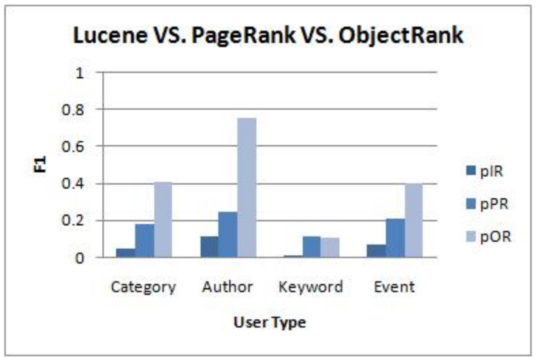


Figure 4.3: $F1$ Values of pIR and PageRank versus ObjectRank for Author and Category users (U=50, D=6, H=30), Event users (U=15, D=1, H=30), and Keyword users (U=10, D=1, H=30).

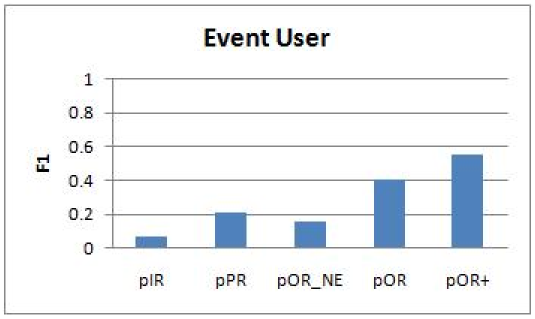


Figure 4.4: $F1$ Values for Event Users with $D = 1$ and $H = 30$ and $U = 15$.

improvement (approximately 100% increase) in the $F1$ value. Similarly, there is another significant increase of the $F1$ value (approximately 50%) when comparing pOR^{+} to pOR. To summarize, both pOR and pOR^{+} provide significant improvements over the baselines pIR and pPR for Event Users.

The Wilcoxon Signed Rank test for statistical significance shows that the F1 score for pOR significantly dominates pPR, and pPR is significantly dominates pIR for author and category users at the 95 confidence level. Furthermore, the F1 scores for pOR and pPR significantly dominate pIR for keyword and event users at the 95 confidence level. Although pOR outperforms pPR for 5 events out of 7, it is not statistically significant due to the small size of the sample data.

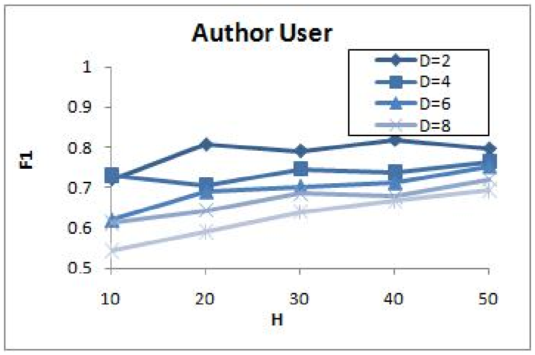


Figure 4.5: Average $F1$ Values of pOR for 50 Author Users ($U = 50$) with Varying Values of $D = 2, 4, 6, 8, 10$ and $H = 10, 20, 30, 40, 50$.

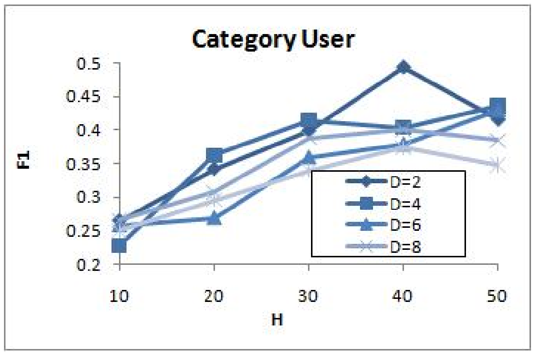


Figure 4.6: Average $F1$ Values of pOR for 50 Category Users ($U = 50$) with Varying Values of $D = 2, 4, 6, 8, 10$ and $H = 10, 20, 30, 40, 50$.

4.3.1.2 Parameter Sensitivity Analysis

Next, I report on the sensitivity and robustness of the personalization variants. Recall the three parameters of my experimental setup: D, the number of distinct queries for a synthetic user; H, the cardinality of the *personalized training base set*; U, the number of synthetic users for each experiment.

Figure 4.5 reports the value of $F1$ for Author Users. The X axis varies the value of H from 10 to 50. Each of the curves in the figure corresponds to values of D from 2 to 10. Each data point in each plot is averaged over $U = 50$ Author Users. As observed in the figure, the value of $F1$ is highest for lower values of $D = 2$ compared to higher values of $D = 10$; this is because larger values of D represent a diversity of queries for each Author User. As expected, the $F1$ value increases

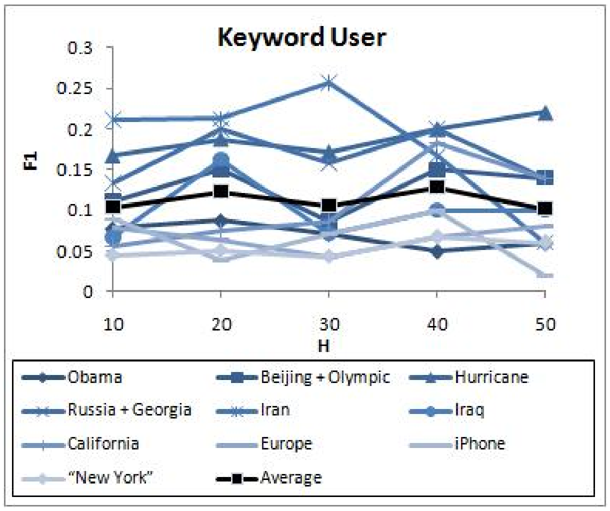


Figure 4.7: $F1$ Values of pOR for 10 Keyword Users with Varying Values of $H = 10, 20, 30, 40, 50$.

as the value of H increases or the cardinality of the *personalized training base set* increases. This benefit from increasing H values is observed clearly for the case of $D = 10$. As the value of H approaches 50, the $F1$ values converge, indicating that there is no additional benefit of higher H values. Overall, personalized pOR has high accuracy for Author Users.

Figure 4.6 reports on Category Users. This figure shows similar trends for varying values of D and H. For $H = 10$, the $F1$ values are low for all values of D; this reflects an insufficient *personalized training base set*. As the value of H increases to 50, a significant improvement in the $F1$ values occurs. However, the $F1$ values for Category Users appear to be lower than the $F1$ values for Author Users. To explain, each post is associated with exactly one Author, but can be associated with multiple Category keywords. Hence, pOR is able to better exploit a post in the *personalized training base set* for Author Users and make accurate recommendations.

