1-1-2015

Plus: A Unique Personalized Literature Recommender System

Jingwen Zhang
Wayne State University,

PLUS: A Unique Personalized Literature Recommender System

by

Jingwen Zhang

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2015

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

DEDICATION

Dedicated to my parents and my little brother. To my husband, Yuanzhe Li.

ACKNOWLEDGMENTS

This is not a easy job for a beginner to conquer by himself. So, I am deeply grateful to my advisor, Prof. Weisong Shi for his help guiding me to the right direction. Without his guidance, I would not have the honor to show my research in this article.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the rapid development of the Internet Era, people have been accustomed to the convenient lifestyle that the Internet has brought to them. Among the numerous means invented by the internet people, there is no doubt that recommendation is playing an increasingly important role in improving the quality of human life in all aspects. The most popular field of recommendation is online shopping, and it almost can be seen everywhere nowadays whenever you go to Amazon.com, eBay.com or any other shopping websites. The increased sales benefited from the recommendations can not be ignored as well.

The reason that recommendation becomes an indispensable to people's life can be explored. With the coming of the Big Data Era, the resources exposed to each individual via the internet is extremely large, and it will be very difficult for a human being to find out what he or she demands in a short period. Undoubtedly, the advent of recommendation provides an efficient method for human beings to solve this problem.

It turns out that people have benefited a lot from recommendation; therefore, researchers and advanced business people start to explore where recommendation can go much further and deeper. Nowadays, in addition to the field of online shopping, recommendation appears in a wide variety of fields such as entertainment, food, education, sports and healthcare. Many applications with recommendation features such as Netflix [1], Youtube [3] and Yelp [12] remains top ranks of popularity among users.

We are concentrating on discovering the possibilities of papers' recommender systems that are more related with our students' research work. Currently, a huge amount of papers are published every year and become accessible online. For students, especially the graduate students who are absolutely inexperienced about their research interest

or who are not proficient enough in their research filed, they have to enhance their knowledge by reading papers from different researchers or groups. The time spent on searching cannot be ignored because it could be a very long period from the aspect of efficiency. A paper recommender system can better reduce the unnecessary wasted time.

In this thesis, we proposed a unique paper or literature recommender system, PLUS, with the aim to assist users in discovering what they want to read and meanwhile become supernumerary experience during their research. Unlike other paper recommender systems that take users' input then generate recognizable tags as their interests and make recommendations, PLUS provides a general and personalized methodology of learning users' interests. In addition to the main source of users' reading history, it considers the notes taken when they read papers as a secondary source of predicting their interests, which makes the predictions more accurate. Moreover, the system will mostly recommend the latest papers to users. It also allows the recommendations to be dynamically updated with the change of users' recent reading activities so that the system can always keep track of users' interests at the current stage.

1.1 Thesis Objective

The objective of this thesis is to present a unique personalized literature recommender system and address some challenges of the existing recommender systems for scholar papers. Generally speaking, we can summary these challenges with the following statements:

- Unrepresentative source to predict users' interest
- Low accuracy in making predictions
- Unable to recommend the latest papers to users
- Unable to update the recommendations timely when users' interests are changed

Starting from the above challenges, this thesis aims to illustrate the detailed features of PLUS from four aspects:

- Representative source to learn users' interest
- Higher accuracy of predictions
- Able to recommend the latest papers to users
- Timely update the recommendations with the change of users' interests

1.2 Thesis Motivation

Nowadays the item-based recommendation has become a mature technique that are widely used in on-line shopping. However, the content-based recommendation involving text mining hasn't achieved much as expected, especially the recommendation of scholar papers. Researchers and students usually search on the Google Scholar or other digital library websites such as IEEE Xplore and ACM Digital Library to find out the papers that they are interested. There are still some challenges that need further studied. First of all, the time spent on finding an satisfying paper is long. Assuming that each satisfying paper needs at least 10 minutes to be found, for a student who writes a literature review which requires at least 40 reference papers, the student has to spend at least 400 minutes, which is approaching 7 hours, to find the papers that he or she is interested. Secondly, the papers found via Google Scholar or other digital library websites are not always what people want because the keywords provided to the search engines are limited and cannot represent a complete paper accurately. Thirdly, some papers that are available on-line but unpublished are invisible in Google Scholar and other digital libraries. Last but not least, the research interest of a particular person could change with the time going by. Most of the existed paper recommender systems can't update the recommendations when a user's interest changes.

Therefore, we proposed a personalized literature recommender system (PLUS) in this thesis, which was designed to recommend the latest published and unpublished papers to users and dynamically update the recommendations based on users' recent research activities. The experiments showed that PLUS could recommend the papers that the users are interested and shorten the time consuming on searching papers.

1.3 Thesis Organization

To accomplish the goals of the thesis, the remaining sections are organized as follows: the second chapter will give a brief introduction to the background knowledge of the recommendation algorithms; the third chapter will list the related work about the literature recommender system; the design and implementation details of PLUS will be demonstrated in the fourth chapter; the fifth chapter will show the evaluation results of PLUS; and the last section will highlight the contributions of the thesis and the work that needs to be further studied in the future.

