Recommendation in Scholarly Big Data

Yuning Mao
Shanghai Jiao Tong University
Dongchuan Road 800
Shanghai, China
morningmoni@sjtu.edu.cn

ABSTRACT

While recommendation has been researched for long, recommendation in scholarly big data is not well defined yet. In this project, we first implement a recommender system in a real academic search system, and then propose a new algorithm *ZeroRank* for judging the heat of newly published literatures without citations so as to recommend potentially hot papers to researchers.

In the first part, we will introduce the recommender system we implement. To address the problem of huge sparsity and cold start, we use a hybrid recommendation method *Collaborative Modeling Regression (CTR)* [15] which leverages both content information and user preferences. We use *Microsoft Academic Graph(MAG)* as data source, and also do experiments using a dataset from *citeulike*. The process of work includes data selection, data processing, model training and adjusting.

In the second part, we define zero citation ranking problem and propose ZeroRank. ZeroRank is an algorithm combining random walk on heterogeneous network and learning to rank framework. We use random walk as a feature extractor to obtain the "author", "venue", "affiliation" features of each paper, and then apply learning to rank method to train a ranking model. We conduct our experiments on *MAG*, and the results show that our algorithm achieves at least 17.6% improvement in NDCG score compared with the state-of-the-art literature ranking and citation prediction algorithms. We experiment on 31 subfields of computer science and observe a different finding from previous work that "author" is a dominant feature for a paper to gain future citations compared with "venue" and "affiliation".

1. INTRODUCTION

While recommendation has been researched for long, recommendation in scholarly big data is not well defined yet. According to [3], over 80 approaches for academic literature recommendation exist today. Of the approaches proposed, 21% were not evaluated. Among the evaluated approaches, 19% were not evaluated against a baseline. Of the user studies performed, 60% had 15 or fewer participants or did not report on the number of participants. Information on runtime and coverage was rarely provided. Due to the fact mentioned above, it's really hard to find a suitable recommendation method for scholarly big data. We happen to be sort of familiar with *Latent dirichlet allocation (LDA)* [4], this leads us to a method called *Collaborative Modeling Regression (CTR)* [15]. It is a hybrid recommendation approach which combines LDA and collaborative filtering. It uses both content information and user preferences. This paper is awarded "Best student paper at KDD 2011" and the proposed method has been used by *New York Times* for their recommendations. Since LDA is suitable for text analysis, the number of users in the academic search system *acemap* is small, and also CTR is very successful in commercial systems, we eventually decided to use this method for the recommendation of *acemap*.

Ranking scientific literatures is helpful for researchers to find high quality papers, potentially promising research directions, and also plays an important role in academic reward system.

Traditional methods use the citation count as a metric. Yet they are too "democratic" in treating all citations as equal and ignoring differences in importance of citing papers [14]. With *PageRank* [9] and *HITS* [7], many graph based ranking methods were proposed to model the citation network as website network in order to measure the prestige of each publication. Nevertheless, the dynamic and evolving nature makes citation network different from *WWW*, because newly published papers are only able to cite earlier published ones. As a result, methods that do not consider this nature are likely to give bias to old papers.

Many efforts have been made to address this issue. Walker *et al.* [14] proposed *CiteRank* to leverage the publication time information by modifying *PageRank* with an exponential initial distribution. To utilize more information such as authors, Sayyadi and Getoor [11] presented the model *FutureRank*, involving a random walk on citation network and author-paper network. Wang *et al.* [17] defined a ranking algorithm integrating citations, authors, venues and publication time information. Wang *et al.* [16] also designed *MRFRank* employing text-feature modeling innovativeness of a paper.

While the work mentioned above leverages different information, these methods are all designed to rank papers of the whole network, and therefore tend to neglect the ranking result of the latest papers without citation information, since these papers are only a small portion of the dataset which makes little contribution to the performance measure function. Meanwhile, they lie on the edge of the citation network and have no indegree, making it hard for *PageRank* based methods to perform ranking.

However, we find that ranking such papers could benefit researchers in the community. Consider the situation: after *CIKM 2016* is held, hundreds of papers are published in the proceedings. It is a natural question to ask, which one will receive the highest citation after 5-10 years among those pa-

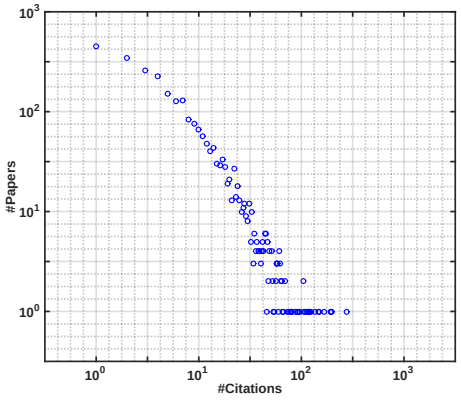


Figure 1: The 5-year citation numbers of 2552 *CIKM* papers published from 1992 to 2011

pers. Answering this question enables providing researchers with potential hot papers and topics. Admittedly, scientific articles are not born equal [12]. We analyze 2552 papers[1] published in *CIKM* from 1992 to 2011. Figure 1 shows the number of citations after 5 years, which presents a power distribution. The huge difference between citation numbers gives us a motivation to raise a new question *zero citation ranking*, *i.e.*, ranking newly published papers without citations.

In this work we choose "author", "venue", "affiliation" as features to characterize a paper's potential of being popular in the future, because there is a strong relation between these features and future citations. However, two challenges for such a scheme are: how to define these three features? And how to quantitatively measure their different contributions to highly cited papers?

To acquire the features, we leverage citations, authors, venues, affiliations and time information, and design a reinforced random walk as a feature extractor. Unlike previous random walk algorithms, we aim to obtain potential of being popular instead of prestige or centrality, thus we apply an average function each iteration. For example, an author's score is based on the average score of all the papers he/she writes. To make our algorithm handle huge network efficiently, we design a parallel random walk algorithm and implement it on *Spark*.

To differentiate the weights, we use learning to rank technology instead of previous methods such as linear regression. Learning to rank enables us to use training set containing many time slices by shifting the current time point, where linear regression can not apply. It can also directly optimize the measure function we select to further refine the results obtained by random walk.

We conduct our experiments on *Microsoft Academic Graph* containing more than 100 million papers. And the results show that our algorithm achieves at least 17.6% improvement in NDCG score comparing *ZeroRank* with the state-of-the-art ranking and citation prediction methods. Furthermore, we run *ZeroRank* separately on 31 subfields of computer science. Unlike previous result [5] in which "venue" plays a major role in future citation number, our experiment shows that "author" is a dominant feature for future citation.

[1]The data is from *Microsoft Academic Graph [13]*

2. COLLABORATIVE TOPIC MODELING FOR RECOMMENDATION

The flow of process is as follows. First we get datasets from different sources; Then we do data selection and data processing; Next we run *LDA* using the processed data, and finally feed the results of *LDA* into the model *CTR*. In this way, we get the feature vectors of users and items and can do recommendation after matrix multiplication.

2.1 Data Selection

We use datasets from two sources. The first one is from citeulike. At CiteUlike, registered users create personal reference libraries; each article usually has a title and abstract. However, in the open dataset, there are only IDs of papers, we hence decided to crawl the title and abstract of each paper. After crawling about twenty percent of the papers, we were banned by citeulike so we turned to the dataset which is shared by [15]. Though the size of this dataset is relatively small, it's well processed and more convenient since we need to make contrast with the results in [15]. According to [15], they merged duplicated articles, removed empty articles, and removed users with fewer than 10 articles (note that in *Acemap* all current users are with fewer than 10 articles) to obtain a data set of 5, 551 users and 16, 980 articles with 204, 986 observed user-item pairs. (This matrix has a sparsity of 99.8%;) On average, each user has 37 articles in the library, ranging from 10 to 403. 93% of the users have fewer than 100 articles. For each article, they concatenate its title and abstract. They remove stop words and use tf-idf to choose the top 8, 000 distinct words as the vocabulary. This yielded a corpus of 1.6M words. These articles were added to CiteULike between 2004 and 2010. On average, each article appears in 12 users' libraries, ranging from 1 to 321. 97% of the articles appear in fewer than 40 libraries.