Figure 4.7 reports on Keyword Users, in which a *single* Keyword query or $D = 1$ is chosen for each user. Given that the ground truth for Keyword Users is determined using a search engine and the fact that the social media graph does not include keywords, high $F1$ scores are not expected. The $F1$ scores for each of the 10

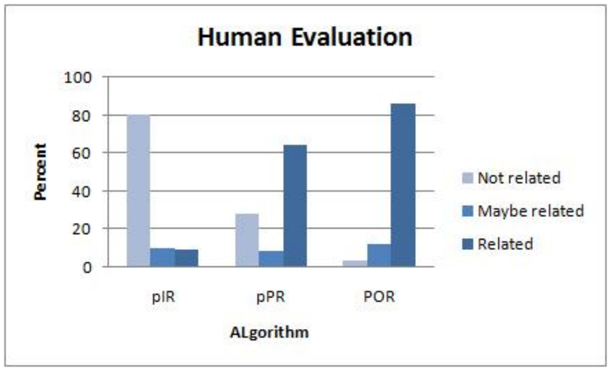


Figure 4.8: Results of Experiment with Real Users

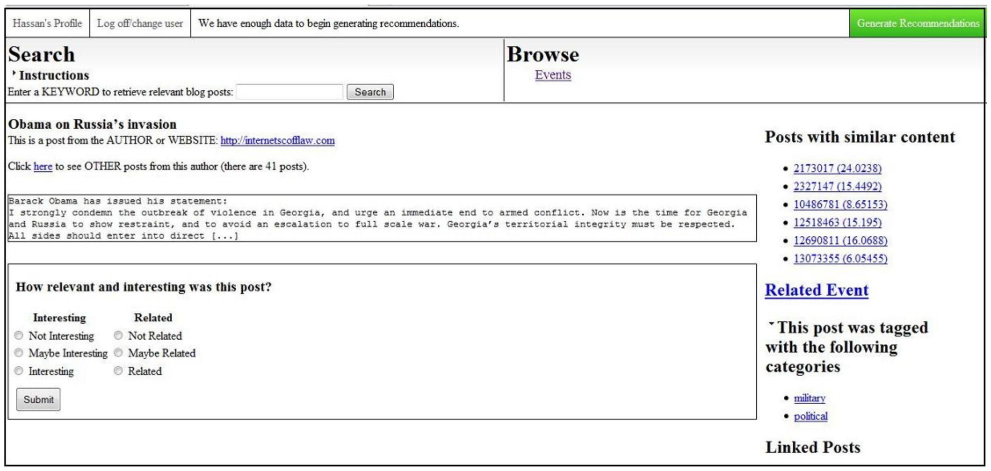
Keyword users are reported individually, as well as the average over all of them; the Keywords are also indicated in the figure. As observed in the figure, the $F1$ scores for Keyword Users are low, and there does not appear to be much benefit from increasing H. Each post typically contains multiple Keywords, and it is a challenge for pOR to exploit a post in making a personalized recommendation.

To summarize, personalized pOR provides high accuracy for Author Users. pOR provides reasonably high accuracy for Category Users with improving performance as the cardinality of H, the *personalized training base set*, increases. While the accuracy for Keyword Users is low, I know of no other personalized ranking for social media that currently provides better accuracy for personalized recommendations across all types of synthetic users.

4.3.2 Evaluation with Real Users

I recognize that real users are more sophisticated and cannot be modeled using a single criterion, as was done for synthetic users. I report an experiment with up to 10 real users. Users were solicited from an undergraduate course in the Smith School of Business and through a message to graduate students in computer science. The majority of users were familiar with the concept of both ranking and personalization.

The protocol was structured as follows:



We have enough data to begin generating recommendations.

Enter a KEYWORD to retrieve relevant blog posts:

Obama on Russia's invasion

This is a post from the AUTHOR or WEBSITE: http://internetscofflaw.com

Click here to see OTHER posts from this author (there are 41 posts).

```
Barack Obama has issued his statement:
I strongly condemn the outbreak of violence in Georgia, and urge an immediate end to armed conflict. Now is the time for Georgia
and Russia to show restraint, and to avoid an escalation to full scale war. Georgia's territorial integrity must be respected.
All sides should enter into direct [...]
```

How relevant and interesting was this post?

Posts with similar content

- 2173017 (24.0238)
- 2327147 (15.4492)
- 10486781 (8.65153)
- 12518463 (15.195)
- 12690811 (16.0688)
- 13073355 (6.05455)

This post was tagged with the following categories

Figure 4.9: The interface for human user experiment

- Each user was presented with a tool to browse the BlogPost dataset. They were free to follow posts using a keyword search, posts labeled with a specific Category, posts by a specific Author, or posts labeled by a specific Event.

- The training period was 30 minutes. Users were informed that the objective was to learn their browsing profile, and they were instructed to be more focused in their browsing behavior. Users had to provide feedback while browsing the posts. The feedback rated the posts as *Relevant*, *May be relevant*, or *Not relevant* to their search profile or query. I asked each user to provide up to 30 relevant posts.

- User feedback was then provided to the 3 ranking variants, pIR, pPR, and pOR. Fifteen recommended documents were obtained from the 3 methods and combined in a random order to create a dataset *pRec* of 45 recommended posts.

- Users were asked to rate the posts as in the training phase, with feedback ranging from *Relevant* to *May be relevant* to *Not relevant* to their original search profile or query from the training phase.

Figure 4.9 shows the interface for the real-user experiment; users could browse posts, provide training data, and then rate the recommendations.

In this experiment, pIR is defined as follows: It uses the content similarity of posts, which was already computed to build the post-to-post links in the graph. For each blog post, I find all the similar blog posts in the blog post baseset given by the user and calculate the sum of the similarity values. The total sum shows the content similarity of the posts to the user blog post baseset. The total similarity is then used to rank and retrieve the top 15 posts. Note that Lucene's document-query is also a solution.

Figure 4.8 reports on the accuracy of each of the three algorithms. pOR outperformed both pIR and pPR, and pPR outperformed pIR. On average, users ranked at least 90% of the recommended posts by pOR to be *Relevant* or *May be relevant*; this value was approximately 70% for pOR but less than 20% for pIR. The number of posts that were ranked *Relevant* was 86% for pOR and 64% for pPR. This number was very low, at 9% for pIR. The number of posts that were ranked *May be relevant* varied from 12% for pOR to 8% for pPR and 10% for pIR. Similarly, the number of posts that were ranked *Not relevant* varied from 3% for pOR to 28% for pPR to 80% for pIR.

The evaluation with real users showed that pOR significantly outperformed pIR. Note that this is not surprising. To explain, in my pre-tests with users, I observed that users followed posts by an *Author* or posts labeled with an *Event*. Hence, comparing the accuracy of recommendations *pRec* from pOR to the posts recommended by pIR, which are retrieved from a search engine that does not distinguish *Author*-related or *Event*-related posts, gave a significant advantage to pOR.

4.4 Discussion

In this chapter, I extended a social media dataset and provided accurate personalized authority flow-based ranking for both synthetic and real users. I then presented a suite of blog ranking techniques, which were then evaluated by both real and synthetic users. One main shortcoming was the fact that personalized edge weights were set manually. In Chapter 5, I introduce a learning algorithm that uses relevance feedback to learn the best edge weights.

Chapter 5

Learning to Rank in Heterogeneous Networks

In this chapter, I address the problem of learning to rank in networks. As discussed in section 2.3, *Learning-to-Rank* is a type of supervised machine learning problem in which the goal is to automatically construct a ranking model from given training data. For example, in information retrieval, query-document pairs are usually represented by numerical vectors called feature vectors, and a loss function associates a "cost" or "error" between a given ranking and the ideal ranking. The ranking model is then constructed by minimizing the error associated with the output of the model and the given training data. Existing learning-to-rank models [8, 10, 14, 17, 18, 24, 39, 52, 66, 70, 73, 74] are limited to one class of algorithms, namely, ranking based on content information or local features. Entity associations or relational features are ignored. Features are limited to the object's properties/attributes. Applying the current learning-to-rank models to network datasets is not trivial. Here, I briefly discuss the challenges in using current learning-to-rank algorithms to rank objects in networks. In an authority flow-based ranking model, ranking scores are computed as follows:

$$P = A \cdot P$$

where P is the score vector and A is the transition matrix. A is parameterized by the edge weights that have to be learned from the training data. According to $P = A \cdot P$, the score of each object is a function of the score of neighbor objects. In the context of learning to rank, this means that the features of each object are the scores of neighbor objects, which by default are all unknown. In current learning-to-rank solutions, all input features are known a priori, and the task is to learn the weight of each feature from training data. Due to the dependency between the scores of the adjacent objects, current learning-to-rank algorithms fail in network datasets.

This chapter introduces a framework for learning-to-rank in heterogeneous networks. It allows learning-to-rank algorithms from non-relational domains to be used for learning to rank in networks. In particular, I present pointwise and pairwise learning-to-rank models:

1. **Pointwise**: In this model, training data include the absolute ranking score of the training objects.

2. **Pairwise**: In this model, training data only include the preference for pairs of training objects.

I will present experimental results that demonstrate the success of this framework in learning the parameters of FutureRank as well as the personalized parameters for personalized ranking. The results show that both pointwise and pairwise models can successfully learn from as few as 10 training samples (per node type) and converge after a few iterations. In addition, both models are shown to be very robust against artificially-introduced noise in the training data.

The rest of this chapter is organized as follows: Section 5.1 describes the proposed learning framework. Section 5.2 presents pointwise and pairwise models, and Section 5.3 presents the experimental results.

5.1 Learning Framework

For a given graph $G = (V, E)$, edge (j, i) from object j to i has type $t(j, i)$, belonging to a set of types numbered $1, ..., T$. Each type t has an associated authority-flow weight represented by β_t. If $\beta_{t(j,i)}$ denotes the weight of edge (j, i), the transition matrix A is defined as follows:

$$A[i,j] = \beta_{t(j,i)} a_{j,i} \tag{5.1}$$

$$a_{j,i} = \begin{cases} \frac{1}{OutDeg(j,t(j,i))} & \text{if exists } (j,i) \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

where $OutDeg(j,t(j,i))$ is the number of outgoing edges from object j, of type $t(j,i)$. P^0 is a *personalization* vector that assigns an initial score p_i^0 to each object i. The score vector P is recursively defined as follows:

$$P = \alpha A \cdot P + (1-\alpha)P^0. \tag{5.3}$$

We can rewrite the score p_i for object i as a function of $T+1$ features, in which T is the number of edge types in the network and each edge type $t(j,i)$ is indexed $1..T$, as follows:

$$\begin{aligned} p_i &= (1-\alpha)p_i^0 + \alpha \sum_{(j,i)} A[i,j] p_j \\ p_i &= (1-\alpha)p_i^0 + \alpha \sum_{(j,i)} \beta_{t(j,i)} a_{j,i} p_j \\ &= (1-\alpha)p_i^0 + \alpha \sum_{t \in 1,\ldots,T} \sum_{t(j,i)=t} \beta_t a_{j,i} p_j \\ &= (1-\alpha)p_i^0 + \sum_{t \in 1,\ldots,T} \alpha\beta_t \sum_{t(j,i)=t} a_{j,i} p_j \\ &= w_0 f^0(i) + \sum_{t \in 1,\ldots,T} w_t f^t(i) \\ &= \sum_{t \in 0,\ldots,T} w_t f^t(i) \\ &= S(f^0(i), \ldots, f^T(i)) \end{aligned} \tag{5.4}$$

where

$$w_0 = 1-\alpha$$

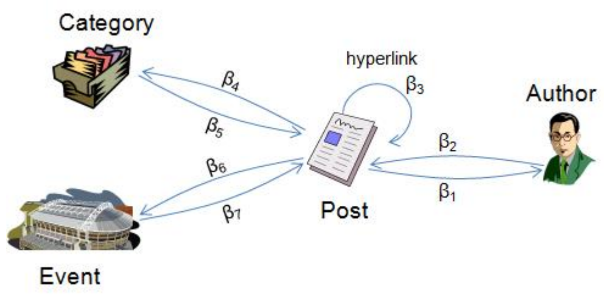


Figure 5.1: Spinn3r Schema Graph

$$\begin{aligned} w_t &= \alpha\beta_t \\ f^0(i) &= p_i^0 \\ f^t(i) &= \sum_{t(i,j)=t} a_{j,i}p_j \end{aligned}$$

From Equation 5.4, the score of each object is then defined as a function of the scores of the adjacent nodes, grouped by type, as follows:

$$p_i = S(f^0(i), ..., f^T(i)) = \sum_{t\in 0,...,T} w_t f^t(i) \tag{5.5}$$

I use the node and edge types of the Spinn3r schema graph to illustrate. Suppose the type of edges in Figure 5.1 are indexed as follows, with weight β_i associated with edge type i:

- $1 : Post \rightarrow Author$
- $2 : Author \rightarrow Post$
- $3 : Post \rightarrow Post$
- $4 : Post \rightarrow Category$
- $5 : Category \rightarrow Post$
- $6 : Post \rightarrow Event$

- $7 : Event \rightarrow Post$

The following expression represents the score of a node of type *Author*, where the first element p_a represents the score of node a, and the following seven elements correspond to the seven edge types for this schema graph:

$$\begin{aligned} p_a &= S(f_a^0, f_a^1, f_a^2, f_a^3, f_a^4, f_a^5, f_a^6, f_a^7) \\ &= S(f_a^0, f_a^1, 0, 0, 0, 0, 0, 0) \\ &= w_0 f_a^0 + w_1 f_a^1 + 0 + 0 + 0 + 0 + 0 + 0 \\ &= (1-\alpha) f_a^0 + \alpha\beta_1 f_a^1 \qquad (5.6) \end{aligned}$$

Thus, f_a^1 is the sum of the incoming scores to a, from any node of type *Post*, along edge type $1 : Post \rightarrow Author$ with weight β_1. Since authors have only one type of incoming link, the rest of the features are always zero for any *Author* node. Similarly, the score of a node d of type *Post* is computed as follows:

$$\begin{aligned} p_d &= S(f_d^0, f_d^1, f_d^2, f_d^3, f_d^4, f_d^5, f_d^6, f_d^7) \\ &= S(f_d^0, 0, f_d^2, f_d^3, 0, f_d^5, 0, f_d^7) \\ &= w_0 f_d^0 + 0 + w_2 f_d^2 + w_3 f_d^3 + 0 + w_5 f_d^5 + 0 + w_7 f_d^7 \\ &= (1-\alpha) f_d^0 + \alpha\beta_2 f_d^2 + \alpha\beta_3 f_d^3 + \alpha\beta_5 f_d^5 + \alpha\beta_7 f_d^7 \qquad (5.7) \end{aligned}$$

where f_d^0 is the personalization value for node d and f_d^2, f_d^3, f_d^5, and f_d^7 are the total incoming scores to d, from any node of types *Author* (along edge type $1 : Author \rightarrow Post$, with weight β_2), *Post*, *Category*, and *Event*.