Another dataset is being used in the system *Acemap*. It comes from *Microsoft Academic Graph(MAG)*[2] provided by *Microsoft*. The Microsoft Academic Graph is a heterogeneous graph containing scientific publication records, citation relationships between those publications, as well as authors, institutions, journals and conference "venues" and fields of study. There are currently more than 120,000,000 papers in the dataset.

Besides, we crawled abstracts of papers of *IEEE*, *ACM*, and *Springer* (this work was done by other students) according to the URL given in *MAG*. In total, we now have papers with abstract about 6,900,000. In contrast, in the academic search system *Acemap*, currently there are 256 users and 50 user-item pairs, which is even sparser than the situation in *citeulike*. This makes recommendation much harder to perform than that in *citeulike*. To get the raw data from database, we use *mysql.connector* to traverse the database containing abstracts of papers and fetch *Title, Publish Year, Abstract, DOI* as raw data. Then we use DOI information to get PaperID, the identifier of papers in *MAG* so that we can acquire other auxiliary information such as *Field of Study (FoS)*. To ensure the quality and freshness of recommended papers, we select two subsets of papers. The first one is those published after 2000 and has gained citations more than 10, and the second one is those published after 2005 and has gained citations more than 50. There are about 390,000 and 28,000 papers in each subset, respectively.

[2]http://research.microsoft.com/en-us/projects/mag/

2.2 Data Processing

We follow similar rules as in [15] to do data processing. The tool we use is *NLTK*. Different from what they did in [15], we concatenate paper's title times three and abstract because we believe that title should have more weight and it's usually short. After trying combination of different text processing approaches, including removing the stop words and digits, recovering original word form of each word. judging whether a word is English or not and so on, we found that the following combination performs best. That is, use *RegexpTokenizer* to split sentences into words and remove punctuation. Use stop word list to remove the stop words and *isdigit()* to remove digits. Use *SnowballStemmer* to judge whether a word can be converted to unicode but don't use it to recover the original word form. Also we don't extract non-English words.

After processing raw data, we get a vocabulary which contains around 340,000 words, which is so large that many of them are even unrecognizable. Therefore, we use calculate *tf-idf* score of each word to select the most important and distinguishable ones. There is a package named *TextCollection* in *NLTK* which can be used to calculate *tf-idf*. However, the efficiency of it is extremely low since it can only calculate for one word a time. We hence calculate the scores manually. After observing the words in a descending order with regard to their *tf-idf* scores, we choose top 7000 words as the final vocabulary. This step is of great importance because there are too many meaningless words in the original vocabulary. A subset of vocabulary before and after extracting can be seen at following figure.

1	srm2975	1	patient
2	foxq1	2	cell
3	thellungiella	3	model
4	mdba	4	system
5	lagotricha	5	network
6	mdbc	6	control
7	cd120a	7	base
8	resultsdespit	8	effect
9	woodi	9	use
10	spiderl	10	imag
11	mdbi	11	studi
12	backgroundeccentr	12	gene
13	dothidea	13	activ
14	fsom	14	cancer
15	8mm2	15	method

Figure 2: The original vocabulary on the left; The top 15 words with highest tf-idf score on the right.

After vocabulary selection, we count word frequency and convert it to the input format of *LDA*. As for LDA, we firstly used the original version of LDA from its original author [3] and then used a faster version called *LightLDA* from Microsoft research [21] [4]. After getting the *doc-word* distribution of each document, CTR can eventually be executed.

2.3 CTR Model

The model we use is collaborative topic regression (CTR) model from [15]. CTR combines traditional traditional collaborative filtering with topic modeling. In this subsection, we will first show you why we use CTR. Then, we will review matrix factorization(a collaborative filtering method) and LDA which are the two basic components of CTR model. At last, we will demonstrate the CTR model.

[3]https://github.com/blei-lab/lda-c

[4]http://research.microsoft.com/en-us/groups/ai/lightlda.aspx

2.3.1 Why CTR?

CTR model matches our acemap academic search engine well.

Acemap is a new project and only a small number of users are active. The information we get from users is limited. Traditional collaborative filtering methods can not deal with completely unrated items. However, most of the papers in acemap are unrated. CTR is born to solve this so called c̈old startp̈roblem since CTR leverages the text information of papers. In acemap, we have the text information of the papers such as abstracts and titles.

CTR is helpful in finding the topic level similarity between users and papers. It not only recommends papers that similar users like, but also ones that have the similar topics as the liked papers. Professor Wang, the leader of acemap, always says that if our system can help researchers to find papers that are not in their field but have the same idea or use the same method, then we are successful. We believe this is a good try.

2.3.2 Matrix Factorization

Matrix factorization is one of the most famous and well-used methods for recommendation. In matrix factorization, we represent users and items in a shared latent low-dimensional space of dimension K. User i is represented by a latent vector $u_i \in R^K$ and item j by a latent vector $v_j \in R^K$. We form the prediction of whether user i will like item j with the inner product between their latent representations,

$$\hat{r}_{ij} = u_i^T v_j \tag{1}$$

To use matrix factorization, we must compute the latent representations of the users and items given an observed matrix of ratings. The common approach is to minimize the regularized squared error loss with respect to U and V.

$$min_{U,V} \sum_{i,j} (r_{i,j} - u_i^T v_j)^2 + \lambda_u ||u_i||^2 + \lambda_v ||v_j||^2, \tag{2}$$

where λ_u and λ_v are regularization parameters.

This matrix factorization for collaborative filtering can be generalized as a probabilistic model cite18. In probabilistic matrix factorization, we assume the following generative process,

1. For each user i, draw user latent vector $u_i \sim N(0, \lambda_u^{-1} I_K)$
2. For each item j, draw item latent vector $v_j \sim N(0, \lambda_v^{-1} I_K)$
3. For each user-item pair(i,j), draw the response

$$r_{ij} \sim N(u_i^T v_j, c_{ij}^{-1}) \tag{3}$$

where c_{ij} is the precision parameter for r_{ij}.

There are two main disadvantages to matrix factorization for recommendation. First, the learned latent space is not easy to interpret; second, as mentioned, matrix factorization only uses information from other users, it cannot generalize to completely unrated items.

2.3.3 LDA

Topic modeling algorithms are used to discover a set of ẗopicsfrom a large collection of documents, where a topic is a distribution over terms that is biased around those associated under a single theme. Topic models provide an interpretable low-dimensional representation of the documents.

latent Dirichlet allocation(LDA) is the simplest topic model. Assume there are K topics $\beta = \beta_{1:k}$, each of which is a distribution over a fixed vocabulary. The generative process of LDA is as follows.

1. Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$
2. For each word w_{jn}
   (a) Draw topic assignment $w_{jn} \sim Mult(\theta)$
   (b) Draw word $w_{jn} \sim Mult(\beta_{z_{jn}})$

This process reveals how the words of each document are assumed to come from a mixture of topics: the topic proportions are document-specific, but the set of topics is shared by the corpus. Our goal is to use topic modeling to give a content based representation of items in a recommender system .

2.3.4 Collaborative Topic Regression(CTR)

CTR represents users with topic interests and assumes that documents are generated by a topic model CTR additionally includes a latent variable ϵ that offsets the topic proportions θ_j when modeling the user ratings. As more users rate articles, we have a better idea of what this offset is. The generative process of CTR is as follows,

1. For each user i, draw user latent vector $u_i \sim N(0, \lambda_u^{-1} I_K)$
2. For each item j,
   (a) Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$
   (b) Draw item latent offset $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$ and set the item latent vector as $v_j = \theta_j + \epsilon_j$
   (c) For each word w_{jn}
      i. Draw topic assignment $w_{jn} \sim Mult(\theta)$
      ii. Draw word $w_{jn} \sim Mult(\beta_{z_{jn}})$
3. For each user-item pair (i, j), draw the rating

$$r_{ij} \sim N(u_i^T v_j, c_{ij}^{-1}) \quad (4)$$

For more detailed information of *CTR*, please refer to [15].