An expectation maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood value evaluated using the current estimates for the parameters, and a maximization (M) step, which recomputes the values for the parameters while maximizing the expected log-likelihood value from the previous E step. These parameter estimates are then used to determine the distribution of the latent variables in the next E step. Recall that with authority flow, the score of each object depends on the (unobserved) scores of each adjacent object as determined by the type of the adjacent object. Thus, for learning to rank in networks, I take an iterative approach similar to expectation maximization (EM) for learning the model parameters (w_is in Equation 5.5) in my framework.

Let vector W denote w_is and A_W denote the adjacency matrix, which is weighted by W. Let $loss(P, Y)$ denote the loss function that measures the disagreement between score vector P and training data Y. The two steps of our iterative approach are as follows:

1. **Expectation step**:

$$\hat{P} = A_{\hat{W}} \cdot \hat{P}$$

2. **Maximum likelihood estimation step**:

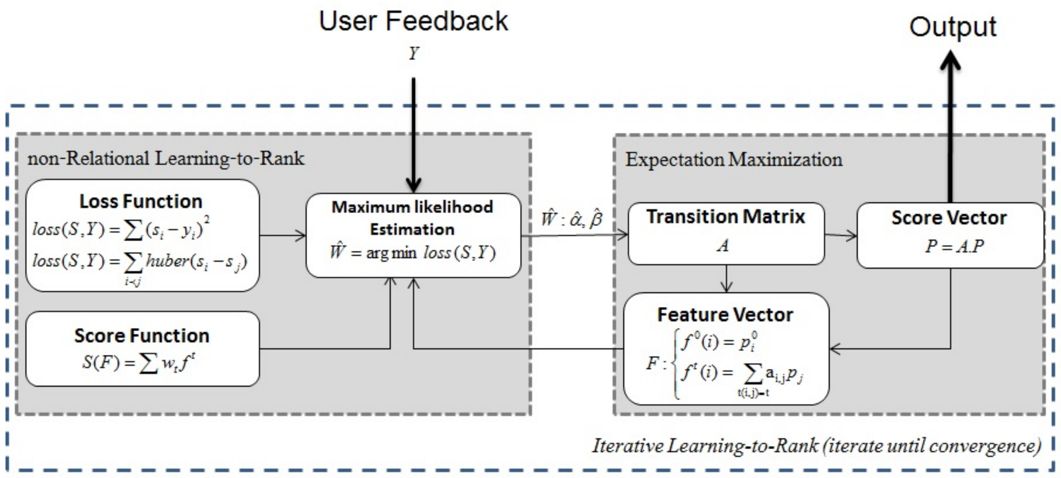
$$\hat{W} = \arg\min\{loss(A_W \cdot \hat{P}, Y)\}$$

based on equations 5.4 and 5.5:

$$\hat{W} = \arg\min\{loss(S_W, Y)\}$$

.

The initial value of $\hat{W}$ is set to equal weights for all edges. The expectation step and the maximum likelihood estimation step will be repeated until convergence. In this learning framework, the learning problem reduces to finding $\arg\min\{loss(A_W \cdot$



User Feedback
Y
Output
non-Relational Learning-to-Rank
Loss Function
$loss(S,Y)=\sum(s_i-y_i)^2$
$loss(S,Y)=\sum_{i<j} huber(s_i-s_j)$
Score Function
$S(F)=\sum w_t f^t$
Maximum likelihood Estimation
$\hat{W} = \arg\min\ loss(S,Y)$
$\hat{W}:\hat{\alpha},\hat{\beta}$
Expectation Maximization
Transition Matrix
A
Score Vector
$P=A.P$
Feature Vector
$F:\begin{cases} f^0(i)=p_i^0 \\ f^t(i)=\sum_{t(i,j)=t} a_{i,j}p_j \end{cases}$
Iterative Learning-to-Rank (iterate until convergence)

Figure 5.2: The workflow of a learning algorithm for ObjectRank

$\hat{P}, Y)\}$ in each iteration. Based on equations 5.4 and 5.5, this is equivalent to finding $\arg\min\{loss(S, Y)\}$, where S is a non-recursive, linear function and Y is the training data. Lets compare the proposed iterative approach with the on-shot learning solution proposed by Chakrabarti et al. [11]. For a given transition matrix A, in practice, the PageRank or ObjectRank score vector is frequently computed via power iteration, by initializing $p = p^0$ and iterating $P = A.P$ until convergence. Chakrabarti et al. expand and truncate the recursion, and suggest $P = A^H.P^0$ for modest values of H (10-50). Therefore, the problem of learning the parameters reduces to optimizing some loss function for $P = A^H.P^0$. They solve this problem for a specific loss function.

The workflows for the proposed iterative model in this chapter and the approach taken by Chakrabarti et al. are shown in Figures 5.2 and 5.3, These figures show the advantages of an iterative model over the one-shot learning algorithm proposed by Chakrabarti et al.:

- First, the score function in the iterative model is independent of the transition matrix. Changing the transition-matrix formula only changes the feature val-

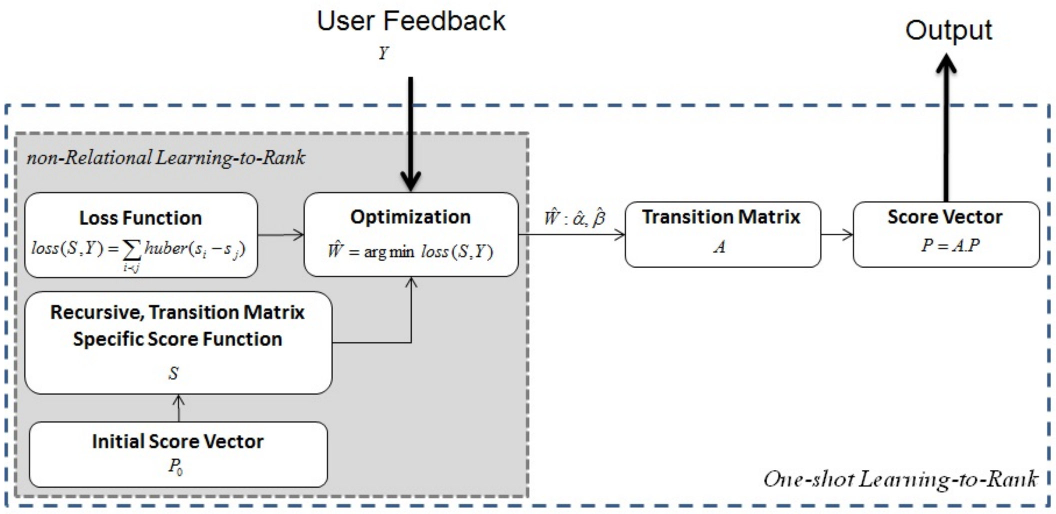

User Feedback
Y
Output
non-Relational Learning-to-Rank
Loss Function
$loss(S,Y)=\sum_{i<j} huber(s_i - s_j)$
Optimization
$\hat{W} = \arg\min \; loss(S,Y)$
$\hat{W}: \hat{\alpha}, \hat{\beta}$
Transition Matrix
A
Score Vector
$P = A.P$
Recursive, Transition Matrix Specific Score Function
S
Initial Score Vector
P_0
One-shot Learning-to-Rank

Figure 5.3: The workflow of a learning algorithm proposed by Chakrabarti et al.

ues. It changes the expectation-maximization step, but does not change the maximum-likelihood-estimation algorithm. In other words, for a different type of transition-matrix formula (in the same class of authority transfer as ObjectRank), the same maximum-likelihood-estimation algorithm can be used. However, in Chakrabarti et al.'s approach, new optimizers must be developed for different formulations of the transition matrix. This is because any change in the transition-matrix formulation will directly change the score-function formulation. Building a new optimizer is not trivial, due to the complex nature of the score function.

- Second, for different domains and applications, different loss functions may be more appropriate. In addition, different types of training data Y need different loss functions. Plugging a new loss function into the iterative model is trivial. The optimizer module is a stand-alone model that optimizes a loss function for a linear score function. The loss function and optimizer in this module can be easily replaced by any off-the-shelf loss function and optimizer in the non-relational learning-to-rank literature without affecting the consistency of the other parts of the framework. In contrast, building a new optimizer for a new loss function would be difficult or expensive using Chakrabarti et al.'s approach.

One limitation of the iterative learning-to-rank framework is that the optimization (edge weight learning) for each node type is performed locally. This framework needs at least a few training samples from all types of nodes to learn all edge weights. Recall feature $f^t(i)$ for node i is as follows:

$$f^t(i) = \sum_{t(i,j)=t} a_{j,i} p_j$$

For learning w_t, $f^t(i)$ must be non-zero and this occurs if and only if edge type t is defined for node i. For example, if edge type t represents the *Paper-to-Author* edge, feature f^t is non-zero for *Author* nodes, but it will be zero for all other types of nodes. To learn the weight w_t for feature f^t, the training data should include at least a few *Author* nodes.

5.2 Loss Function

In each iteration of learning, $\arg\min\{loss(S_W, Y)\}$ has to be calculated. $loss(.)$ maps the difference between the ranking by S_W and training data Y onto a real number, intuitively representing the "error" associated with the ranking. In the following sections, the appropriate loss functions for pointwise and pairwise models will be presented.

5.2.1 Pointwise Learning

In a pointwise learning model, training data contains a set of objects with their absolute ranking scores. In this case, a natural choice of loss function is a squared error function that measures the total squared differences between the predicted scores and the gold-standard scores in training data Y:

$$loss(S, Y) = \sum_i (s(i) - Y(i))^2$$

Since S is a linear function, finding $\arg\min\{\sum_i (s(i) - Y(i))^2\}$ is in fact a least-square linear regression problem.

5.2.2 Pairwise Learning

In this setting, training data include the ranking preference for pairs of training objects. Training data only provides relative orders, unlike the pointwise model, which requires absolute values. It is easier for humans to compare two objects and express their preference rather than providing absolute values. In addition, other studies have collected large amounts of implicit relevance judgments by recording facets of user interactions between the system and the browser during actual search sessions [43]. For example, if a user clicks on the third link before clicking on the first or second links, we infer that the user prefers the third link to the first and second links. However, we may not infer preferences for the links that appear after the third link. This way, we can collect a huge amount of training data that can help improve parameter estimation.

The pairwise model uses a step loss function. Chakrabarti et al.[11] showed that a *Huber* function with window L is a proper step function for pairwise learning:

$$loss(S, Y) = \sum_{i \prec j \in Y} huber(s(i) - s(j)) \tag{5.8}$$

$$huber(z) = \begin{cases} 0 & z \leq 0 \\ z^2/(2L) & y \in (0, L] \\ z - L/2 & y > L. \end{cases} \tag{5.9}$$

5.3 Evaluation

This section demonstrates the performance of the proposed learning framework. Two case studies are conducted: First, real-world networks and synthetic parameters are used to generate synthetic datasets and the performance of the proposed framework using different types of training data is evaluated. Note that the parameters used to generate the training dataset are hidden from learners. Finally, performance of both pointwise and pairwise algorithms for varying amounts of training data and noise is demonstrated.

In the second case study, I test whether the proposed framework can learn FutureRank, which predicts the future PageRank of scientific articles.

5.3.1 Datasets

- **Spinn3r Dataset**: dataset provided by Spinn3r.com, which was used in Chapter 4 to evaluate the performance of the proposed personalization algorithm.
- **arXiv Dataset**: This is a dataset of scientific articles, called the arXiv (hep-th) dataset[1], which was used in Chapter 3 to evaluate the performance of FutureRank.

5.4 Case Study 1: Evaluate the Performance of Learning Variants

In this case study, the performance of the proposed learning framework for different types of training data is evaluated. The performance of both pointwise and pairwise learning models is compared with that of a baseline model. The baseline algorithm is an ObjectRank model with no learning, in which the outgoing edges from each node type have equal weights (βs) and are summed up to 1. Finally, the performance of each model for varying amounts of training data and noise will be demonstrated.

5.4.1 Evaluation

The main steps of the evaluation schema are as follows:

1. Get graph G from a real dataset.
2. Generate training data Y.
    - Assign hidden parameters α and βs and compute ObjectRank using hidden parameters.
    - Draw random objects (for pointwise learning) or random pairs of objects (for pairwise learning) as training objects.
3. Give G and Y to the learning algorithm but not the hidden parameters.
4. The learner estimates α^* and β^*s.
5. Compute the ranking R^* using α^* and β^*s.
6. Compare the top K results of R^* with the top K results of ranking R, which is computed using the hidden parameters.

Figure 5.4 shows the accuracy of the baseline algorithm, as well as the pointwise and pairwise models, for the arXiv and Spinn3r datasets. The X axis shows the size of the training data, and the Y axis shows the F1 score for the top 50 results for each algorithm. Since the baseline algorithm does not perform any learning, its performance is constant and independent of the size of the training data. In both real datasets with synthetic parameters, both pointwise and pairwise models successfully learn the parameters with as few as 10 training data samples (per node type). However, pointwise learning outperforms pairwise learning for smaller amounts of

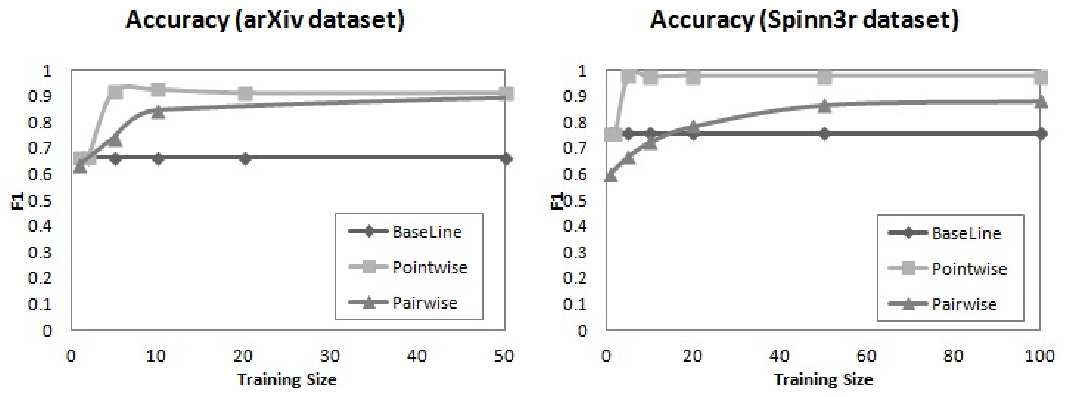


Figure 5.4: Accuracy of pointwise learning and pairwise learning. The X axis shows the size of the training data per node type and the Y axis shows the F1 for the top 50 results.

training data. Figure 5.5 shows the accuracy of the same three algorithms after each iteration of learning. The X axis shows the iteration number, and the Y axis shows the F1 score for the top 50 results for each algorithm. For each dataset, both pointwise and pairwise algorithms converge after a few iterations. Again, since the baseline algorithm does not perform any learning, its performance does not change.