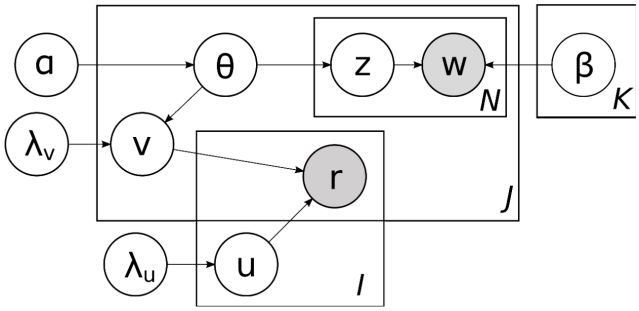


Figure 3: The graphical model for the CTR model.

2.4 Experiments

We did experiments based on both *citeulike* dataset and MAG.

For citeulike dataset, We calculate the recall rates of different user sizes to measure the performance of *CTR* in different development periods of a system, i.e., when the system is just online and has few users, when the system has been running for a while and has a few users, and when the system has been running for a long time and has many users. From figure We can see that the recall increases drastically when the number of users in the system increases from 33 (selected from the original dataset where all users have only fewer than 10 articles in their libraries) to 100 (the first 100 random users selected from the original dataset). Surprisingly, the recall of 100 users is similar or even better than that of the whole dataset, where user number is 5551. This indicates CTR addresses cold start problem very well and hence should perform well for new system like *Acemap*.

We also try to leverage the information of Field of Study (FoS) in the MAG dataset to acquire better results. In MAG, there is a four-level hierarchical structure of FoS. Each article may or may not attach some FoSes. We simulate users according to the FoS information. More specifically, we simulate users in the following manner: for each article with FoS information F, simulate a user who only favors these FoSes and hence favorites papers containing FoS F', where $F \cap F' \neq \varnothing$. However, the results are not satisfactory. We ran the model with 33 users, 100 users, and the total 5551 users, along with simulated ones, and found that the results didn't get improved and even got worse at times. We then analyzed the number of FoSes of papers real users favorite. It showed that these numbers vary a lot, ranging from 0 to 200 (This can be seen in figure). Maybe that's why we shouldn't simulate virtual users in the above way.

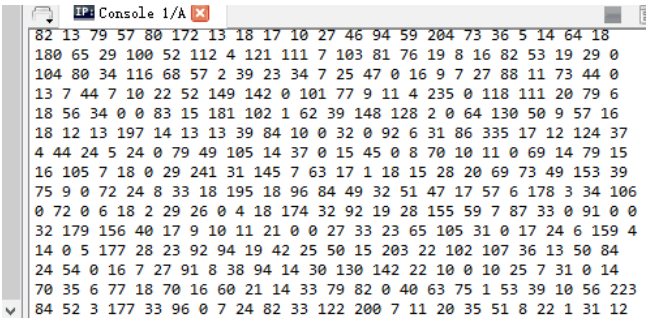


Figure 4: The numbers of FoSes of papers which are favored by real users vary.

For the academic search system *Acemap*, we run the CTR model based on 46 user-item pairs and observe the results in a straightfoward way. We (including some volunteers) act as the users to save some papers out of our own propensities in the favorite lists, and then check whether the recommended papers look appealing to us. Some results are quite thrilling. For instance, the guy who kindly helped us put the recommended paper list on the front-end saved only one paper about Bayesian classifier. Among the recommended papers, there is one paper titled "A Family of Algorithm For Approximate Bayesian inference" which seems highly related to that one. This is incredible since we didn't use any rule-based recommendation method directly. And yet some are somewhat disappointing. The recommended papers for me, for example, are not quite of my tastes. But there is a word with high frequency to appear: "risk"'. I guess this is because the papers I favorite contain no abstracts and the word "risk" also appears very frequently.

3. ZERORANK

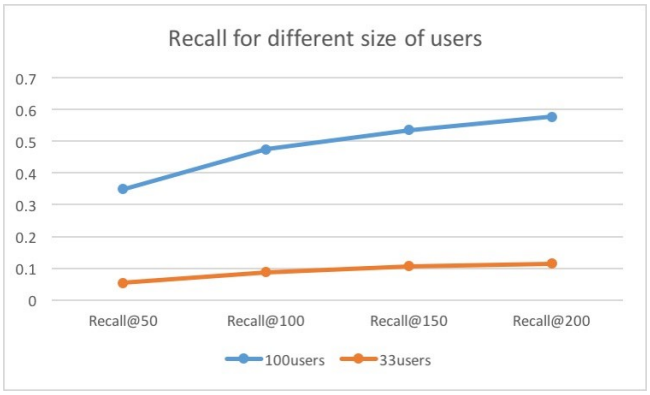


Figure 5: The recall rate of different sizes of users.

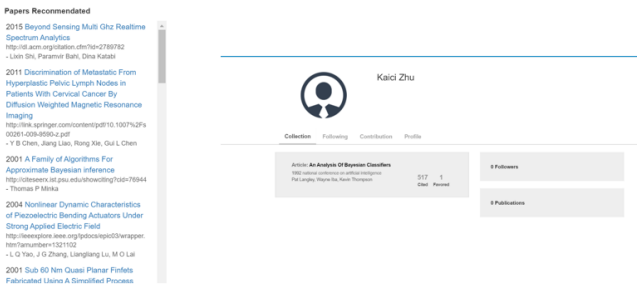


Figure 6: The front-end web page of a user who only favorites one paper.

Ranking scientific literatures is an important but challenging task. Current ranking algorithms aim to measure the prestige of each paper in a given academic network. Though the dynamic nature of citation network is considered, most of them are not specifically designed to rank nodes lying on the edge of the network, which are newly published papers without citation information, thus incur inaccurate ranking in such task. In this project, we define *zero citation ranking* problem and propose *ZeroRank* to deal with this issue. *ZeroRank* is an algorithm combining random walk on heterogeneous network and learning to rank framework. We use random walk as a feature extractor to obtain the "author", "venue", "affiliation" features of each paper, and then apply learning to rank method to train a ranking model. To make our algorithm capable of handling huge network efficiently, we design a parallel random walk algorithm and implement it on *Spark*. We conduct experiments on *Microsoft Academic Graph* and the results show that our algorithm achieves at least 17.6% improvement in NDCG score compared with the state-of-the-art literature ranking and citation prediction algorithms. We experiment on 31 subfields of computer science and observe a different finding from previous work that "author" is a dominant feature for a paper to gain future citations compared with "venue" and "affiliation".

3.1 PRELIMINARIES

We model the academic network as a heterogeneous graph containing four kinds of nodes and four kinds of edges, and define *zero citation paper set* and *zero citation ranking* problem based on it.

3.1.1 Notations and Definitions

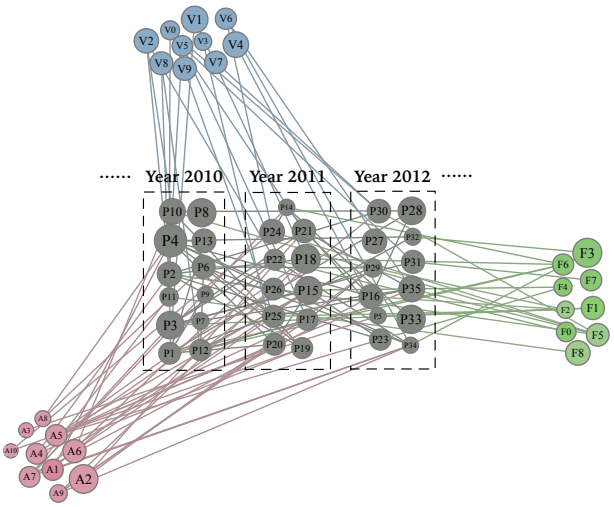


Figure 7: A demonstration of the heterogeneous graph in our method, which contains four kinds of nodes, *i.e.*, papers, authors, venues and affiliations.