In this case study, training data are generated from real graphs with synthetic parameters that are hidden from the learning algorithms. There is also an assumption that there will be no noise or inconsistency in the training data. However, in a real-world situation, users may provide inconsistent feedback/training data. In addition, many sources of training data are implicit data that are automatically extracted from the user's interaction with the system. For such training data, there is the potential to miss-interpret the user's actions and collect training data that are not perfect. Due to the existence of such sources of noise, I also evaluate the performance of both pointwise and pairwise algorithms against noisy data. x percent noise is added to the training data as follows: For each object o in the training dataset,

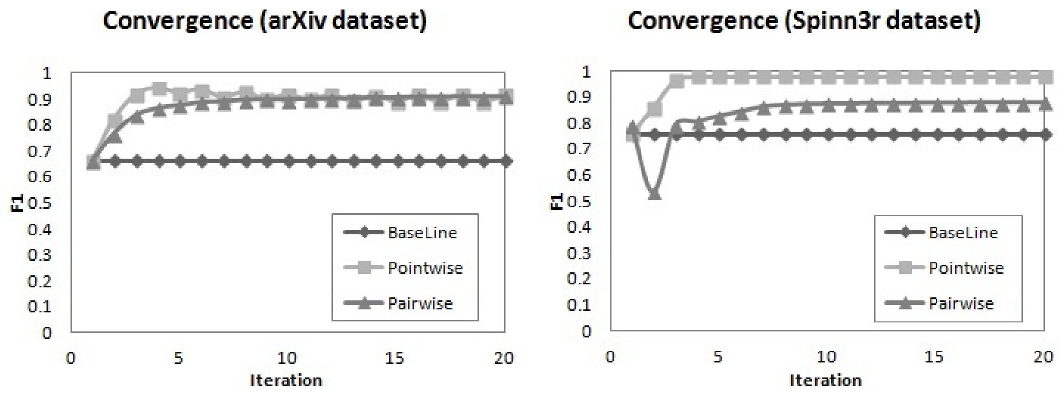


Figure 5.5: Convergence of pointwise learning and pairwise learning. The X axis shows the number of learning iterations and the Y axis shows the F1 for the top 50 results.

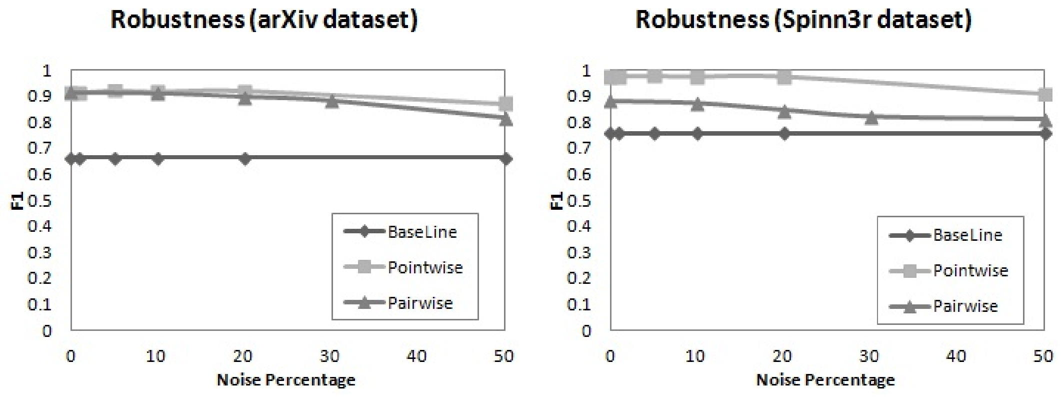


Figure 5.6: Performance of pointwise learning and pairwise learning with up to 50% noise in the training data. The X axis shows the percentage of noise and the Y axis shows the F1 value for the top 50 results.

score s_o is replaced with a random number drawn with a uniform distribution from:

$$[(1-x)*s_o,(1+x)*s_o]$$

Figure 5.6 shows the performance of both algorithms, as well as the baseline algorithm for the noisy data. Again, since the baseline algorithm does not perform any learning, its performance is constant and independent of the amount of noise in the data. The performance of the pointwise model drops slightly. Similarly, the pairwise model also shows a slight drop.

In these experiments, the assumption was that there is no missing or noisy information to distort the network structure and only the training data may be noisy. Real-world datasets are often incomplete, which results in missing nodes and missing edges in networks. As discussed in Chapter 2, there are many different ways to handle the missing data in non-network domains [33, 69]. Missing data regarding the network structure can results in the following two scenarios:

- **Uniformly distributed missing information:** In this scenario, the missing nodes and edges are uniformly distributed. Recall the ObjectRank formulation. The features f^t for each object is the sum of all incoming authority flows from edges of type t. If nodes and edges are uniformly missing, feature f^t will exist. Learning w_t for f^t will have to accommodate noisy values because of the missing edges and neighbor nodes of type t. Hence, this scenario will resemble the problem of noisy data (feature level) in non-network datasets. The best solution for such a scenario is to use link prediction algorithms to predict the missing edges [3]. The advantage of link prediction algorithms over the non-network imputation methods such as regression imputation or multiple imputation [33, 69] is that link prediction models consider both non-relational features and the network structure to predict the missing edges.

- **Skewed missing information:** In this scenario, (a subset of) particular nodes may be missing all edges of a particular type. For example, in the bibliographic network (arXiv), *Author* edges may be missing for a subset of papers. Similar to the previous scenario, such missing edges can be predicted using the link prediction algorithms [3]. However, if the *Author* edge is missing for most papers, then this is equivalent to missing a particular feature (for the most of the records) in a non-network dataset. There is a little benefit in imputing or predicting such values or such edges. The common practice is to ignore these features/edge types entirely.

5.5 Case Study 2: Learning the Parameters of FutureRank

In the second case study, I evaluate the performance of the proposed framework for learning the parameters of FutureRank. Note that there is no set of parameters that can perfectly predict the number of future citations or future PageRank. The best results obtained by exhaustive searches had a F1 value of 52% for the top 50 results and Spearman's rank correlation of 58. Hence, in this case study, obtaining the same performance as an exhaustive search approach is considered as learning perfectly.