Definition 1. *We denote the heterogeneous graph containing papers, authors, venues (conferences and journals) and affiliations as*

$$G = (P \cup A \cup V \cup F, E^{PP} \cup E^{PA} \cup E^{PV} \cup E^{PF}) \quad (5)$$

Where P, A, V, F are the sets of nodes representing papers, authors, venues and affiliations. Each edge $(p_v, p_u) \in E^{PP}$ indicates a reference from paper v to paper u. E^{PA} denotes the authorship. $(p_v, v_u) \in E^{PV}$ denotes paper v is published on venue u. $(p_v, f_u) \in E^{PF}$ denotes paper v has an author from affiliation u. Note that here we model the affiliation to be one "attribute" of the paper instead of the author, in order to keep the centrality of papers and perform the reinforced random walk in Algorithm 1. A demonstration of the network is shown in Figure 7.

Definition 2. *Let the heterogeneous graph consist of papers published over time $t_0 < t_1 < \cdots < t_{crt}$, where t_0 is the publication time of the oldest paper in the network, t_{crt} is the current year. Let $t(p_i)$ be the publication year of paper p_i, we define zero citation paper set Z as*

$$Z = \{p_z \in P |\ t(p_z) = t_{crt}\} \quad (6)$$

Note that in our definition, *zero citation paper set* only contains papers published in the current year, instead of older papers without citations. We also assume a paper can only cite papers older than its publication year. In other words, *zero citation paper set* is equal to the set of current year papers.

3.1.2 Problem Formulation

Zero ranking problem aims to only rank newly published papers without citations. More formally, We define the *zero ranking problem* as following: we aim to obtain a ranking model r, such that for an academic graph G, it can output $r(G)$, a permutation of the *zero citation paper set* Z of G. In other words, $r(G)$ represents a ranked list of Z according to the predicting citation number after Δt years. Our optimization goal is:

$$\max_r E(r(Z), \boldsymbol{y}_{\Delta t}) \quad (7)$$

Where $\boldsymbol{y}_{\Delta t}$ is the ranked list according to real citation numbers after Δt years. $E(\boldsymbol{l}_x, \boldsymbol{l}_y)$ denotes a performance measure function for an output list $\boldsymbol{l}_x$ with baseline list $\boldsymbol{l}_y$. In our work we set $\Delta t = 5$. Note that this problem is different from citation prediction, because here we do not care the exact citation number, but an order by predicted future citations.

3.2 ZeroRank Algorithm

Table 1: Notations

Notation	Explanation
P, A, V, F	Papers/authors/venues/affiliations nodes in Graph G
$\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{v}, \boldsymbol{f}$	Papers/authors/venues/affiliations scores in random walk
$\boldsymbol{x_t^a}, \boldsymbol{x_t^v}, \boldsymbol{x_t^f}$	"author", "venue", "affiliation" features for slice t
$\boldsymbol{y_t}$	Real citation rank list for slice t
$\{(\boldsymbol{x_t^a}, \boldsymbol{x_t^v}, \boldsymbol{x_t^f}, \boldsymbol{y_t})\}_{t=t_0}^{t_{crt}-1}$	Training set
r	Ranking model
$E(\cdot, \cdot)$	Performance measure function
$k_n \in \{k^A, k^V, k^F\}$	weaker ranker

In this work we choose "author", "venue", "affiliation" as features to characterize a paper's potential of being popular in the future, because intuitively a paper written by an influential author, from a top tier venue or affiliation is likely to receive more citations in that it is probably of high quality and is more likely to be read. However, giving a quantitative measurement to these "attributes" is difficult. If we directly apply citation based statistics such as IF to define an author or a venue, we are likely to ignore the difference of citing papers.

In fact, the measurement of papers, authors, venues and affiliations are correlated. For example, the measurement of an author is based on its publications. Therefore, in our algorithm, instead of using citation based statistics to define features, we design a reinforced random walk as a feature extractor.

To better use the information we have, we shift our current time t from t_0 to $t_{crt} - 1$ to construct a training set, and each time we hide the information after the current time point t, and can form a new *zero citation paper set*. We then perform our feature extraction algorithm on the network of this time and obtain a set of features. Moreover, this method enables us to obtain the citations between t to t_{crt} as a label for training. We denote the set of features and citations when current time point is set to t as slice t. By applying this method we can have a training set containing $t_{crt} - t_0$ slices: $S = \{(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}, \boldsymbol{y_t})\}_{t=t_0}^{t_{crt}-1}$, where $\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}$ are "author", "venue", "affiliation" features for slice t, and $\boldsymbol{y_t}$ is real citation rank list for slice t.

To train a ranking model combining different features, many citation prediction methods apply *linear regression* or *KNN*. However, these methods are not suitable for our problem due to its unique character: it is unreasonable to train one *linear regression* model using two papers from different slices because they are from different years and their citation numbers may be influenced by their different historic situations. In addition, the real citation observation interval $t_{crt} - t$ decreases as t increases, making the smaller interval slice inevitably receive fewer citations. To deal with this issue, we design a learning to rank algorithm based on *AdaRank*. Each slice is handled as a query, and with boosting training, we optimize our ranking model iteratively by the performance measure function, and maintain the weights for each feature.

There are 3 phases in our algorithms, which are random walk phase, learning to rank phase and deployment phase. In the random walk phase, we perform a reinforced random walk to get the authority scores of authors, venues and affiliations for each paper. In the training phase, we use the authority scores as features and real future citations as labels to train a learning to rank model. In the deployment phase, we use the trained learning to rank model with the authority scores to predict the zero citation rank.

3.2.1 Feature Extraction phase

The random walk algorithm, which leverages citations, authors, venues, affiliations and time information, is based on the following assumptions:

- Important papers are often cited by many important papers.
- Influential researchers are more likely to publish high quality papers, and high quality papers increase their authors' influence.
- Top tier venues are more likely to publish high quality papers, and high quality papers increase venues' reputation.
- Top tier affiliations are more likely to publish high quality papers, and high quality papers increase affiliations' fame.
- Recent papers are more convincing in showing the authority of authors, affiliation and venues, as well as the popularity of papers at present.

Based on these assumptions, we design our reinforced random walk algorithm shown in Algorithm 1. Initially each paper will be assigned a score of $\frac{1}{N}$, where N denotes the total number of papers. Then the algorithm performs an iterative computation until convergence when for any paper i, its scores of two consecutive iterations p_i and p_i' satisfy $|p_i' - p_i| < \epsilon$, where we set $\epsilon = 10^{-9}$.

Each iteration contains two steps:

- The scores of authors, venues and affiliations are computed by their related papers.
- The scores of papers are obtained by their related papers, authors, venues, affiliations as well as a time-aware constant.

In the first step, we compute author, venue, affiliation scores by averaging their related papers' scores. $AVG(\cdot)$ denotes the average value function. For example, for author i, his/her score will be the average score of all the publications belonging to him/her.

In the second step, a paper's score is obtained by linear combining these five parts: scores of papers citing it, authors, venue, affiliations it related, and a time constant. For the first part, a classic *PageRank* algorithm will be used. For computing the authors/affiliations, an average function

Algorithm 1 *ZeroRank* Random Walk

Require:
Graph: G
Parameter: $w_1, w_2, w_3, w_4, w_5, \rho, t_{crt}$
Ensure:
Scores: $\boldsymbol{p}$,$\boldsymbol{a}$,$\boldsymbol{v}$,$\boldsymbol{f}$
for all paper i **do**
  $p_i = \frac{1}{N}$
while not converge **do**
  for all author i **do**
    $a_i = AVG_{P_j \in neigh(A_i)}(p_j)$;
  for all venue i **do**
    $v_i = AVG_{P_j \in neigh(V_i)}(p_j)$;
  for all affiliation i **do**
    $f_i = AVG_{P_j \in neigh(F_i)}(p_j)$;
  for all paper i **do**

$$
\begin{aligned}
p_i' = \quad & w_1 \sum_{P_j \in in(P_i)} \frac{p_j}{|out(P_j)|} + \\
& w_2 \frac{1}{Z_A} AVG_{A_j \in neigh(P_i)}(a_j) + \\
& w_3 \frac{1}{Z_V} AVG_{V_j \in neigh(P_i)}(v_j) + \\
& w_4 \frac{1}{Z_F} AVG_{F_j \in neigh(P_i)}(f_j) + \\
& w_5 \frac{1}{Z_T} \exp(-\rho(t_i - t_{crt}))
\end{aligned}
$$

will be used, and since one paper can only published on one venue, no average function is needed to process venue part. Note that to make sure the algorithm converges, a normalization process is performed to make all the scores adding to papers from authors, venues, affiliations and time sum to 1 by normalization variables Z_A, Z_V, Z_F, Z_T. And for the last part, because old papers have more edges, which exaggerate their impact even if they are obsolete, we use a damping factor ρ to compensated newly published papers. $w1 \sim w5$ denote the weights for the five parts, and sum to 1.

Note that in real dataset, the information of authors, venues and affiliations is often incomplete. To tackle this problem, we bring the idea of virtual nodes . For example, if a paper u has no author, we will give it a virtual author whose publication contains only u.

Our random walk is in consistent with the five assumptions, and reveals the innate reinforcement relations between papers, authors, venues and affiliations. In particular, only the citation reinforcement is unidirectional, becauses papers only contribute to their references but not vice versa. Nevertheless, paper-author, paper-venue, paper-affiliation relationships are mutual reinforcement.

Convergence

Here we prove the convergence of the random walk phase. We rewrite Algorithm 1 into the matrix form. Let A_P,A_A, A_F,A_V denote the normalized adjacent matrices of the graphs $G_P = (P, E^{PP})$, $G_A = (P \cup A, E^{PA})$, $G_V = (P \cup V, E^{PV})$, $G_F = (P \cup F, E^{PF})$. $\tilde{A_A}, \tilde{A_F}, \tilde{A_V}$ denote the normalized transpose adjacent matrices of the same graphs. The States 5,7,9 in Algorithm 1 can be rewritten as

$$\boldsymbol{a} = A_A \boldsymbol{p} \tag{8}$$

$$\boldsymbol{v} = A_F \boldsymbol{p} \tag{9}$$

$$\boldsymbol{f} = A_V \boldsymbol{p} \tag{10}$$

Since $||p||_1 = 1$, we can denote the last term in State 11 as $(\boldsymbol{d} \times \boldsymbol{e})$, where $\boldsymbol{d}$ is the damping factor, and $\boldsymbol{e} = (1 \cdots 1)$ is the vector consisting of all $1s$. Then iteration process for paper can be rewritten as:

$$
\begin{aligned}
\boldsymbol{p_{k+1}} = (\omega_1 A_P + \omega_2 \tilde{A_A} A_A + \omega_3 \tilde{A_V} A_V \\
+ \omega_4 \tilde{A_F} A_F + \boldsymbol{d} \times \boldsymbol{e}) \boldsymbol{p_k}
\end{aligned} \tag{11}
$$

We denote $M = \omega_1 A_P + \omega_2 \tilde{A_A} A_A + \omega_3 \tilde{A_V} A_V + \omega_4 \tilde{A_F} A_F$, and according to [10] the sequence $\boldsymbol{p_k}$ will converge to the unique principal eigenvector of M.

3.2.2 Learning to Rank phase

After the random walk algorithm converges, we can obtain the "author", "venue", "affiliation" features of each paper i by compute the average authority scores related to P_i:

$$x_i^A = AVG_{A_j \in neigh(P_i)}(a_j) \tag{12}$$

$$x_i^V = AVG_{V_j \in neigh(P_i)}(v_j) \tag{13}$$

$$x_i^F = AVG_{F_j \in neigh(P_i)}(f_j) \tag{14}$$

Here note that the features computed by Equations 12, 13, 14 are innately of the same scale, which can be used in training without another scaling process.

In order to construct the training set, we shift our current time t from t_0 to $t_{crt} - 1$, each time we hide the graph information after the current time point t, obtaining a new *zero citation paper set*. We then perform our feature extraction algorithm on network of this time and get a set of features. By apply such method we can have a training set containing $t_{crt} - t_0$ slices: $S = \{(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}, \boldsymbol{y_t})\}_{t=t_0}^{t_{crt}-1}$, where $\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}$ are "author", "venue", "affiliation" features for slice t, $\boldsymbol{y_t}$ is real citation ranking for slice t. In practice we observe that t_0 is not necessarily set to the first year of graph G, because time slice that is too old could help little to reveal the authority at present, thus in our experiment we set $t_0 = t_{crt} - 10$.

Then we modify the *AdaRank* algorithm to train a ranking model based on the training set. In Algorithm 2, a boosting algorithm is performed. Each iteration we select a weak ranker and finally we linearly compose all the weak rankers to obtain the ranking model r.

To be specific, a weight distribution for each slices is maintained. We denote it as $\boldsymbol{P_n}$ for nth iteration. After each iteration, $\boldsymbol{P_n}$ is modified to make those slices which have a "bad" ranking result take up a higher weight, and those having a "good" ranking result take up a lower weight. In this way, in the next iteration *ZeroRank* will try to select a weak ranker focusing on these "hard" slices. Furthermore, a weak ranker will be directly selected from the features. That is, we select a weak ranker which can maximize Equation 15, where $E(\cdot, \cdot)$ is the performance measure function. For example, weak ranker k^A will rank only according to the author feature. After k_n is selected, α_n will be computed as a measurement of effectiveness of k_n. And finally r is composed of the linear combination of k_n with weight α_n.

Note that in our algorithm we track the performance changing, and stop the iterative process when the performance no

longer increases. According to Theorem 1 in [18], there exists a lower bound for the accuracy of the training function. Meanwhile, since the performance continuously increases and has an upper bound 1, the algorithm must converge.

Algorithm 2 *ZeroRank* Ranking Model Training

Require:
$S = \{(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}, \boldsymbol{y_t})\}_{t=t_0}^{t_{crt}-1}$
Ensure:
Ranking model r
1: $P_1(t) = \frac{1}{t_{crt}-t_0}$
2: **while** performance increasing **do**
3: create weak ranker $k_n \in \{k^A, k^V, k^F\}$ such that

$$\max_{k_n} \sum_{t=t_0}^{t_{crt}-1} P_n(t) E(k_n(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}), \boldsymbol{y_t}) \quad (15)$$

4: $\alpha_n = \frac{1}{2} \ln \frac{\sum_{t=t_0}^{t_{crt}-1} P_n(t)(1+E(k_n(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}), \boldsymbol{y_t}))}{\sum_{t=t_0}^{t_{crt}-1} P_n(n)(1-E(k_n(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}), \boldsymbol{y_t}))}$
5: $r_n = r_{n-1} + \alpha_n k_n$
6: $P_{n+1}(t) = \frac{\exp\{-E(k_n(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}), \boldsymbol{y_t})\}}{\sum_{t=t_0}^{t_{crt}-1} \exp\{-E(k_n(\boldsymbol{x_t^A}, \boldsymbol{x_t^V}, \boldsymbol{x_t^F}), \boldsymbol{y_t})\}}$

3.2.3 Parallel Random Walk

We first analyze the complexity of our algorithm. The random walk phase time has the same time cost as *PageRank*, that is $O(M/\ln(1/\alpha))$ [2], where M denotes the number of edges and α is the damping factor. And the learning to rank phase will cost $O(TNlogN)$ [18] in time, where N is the number of papers in the training set and T denotes the number of iterations.

Besides the time cost, the space cost, $O(M+N)$, makes it unacceptable to run this algorithm on a single node, implying a parallel algorithm is needed. Experiment result in Figure 8 shows that the random walk phase takes a major running time. Consequently, we propose a parallel solution for random walk and implement it on *Spark*. As shown in Algorithm 3, there are two kinds of nodes, which are nodes of authors, venues and affiliations and nodes of papers. The first category runs $RankAVF$ procedure while the second runs $RankP$ procedure. During each iteration, nodes get a message list as parameters and then update their values before they send messages to their adjacent nodes.