I will first show how to implement FutureRank by ObjectRank. Let α denote the weight of the citation links and β denote the weights of the links from authors to papers. The weight of the links from papers to authors is 1. γ represents $(1-\alpha)$ in ObjectRank notation, and R^{Time} is the *personalization* vector:

$$A_{j,i} = \begin{cases} \frac{1}{outdegree(j,\text{“}Paper''\text{)}} & \text{if } i \text{ is the author of } j; \\ \frac{\alpha}{outdegree(i,\text{“}Author''\text{)}} & \text{if } j \text{ is the author of } i; \\ \frac{\beta}{outdegree(i,\text{“}Paper''\text{)}} & \text{if } i \text{ is citing } j; \\ 0 & \text{otherwise;} \end{cases}$$

$$P = \gamma R^{Time} + (1 - \gamma)A.P$$

Now, let us use the proposed learning algorithm to learn the FutureRank. I will compare the performance of the learned model with the original FutureRank model, which was obtained by an exhaustive search on the parameter space.

5.5.1 Evaluation

Here are the main steps of the evaluation schema:

1. Get graph G from the arXiv dataset.
2. Generate training data Y:
   - Split the dataset into two partitions: The first partition, historical data, contains all papers published before 2001 and the second partition, future data, contains papers published in or after 2001.
   - Run PageRank on the second partition to compute the future PageRank score for each object.
   - Draw a set of random objects as training data.
3. Give G and Y to the learner.
4. The learner estimates α^* and β^*s.
5. Compute ObjectRank using α^* and β^*s.
6. Evaluate the top K results of the ObjectRank using α^* and β^* and compare them with the top K results of Future PageRank.

Figure 5.7 shows the accuracy of an exhaustive search, as well as the pointwise algorithm for learning FutureRank. It shows that the learning algorithm successfully

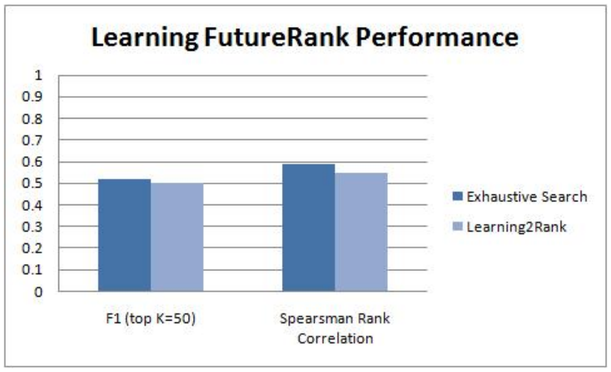


Figure 5.7: Performance for learning FutureRank in comparison with the exhaustive search algorithm.

finds the best parameters (which were found by an exhaustive search). Both the F1 measure and Spearman's correlation obtained by the learner are similar to the F1 measure and Spearman's correlation obtained by an exhaustive search approach.

Chapter 6

Conclusion

In this dissertation, I have addressed some of the challenges of ranking in heterogeneous networks.

- **Ranking in Evolving Networks:** In Chapter 3, I introduced a new measure for ranking scientific articles that I referred to as the future PageRank score. The future PageRank score is the PageRank score computed based only on the citations that will be accrued in the future. I then presented FutureRank, a prediction algorithm for predicting the future PageRank score from the historical network structure. I compared FutureRank with existing approaches and showed that FutureRank is more accurate than traditional models such as citation count, PageRank, or CiteRank [68] for finding and ranking publications.

  FutureRank addressed the specific problem of predicting the future PageRank in a citation network. Studying prediction models in other networks is a possible direction of future work. For example, predicting the number of retweets that a tweet will obtain in the future is a an interesting problem to look at.

- **Personalized Ranking:** In Chapter 4, I presented a personalized random-walk ranking algorithm for recommendations in social media. The personalization algorithm was then evaluated through an experiment with real users, as well as an extensive study of a range of synthetic users. The results showed that my personalized recommendations outperforms baseline approaches.

  A possible extension to this work is to apply a similar algorithm to other domains such as movie recommendations. Movie rating datasets can also be enriched with metadata to build a network:

1. Create one node per movie, actor, director, and genre.
2. Create metadata links between each movie and its actors, directors, and genres. An actor link can be weighted by the role (first actor, second actor, etc).
3. Finally, add a (similarity) link between two movies weighted by similarity measures by algorithms such as collaborative filtering.

An interesting question would then be how the performance of the personalized network ranking algorithm on such graphs compares to a non-network recommendation algorithm.

- **Learning to Rank in Heterogeneous Networks:** In Chapter 5, I presented a framework for learning to rank in networks. The goal was to build an effective solution for estimating the parameters of an authority flow-based model. A pointwise and a pairwise learning algorithm were presented under the proposed framework. The experimental results demonstrated that the learning framework successfully learns the parameters of FutureRank as well as the personalized parameters for personalized ranking. In addition, the results show that both pointwise and pairwise models can successfully learn from as few as 10 training samples (per node type). Both models are shown to be very robust against artificially-introduced noise in the dataset. The main advantage of my framework is that it allows learning-to-rank algorithms from a non-relational domain to be utilized for learning-to-rank in heterogeneous networks. Furthermore, it was shown to converge after a few iterations with high accuracy.

Possible extensions to this work could be an analogy of the behavior of other learning-to-rank algorithms and loss functions in networks. For example, listwise models [10, 70] have been shown to have a superior performance compared to pointwise and pairwise models. Testing the proposed framework on more real-world applications and networks is another direction for future work.

As mentioned in Chapter 5, real-world datasets are often incomplete, which could result in missing nodes and missing edges in the networks. While we addressed the problem of noisy training data in Chapter 5, the problem of incomplete network structure is an open question. Future work would be to study the effect of missing edges and nodes on the performance of the FutureRank, the personalized ranking algorithm, and the learning-to-rank framework.

Bibliography

[1] www.cs.cornell.edu/projects/kddcup/datasets.html.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR conference on Research and development in information retrieval*, 1998.

[3] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[4] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB '04: Proceedings of the 13th international conference on Very Large Data Bases - Volume 30*, pages 564–575, 2004.

[5] Matthias Bender, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane Xavier Parreira, Ralf Schenkel, and Gerhard Weikum. Exploiting social relations for query expansion and result ranking. In *Proceedings of the 24th International Conference on Data Engineering Workshops*, pages 501–506, 2008.

[6] Klaus Berberich, Srikanta Bedathur, Michalis Vazirgiannis, and Gerhard Weikum. Buzzrank . . . and the trend is your friend. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 937–938, New York, NY, USA, 2006. ACM.

[7] J. Bollen, M.A. Rodriquez, and H. Van de Sompel. Journal status. *Scientometrics*, 69(3):669–687, 2006.

[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[9] K. Burton, A. Java, and I. Soboroff. The icwsm 2009 spinn3r dataset. In *Proceedings of the Conference on Weblogs and Social Media (ICWSM 2009)*, 2009.

[10] Zhe Cao and Tie yan Liu. Learning to rank: From pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[11] Soumen Chakrabarti and Alekh Agarwal. Learning parameters in entity relationship graphs from ranking preferences. In *In ECML/PKDD, volume 4213 of LNCS*, pages 91–102, 2006.

[12] P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with Google. *Journal of Informetrics 1, 2007.*

[13] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1:5–32, 1999.

[14] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 169–176, New York, NY, USA, 2007. ACM.