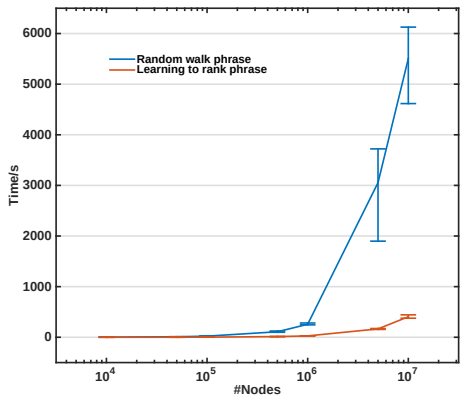


Figure 8: The running time of random walk phase and learning to rank phase among different sizes. The algorithm is implemented is *C++* and performed on a server with 2 Intel Xeon CPU E5-2650 v3 processors and 128 GB ram.

To be specific, for a P node, it will first compute a score based on $Pmsgs$ (denoting messages from P nodes in the previous iteration) and $Amsgs$ (denoting messages from AVF nodes). After that, it will compute and send messages to its successor P nodes and adjacent AVF nodes. While an AVF node computes the authority score based on $Hmsgs$ and transmits to P nodes. If the score computed by one P node converges, the node will vote to stop, and the algorithm halts if all nodes vote to stop.

3.3 Experiment

3.3.1 DataSet

We use *Microsoft Academic Graph(MAG)*[5] provided by *Microsoft* [13] as our experiment dataset. We first remove papers published in 2017 and edges pointing from past to future, obtaining 126, 908, 750 papers published from 1800 to 2016 as well as 526, 449, 409 edges. And the dataset also contains 114, 698, 004 authors, 23, 404 journals, 1, 283 conferences and 19, 843 affiliations. The distribution of papers

[5]http://research.microsoft.com/en-us/projects/mag/

Algorithm 3 *ZeroRank* Parallel Random Walk

Require:
Graph: G
Parameters: $w_1, w_2, w_3, w_4, w_5, \rho, t_{crt}$
Ensure:
Scores: $\boldsymbol{p}$,$\boldsymbol{a}$,$\boldsymbol{v}$,$\boldsymbol{f}$;
1: **procedure** RankAVF(v:$AVFid$, $Hmsgs$:$List$)
2: var $msgSum, msgNum = 0$
3: **for all** $m \leftarrow msgs$ **do**
4: $msgSum$ += m
5: $msgNum$ ++
6: $v.score = msgSum/msgNum$
7: **for all** $j \leftarrow neighP(v)$ **do**
8: $msg = v.score$
9: $send\_msg(to{=}j,\ msg)$ ▷ send to adjacent papers

10: **procedure** RankP(v:Pid, $Pmsgs$:$List$, $Amsgs$:$List$)
11: var $msgSum.p, msgSum.a, msgSum.v, msgSum.f$=0
12: var $msgNum.a, msgNum.v, msgNum.f = 0$
13: **for all** $m \leftarrow Pmsgs$ **do**
14: $msgSum.p$ += m
15: $v.score = w1 * msgSum.p + w5 * v.timeScore$
16: **for all** $m \leftarrow Amsgs$ **do**
17: $msgSum.(m.type)$ += $m.value$
18: $msgNum.(m.type)$ ++
19: **for** i *in* $\{a, v, f\}$ **do**
20: $v.score$ += $w_i * Norm(msgSum.i/msgNum.i)$
21: **for all** $j \leftarrow v.outP$ **do**
22: $msg = v.score/|v.outP|$
23: $send\_msg(to{=}j,\ msg)$ ▷ send to adjacent papers
24: **for all** $j \leftarrow v.outAVF$ **do**
25: $msg = (type{=}j.type, value{=}v.score)$
26: $send\_msg(to{=}j,\ msg)$ ▷ send to adjacent AVFs
27: **if** converged($v.score$) **then**
28: $voltToHalt(v)$

Table 2: Distribution of Papers Over Publication Year

year	before 2000	2000	2001	2002	2003	2004	2005	2006	2007
# of papers	47,627,409	2,668,362	2,759,899	3,020,753	3,240,642	3,514,052	3,833,380	4,370,690	4,702,195
year	2008	2009	2010	2011	2012	2013	2014	2015	2016
# of papers	5,177,040	5,964,730	6,144,130	6,562,040	6,750,651	7,315,938	7,057,228	5,729,525	470,086

over publication year is shown in Table 2.

In the following experiments, we extract different subsets of *MAG* to test different features of our algorithm. The detail about how we extract the subsets will be described in related subsections.

In the following subsections, we first choose the parameter ρ in our algorithm based on computer science field, and then perform 4 experiments. The first experiment shows that the random walk phase does extract features leading to high citations, and the second presents that comparing to other scientific ranking and citation prediction methods, our algorithm has a high accuracy on *zero ranking problem*. Then we compare the performance of *ZeroRank* and *FutureRank* based on varying parameters and observe our method has an average better result. Finally, we run our algorithms on 31 subfields of computer science to find the major feature for highly cited papers.

Aging Parameter Choosing

In the random walk phase, the aging parameter ρ compensates scores of newly published papers, thus finding a best value of ρ can adequately model the aging effect. To achieve this, since in the following experiments we mainly focus on CS field, we first extract a subset of papers whose keywords map to computer science, which contains $8,884,763$ papers. Then we plot their citation numbers over time after published. Inspired by [11], we ignore the points for year 0, 1 after published, and find the best exponential function which matches the figure is:

$$ce^{-0.124t} \tag{16}$$

So we set $\rho = -0.124$. It is interesting to point out this is different from $\rho = -0.62$[11] for arXiv (hep-th) dataset, which implies these two datasets have different structures.

3.3.2 *Feature Extraction*

Table 3: Top 10 "author", "venue", "affiliation" features extracted by *random walk*

Rank	Author		Venue		Affiliation	
	Cits	CRank	Cits	CRank	Cits	CRank
1	15	4	15	1	0.5	369
2	28	3	3	5	0	1156
3	77	1	0	636	4.25	7
4	7.5	22	1	56	3.9	10
5	12	8	0	636	0	1156
6	41	2	0	636	0	1156
7	2	451	3	5	3	12
8	8	19	0.5	218	3	12
9	5	80	6	2	0.4	515
10	0	5572	0.5	218	0.5	369

In this experiment, we evaluate *ZeroRank's* ability to extract features that related to high citations. We first extract the set of papers whose keywords map to both computer science and data mining, and obtain $20,320$ papers, $29,586$ authors, $1,738$ venues and $2,836$ affiliations. We first set the current time $t_{crt} = 2006$ and adjust the parameter that gives us the best prediction result in 2011. Then we set $t_{crt} = 2011$ to evaluate the feature extraction process.

Table 3 shows the top 10 "author", "venue", "affiliation" features extracted by *random walk*, where "Cits" denotes real average citations and "CRank" denotes ranking for "Cits". Note that we do not aim to select authors, venues, affiliations publishing a large quantity of papers, but those who are likely to publish highly cited papers. So we evaluate the result by comparing the average citation instead of total citations. We can observe that *ZeroRank* extracts promising features because of all the 30 items above, 15 of them are in the top 30 CRank. In addition, the performance of "author" feature extraction is the best and 7 of the items are in top-20 features from 29586 in total. While the affiliation seems to be the worst, but considering 59.3% affiliations have no citation from 2011 to 2016 years, many affiliations themselves are indistinguishable.

3.3.3 *Zero Citation Ranking*

In this experiment we evaluate *ZeroRank* for *zero citation ranking* problem by comparing with state-of-the-art ranking and citation prediction algorithms.

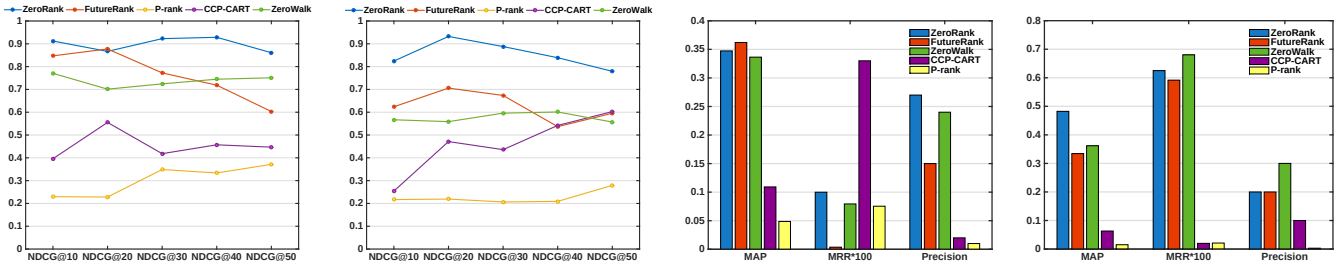
Dataset and Time Setup

We select two subsets from *MAG*: data mining(DM) and database(DB). For each dataset, we first collect all papers whose keywords map to the label, and then enlarge the set by adding other papers directly linked it. Finally we obtain DM with $461,392$ papers, and DB with $434,442$ papers.

Similar to experiment in Subsection 3.3.2, we first adjust the parameters of all the following algorithms on the current time point 2006 with the future citation numbers from 2006 to 2011. After that we evaluate them on the current time from 2011 with future citation numbers from 2011 to 2016. The parameters $w_1 \sim w_5$ for data mining and database are $0.4, 0, 0.1, 0.1, 0.4$ and $0.4, 0, 0.3, 0.2, 0.1$.

Baseline

We compare our algorithms with *FutureRank* [11], *P-rank* [19], *CCP-CART*[20], and *ZeroWalk*. *CCP-CART* is a citation prediction method using Classification and Regression Tree, and we rank the paper list according to the predicting citations. Due to the limit of dataset, for *CCP-CART* we do not implement content based features. *ZeroWalk* denotes the random walk phase of our algorithm without learning to rank. For *ZeroRank*, we set the performance measure function in learning to rank phase as NDCG@10, because we would like to focus on the top highly cited papers.



(a) *NDCG-DM* (b) *NDCG-DB* (c) *Others-DM* (d) *Others-DB*

Figure 9: Figure 9a, 9b illustrate the NDCG scores of DM, DB, Figure 9c, 9d show MAP, MRR, Precision scores of DM and DB. All methods choose the best parameters adjusted from 2006 to 2011.

Evaluation Metrics

In order to evaluate the performance of these ranking algorithms, We introduce four kinds of metrics. Since a large part of the papers in *zero citation paper set* obtain no citation from 2011 to 2016 and researchers mainly care a small portion of papers, it's adequate to choose metrics that emphasize the top part of papers:

NDCG Normalized discounted cumulative gain [6] measures the performance based on the ground truth and gives top results high weights. The NDCG score is computed as

$$NDCG@k = n_i \sum_{i=1}^{k} \frac{2^{r_i} - 1}{log_2(i+1)} \tag{17}$$

where r_i is the score of the ith paper, k is a constant and n_i is the normalization constant to ensure that a ideal rank will get score 1. Since citation number is unsuitable to directly used in NDCG, We sort the papers according to their future citations in descending order, and assign the $0\% - 10\%$, $10\% - 30\%$, $30\% - 60\%$, $60\% - 100\%$ papers with scores 3, 2, 1, 0 separately.

MAP Mean of the average precision scores [1] is defined as following:

$$MAP@k = \sum_{i=1}^{k} \frac{p_i}{d_i} \tag{18}$$

where p_i is the ith relevant paper while d_i is the rank of the this paper in the list, k is the number of papers we compute, where we set $k = 100$. In our experiment, we set papers of top 1% future citations with $p_i = 1$, otherwise $p_i = 0$.

MRR: Mean reciprocal rank, which measure the performance of a ranking algorithm by top 1 paper in real rank:

$$MRR = \frac{1}{x} \tag{19}$$

where x is the position of real top 1 paper ranked by the algorithm.

Precision: Precision[11] is defined by the union of real top k papers with the top k returned by algorithm.

$$Precision@k = \frac{|\text{realTopk} \cap \text{rankTopk}|}{k} \tag{20}$$

In the experiment we set $k = 100$.

Experiment Result

Figure 9a, 9b demonstrate the NDCG scores of the five algorithms. Figure 9c, 9d show the MAP, MRR, Precision scores. To plot MAP, MRR, Precision scores on one figure, we scaling the scores of different metrics.

For NDCG, we can observe that in both datasets *ZeroRank* achieves the best score, with average scores 0.898 and 0.853. While *FutureRank* obtains the average scores 0.764 and 0.627. The score of *ZeroRank* is 17.5% and 35.9% higher separately. And comparing *ZeroRank* with *ZeroWalk*, we can observe *learning to rank* refines the ranking result by adjusting the weights of each feature according to training set.

For MAP, MRR and Precision, we can observe *ZeroRank* and *ZeroWalk* together achieve 4 best results of 6. And for the comparison between *ZeroRank* and *ZeroWalk*, the result shows that *ZeroWalk* outperforms *ZeroRank* in some cases. This can be explained by the NDCG@10 performance measure function we choose in learning to rank phase. Different metrics are not ideally identical and a trade-off is implied between them. The performance measure function help *ZeroRank* achieve a higher score in NDCG while leads a lower score in other metrics. However, if we choose different performance measure function, we can achieve higher score based on the measurement function. This implies another advantage of learning to rank phase, that is enabling us to select different optimization goal based on our unique requirement.

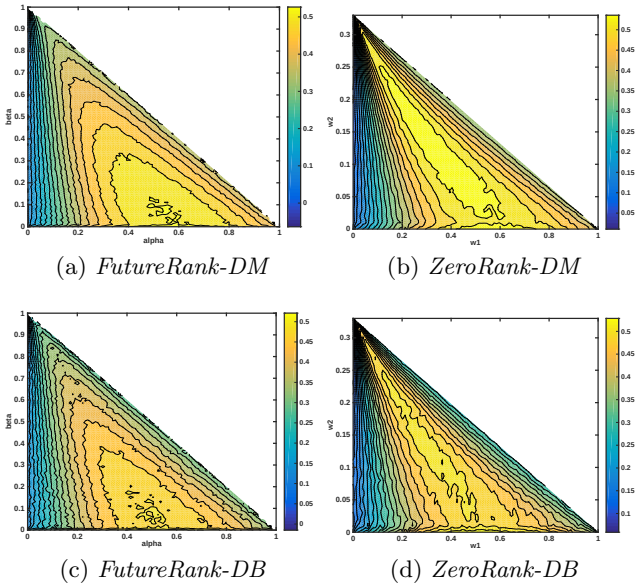
3.3.4 *Performance Comparison Based on Varying Parameters*

In this Subsection we evaluate the performance of *ZeroRank* based on varying parameters. We use the DM and DB in Subsection 3.3.3 as the evaluation datasets, and compare *ZeroRank* with *FutureRank*. Considering the learning to rank phase uses additional information and refines the feature weights, we only compare *FutureRank* with the *ZeroRank*'s random walk phase.

We choose the Spearman's rank correlation coefficient [8] as our evaluation metric, which measures the similarity of two rank lists:

$$\rho = \frac{\sum_i (R_1(P_i) - \bar{R}_1)(R_2(P_i) - \bar{R}_2)}{\sqrt{\sum_i (R_1(P_i) - \bar{R}_1)^2 \sum_i (R_2(P_i) - \bar{R}_2)^2}}$$

Where $R_1(P_i)$ and $R_2(P_i)$ are the rank positions of paper i in rank list 1 and rank list 2. $\bar{R}_1$ and $\bar{R}_2$ are the average



(a) *FutureRank-DM* (b) *ZeroRank-DM*

(c) *FutureRank-DB* (d) *ZeroRank-DB*

Figure 10: Spearman's rank correlation coefficient for *FutureRank/ZeroRank* on DM/DB. Comparing the left and right part, we can conclude that *ZeroRank*'s random walk phase has a better score for different parameter settings

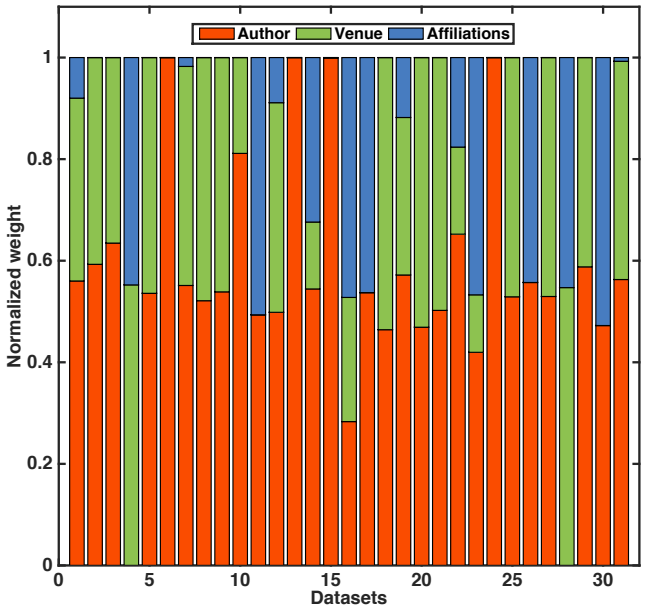


Figure 11: The Normalized Weight for ZeroRank Running on 32 Subfields of computer science. We can see "author" is a dominant feature for *zero ranking problem*, and "affiliation" is of the least importance.

rank positions of rank list 1 and 2.