[15] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. Ranking the web frontier. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 309–318, New York, NY, USA, 2004. ACM.

[16] Tamer Elsayed, Jimmy Lin, and Douglas W. Oard. Pairwise document similarity in large collections with mapreduce. pages 265–268, 2008.

[17] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[18] Jianfeng Gao, Haoliang Qi, Xinsong Xia, and Jian yun Nie. Linear discriminant model for information retrieval. In *In Proceedings of the 28th international ACM SIGIR conference*, pages 290–297. ACM Press, 2005.

[19] Eugene Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178:471–479, 1972.

[20] Eugene Garfield. How to use citation analysis for faculty evaluations and when is it relevant?(part 1). *Current Contents*, pages 5–13, 1983.

[21] W. Glnzel. The need for standards in bibliometric research and technology. *Scientometrics*, 35:167–176, 1996.

[22] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB '04: Proceedings of the Thirtieth international conference on Very Large Data Bases*, pages 576–587, 2004.

[23] Taher H. Haveliwala. Topic-sensitive pagerank. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.

[24] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *ICANN '99: Proceeding of the 9th international conference on Artificial Neural Networks*, pages 97–102, 1999.

[25] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Can social bookmarking improve web search? In *WSDM '08: Proceedings of the international conference on Web Search and Web Data Mining*, 2008.

[26] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications*, pages 411–426, 2006.

[27] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.

[28] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[29] M. Koenig. Determinants of expert judgement of research performance. *Scientometrics*, 4:361–378, 1982.

[30] Michael E. D. Koenig. Bibliometric indicators versus expert opinion in assessing research performance. *Journal of the American Society for Information Science*, 34:136–145, 1983.

[31] RN Kostoff. Performance measures for government-sponsored research: Overview and background. *Scientometrics, v36, 281-292.*

[32] Apostolos Kritikopoulos and Martha Sideri. The compass filter: Search engine result personalization using web communities. In Bamshad Mobasher and Sarabjot S. Anand, editors, *ITWP*, Lecture Notes in Computer Science, pages 229–240, 2003.

[33] Kamakshi Lakshminarayan, Steven A. Harp, Robert P. Goldman, and Tariq Samad. Imputation of missing data using machine learning techniques. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *KDD*, pages 140–145. AAAI Press, 1996.

[34] Amy Langville and Carl Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[35] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67, October 2010.

[36] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.

[37] Lawani and Bayer. Validity of citation criteria for assessing the influence of scientific publications: New evidence with peer assessment. *Journal of the American Society for Information Science*, 34:59–66, 1983.

[38] Sune Lehmann, Andrew D. Jackson, and Benny E. Lautrup. Measures and mismeasures of scientific quality. 2005.

[39] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.

[40] Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. A probabilistic model for retrospective news event detection. In *SIGIR '05*, pages 106–113, New York, NY, USA, 2005. ACM.

[41] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. 2009.

[42] Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. Co-authorship networks in the digital library research community. *Inf. Process. Manage.*, 41(6):1462–1480, 2005.

[43] Yiqun Liu and Yupeng Fu. Automatic search engine performance evaluation with click-through data analysis. In *In Proceedings of the 16th international conference on World Wide Web*, 2007.

[44] Lucene. http://lucene.apache.org/java/docs/.

[45] Nan Ma, Jiancheng Guan, and Yi Zhao. Bringing pagerank to the citation analysis. *Inf. Process. Manage.*, 44(2):800–810, 2008.

[46] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval.* Cambridge University Press, New York, 2008.

[47] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *EMNLP '04: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

[48] Einat Minkov and William W. Cohen. Learning graph walk based similarity measures for parsed text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 907–916, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[49] Einat Minkov, William W. Cohen, and Andrew Y. Ng. Contextual search and name disambiguation in email using graphs. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 27–34, New York, NY, USA, 2006. ACM.

[50] Gilad Mishne and Maarten de Rijke. A study of blog search. *Lecture Notes in Computer Science*, 3936/2006, 2006.

[51] Julie L. Morrison, Rainer Breitling, Desmond J. Higham, and David R. Gilbert. Generank: Using search engine technology for the analysis of microarray experiments. *BMC Bioinformatics*, 6:233, 2005.

[52] Ramesh Nallapati. Discriminative models for information retrieval. In *SIGIR '04: Proceedings of the 27th international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, New York, NY, USA, 2004. ACM.

[53] Narin. Evaluative bibliometrics: The use of publication and citation analysis in the evaluation of scientific activity. *Computer Horizons*, 1976.

[54] Narin and Hamilton. Bibliometric performance measures. *Scientometrics*, 36, 1996.

[55] Blair Neate, Warwick Irwin, and Neville Churcher. Coderank: A new family of software metrics. In *ASWEC '06: Proceedings of the Australian Software Engineering Conference*, pages 369–378, Washington, DC, USA, 2006. IEEE Computer Society.

[56] Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. Object-level ranking: bringing order to web objects. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 567–574, New York, NY, USA, 2005. ACM.

[57] 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.

[58] Francis Pinski, Gabriel; Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, pages 297–312, 1976.

[59] Mathew Richardson and Pedro Domingos. *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[60] Hassan Sayyadi and Lise Getoor. Future rank: Ranking scientific articles by predicting their future pagerank. In *SDM '09: Proceeding of the SIAM International Conference on Data Mining*, April 2009.

[61] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *Proceedings of International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[62] Hassan Sayyadi and Louiqa Raschid. A graph analytical approach for topic detection. *ACM Transactions on Internet Technology*, 2014.

[63] M. Spiliopoulou and L. C. Faulstich. WUM: A tool for Web utilization analysis. *Lecture Notes in Computer Science*, 1590:184–203, 1999.

[64] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW '04*, pages 675–684, 2004.

[65] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, 2005.

[66] Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: a ranking method with fidelity loss. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, New York, NY, USA, 2007. ACM.

[67] Ramakrishna Varadarajan, Vagelis Hristidis, and Louiqa Raschid. Explaining and reformulating authority flow queries. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 883–892, Washington, DC, USA, 2008. IEEE Computer Society.

[68] Dylan Walker, Huafeng Xie, Koon-Kiu Yan, and Sergei Maslov. Ranking scientific publications using a simple model of network traffic. *CoRR*, abs/physics/0612122, 2006.

[69] Jeffrey C. Wayman. Multiple Imputation For Missing Data: What Is It And How Can I Use It?, 2003.

[70] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *ICML '08: Proceedings of the 25th International Conference on Machine learning*, pages 1192–1199, New York, NY, USA, 2008. ACM.

[71] Y. Yang, T. Pierce, and J. G. Carbonell. A study on retrospective and on-line event detection. In *SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR conference on Research and development in information retrieval*, 1998.

[72] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis G. Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by document. In *WSDM '09: Proceedings of the international conference on Web Search and Web Data Mining*, pages 34–43, 2009.

[73] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294, New York, NY, USA, 2007. ACM.

[74] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis,

editors, *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, Cambridge, MA, 2008.

[75] Ding Zhou, Sergey A. Orshanskiy, Hongyuan Zha, and C. Lee Giles. Co-ranking authors and documents in a heterogeneous network. *ICDM '07: Proceeding of the 7th IEEE International Conference on Data Mining*, 2007.