FutureRank [11] has three parameters α, β, γ, and $\alpha+\beta+\gamma = 1$. We enumerate its parameter setting and set the x-axis as α and y-axis as β. For *ZeroRank*, since it involves 5 parameters $w_1 \sim w_5$ and $\sum_{i=1}^{5} w_i = 1$, we can not enumerate the parameters freely. Considering $w_2 \sim w_4$ denote the authority weights for authors, venues and affiliations, here we set these three with the same value, *i.e.*, $w_2 = w_3 = w_4$. And we enumerate parameters with x-axis as w_1 and y-axis as w_2.

Figure 10 demonstrates the results. Figures 10c, 10d denote *FutureRank* and *ZeroRank* on database dataset, while Figures 10a, 10b denote *FutureRank* and *ZeroRank* on data mining. The color in each point represents the Spearman's rank correlation coefficient, where the brighter point indicates a higher score. From both pairs we can find *ZeroRank* outperforms *FutureRank*. For a quantitative measure *ZeroRank* achieves an average value 0.3988 in database and 0.4238 in data mining, while *FutureRank* achieves 0.3759 in database and 0.3908 in data mining. From these results, we can draw the conclusion that ZeroRank not only has a higher maximum, but also has a higher average score among all kinds of parameter settings.

Note that from Section 3.3.5 we can know actually these features have significantly different weights, so equal weight setting by all means decreases *ZeroRank* score. This means the average score of *ZeroRank* could be higher if we select a non-equal setting.

3.3.5 Feature Importance in CS Field

Learning to rank algorithm gives us the ability to quantitatively measure the importance of different features in determining whether a paper will receive more citations than others in the future.

We conduct this experiment based on "field of study" information in the dataset. We focus on the papers belonging to computer science. According to the dataset, there are 35 subfields[6] under CS.

We first perform a preprocessing among the full dataset. A paper containing author, venue, affiliation information and mapping both to CS and one of the subfields will be selected and other papers will be ignored. After this step, we delete 4 subsets, which are computational science, Internet privacy, management science, and theoretical computer science, because they have too few papers. Finally we get a dataset containing papers of 31 subfields. Each subfield has 1204 to 77,898 papers. And the total papers is 388,688. Note that we choose a rather "strict" filtering condition, causing the final qualified paper set small. A looser filtering method may get a different result but it is possible to involve many papers outside CS field.

We run *ZeroRank* separately on 31 datasets and the result is shown in Figure 11, where we can find the "author" feature dominates most datasets. The average normalized weight of this feature is 56%, and the "venue" feature follows with an average weight 29%. And the least important feature is "affiliation". The result is somehow amazing, because intuitively there will be not so huge differences. But recalling the citation distribution demonstrating in Figure 1, we can understand why "venue" and "affiliation" seem not to be so promising.

4. CONCLUSION AND FUTURE WORK

In this project, we implement a novel recommendation method in a real academic search system *acemap*. We prove experimentally that the CTR model is suitable for content-based recommendation like recommending papers and for

[6]Fields are map to hierarchy from $L0$ to $L3$ from high to low. Here CS is mapped to $L0$ and the 35 subfields we choose are mapped to $L1$

new systems without many users. There are a mass of things we can do in the future. Firstly, about the model itself, the number of topics (the dimension of latent features) K needs to be tuned. Secondly, currently each step of the recommendation is executed manually, we need to write batch files to let them run automatically. There are other questions such as how to simulate virtual users more accurately, how to select raw data to better satisfy users' preferences, how to process the data in a more fine grained way. In addition, there are a myriad of other recommendations in the system that need to be done. For instance, recommend papers in topic homepage; Recommend similar papers in paper homepage; Recommend relation-based and interest-based authors in author homepage. Although these kinds of recommendations mentioned above have been implemented, they are very simple in that they just use the count of citations or information from nearby networks and some of the results are poor. Therefore, we have to research more to find or design methods for these kinds of recommendations.

In the second part, we propose a new ranking problem *zero citation rank*, which means ranking newly published scientific articles without citations in an academic network. To deal with this issue, we introduce a novel algorithm *ZeroRank*. It leverages the citations, authors, venues, affiliations and time information to construction a heterogeneous network, and uses a reasonable random walk algorithm as a feature extractor. After that, it trains an efficient ranking model based on *learning to rank* technology. To enable the algorithm's scalability, we parallel the random walk phase and implement it on *Spark*. Experimental evaluations show that our algorithm outperforms the state-out-art scientific ranking and citation prediction algorithms. We also do an experiment and find that in computer science field, "author" is a dominant feature in making your paper gain more citations than others. For future work, we plan to do experiments on other fields to see whether they have the same major feature as computer science. Furthermore, we will add more features such as innovativeness of papers, popularity of topics or degree of difficulty of algorithms to the learning to rank phase and measure their importances. Finally, we will further study why "author" plays such a significant role while "venue" and "affiliation" do not.

5. REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[2] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment*, 4(3):173–184, 2010.

[3] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger. Research paper recommender system evaluation: a quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, pages 15–22. ACM, 2013.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[5] F. Didegah and M. Thelwall. Determinants of research citation impact in nanoscience and nanotechnology. *Journal of the American Society for Information Science and Technology*, 64(5):1055–1064, 2013.

[6] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM, 2000.

[7] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[8] J. L. Myers, A. Well, and R. F. Lorch. *Research design and statistical analysis*. Routledge, 2010.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[10] T. Sauer. *Numerical Analysis*. Pearson Addison Wesley, 2006.

[11] H. Sayyadi and L. Getoor. Futurerank: Ranking scientific articles by predicting their future pagerank. In *SDM*, pages 533–544. SIAM, 2009.

[12] J. Shen, Z. Song, S. Li, Z. Tan, Y. Mao, L. Fu, L. Song, and X. Wang. Modeling topic-level academic influence in scientific literatures. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, and K. Wang. An overview of microsoft academic service (mas) and applications. WWW - World Wide Web Consortium (W3C), May 2015.

[14] D. Walker, H. Xie, K.-K. Yan, and S. Maslov. Ranking scientific publications using a model of network traffic. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(06):P06010, 2007.

[15] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[16] S. Wang, S. Xie, X. Zhang, Z. Li, S. Y. Philip, and X. Shu. Future influence ranking of scientific literature. In *SDM*, pages 749–757. SIAM, 2014.

[17] Y. Wang, Y. Tong, and M. Zeng. Ranking scientific articles by exploiting citations, authors, journals, and time information. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[18] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[19] E. Yan, Y. Ding, and C. R. Sugimoto. P-rank: An indicator measuring prestige in heterogeneous scholarly networks. *Journal of the American Society for Information Science and Technology*, 62(3):467–477, 2011.

[20] R. Yan, J. Tang, X. Liu, D. Shan, and X. Li. Citation count prediction: learning to estimate future citations for literature. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1247–1252. ACM, 2011.

[21] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng,

E. P. Xing, T.-Y. Liu, and W.-Y. Ma. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1351–1361. International World Wide Web Conferences Steering Committee, 2015.