CHAPTER 2

TECHNICAL BACKGROUND

The recommendation algorithm defines the work flow that how the recommendations are learned and generated. Traditionally, the recommendation algorithms can be classified into three categories: collaborative filtering, content-based filtering, and the hybrid ones. The collaborative filtering is the most widely used in the development of recommender systems. Basically, it predicts a user's preference or rating for a particular item from other similar users' preferences or ratings. A typical instance is the on-line shopping recommendations by Amazon.com [10]. The content-based filtering is usually applied to discover the users' preferences through the description of items. For example, when recommending a book, we need to consider the category of a book, the author, the publisher, etc. The hybrid algorithm combines the features of both collaborative filtering and content-based filtering. It takes advantages of collaborative filtering and content-based filtering to achieve higher accuracy of recommendations and better performance of the system.

2.1 Collaborative Filtering

As aforementioned, the collaborative filtering is a very popular technique to predict users' ratings of items. It can also be divided into three kinds: memory-based collaborative filtering, model-based collaborative filtering, and the hybrid collaborative filtering [15]. The following subsections will demonstrate the three different kinds of algorithms in details.

2.1.1 Memory-based CF

In the memory-based collaborative filtering, the mechanism usually uses users' preferences or ratings to compute the similarity between users or items. If a user's rating for

a particular item is predicted from user-to-user utilization matrix, this kind of method is called *user–based* collaborative filtering, while if the user's rating for an item is obtained from the item-to-item utilization matrix, this kind of method is called *item–based* collaborative filtering. Both of the user-based collaborative filtering and the item-based collaborative filtering can be considered as memory-based. A typical example of memory-based collaborative filtering is the recommender system deployed by Amazon. When a user searches an item at the on-line store of Amazon.com, there are always some recommendations listed at the bottom of the website, which contain the items of some similar buyers compared with the user or some similar items that the user has purchased before.

2.1.2 Model-based CF

The model-based collaborative filtering predict users' preferences or ratings from the probabilistic perspective [2]. This method builds probabilistic models for collaborative filtering such as cluster models and Bayesian networks. In [7], the researchers investigated the differences of two models, the aspect model and the two-sided clustering model, for make predictions on users' individual preferences and choices. In [8], Jon and Mark proposed a recommendation algorithm in a probabilistic mixture model and introduced certain parameters which were identified to be able to achieve the best performance of a recommender system. The experimental results showed that any system which was capable of being bounded with these parameters had the possibility to generate the optimal recommendations with the growth of data size.

2.1.3 Hybrid CF

As aforementioned, data sparsity and scalability issues will influence the performance of the memory-based approaches. Although the model-based methods have been introduced to mitigate these effect of these problems, they also bring other limitations in the range of the users; therefore, some researchers have made an attempt to explore combining the memory-based and the model-based into hybrid format.

2.2 Content-based Filtering

Content-based filtering strategies make recommendations to a user based on the description of the items that the user has rated before . In this method, the system no longer cares about the ratings of items. On the contrary, it will take the features of items into consideration. It is usually necessary to obtain the user's historical data, then the system select some key features which could identify the items uniquely and extract them. From these features, the system always try to quantify and formulate the user's preference or interest. For the test data to be recommended, the system will compute the similarity between the test data and the historical data in the form of the formulas and recommend the items with the highest similarity. Content-based filtering is often applied in the field of information retrieval such as text recognition. For example, a book usually has its title, author, publisher, and publication date. If you want to be recommended a book, a basic content-based system would find the commonalities between the books that the user has read in the past, learn the user's preference for books, and recommend the ones with the highest similarity to the user's taste. Many machine learning such techniques as Bayesian, decision tree can be deployed in the content-based filtering systems.

2.3 Hybrid

Although the content-based filtering is able to mitigate the influences caused by data sparsity, the cold-start problem and the difficulties in scalability still exist in the content-based filtering. The cold-start problem refers to that the system is not able to recommend to new users because the new users don't have enough historical data, and their preferences cannot be learned accurately. Generally people always deploy the hybrid method to solve the cold-start problems. The following Table2.1 shows the problems embedded in these three different recommendation strategies.

	Data sparsity	Difficulty in scalability	Cold-start problem	Profile injection attacks
Collaborative filtering	yes	yes	yes	yes
Content-based filtering		yes	yes	yes
Hybrid		yes		yes

Table 2.1: Problems exist in different recommendation strategies

CHAPTER 3

RELATED WORK

The concept of paper recommender systems are not brand new at all. Many existing works have made great efforts in introducing more efficient recommendation algorithms and applications. In this chapter, we will briefly review several representative works of recommender systems; then, we will compare some well-known evaluation metrics in measuring the performance of a recommender system.

Compared with these existing works, PLUS is innovative in taking both of the notes of users and their following researchers and organizations into consideration to predict users' personal interests, and a comprehensive survey is also deployed to evaluate the performance of the PLUS, which will enhance the personality of PLUS to some degree.

3.1 Existing Paper Recommendation Systems

In order to design and implement a good recommender system, learning what users' interests plays an important role in making their desired recommendations. Due to the popularity of the collaborative filtering approach, people easily proposed to deploy similar strategies to predict users' research interests from textual information of papers. During the procedure, computing the similarity of two papers is a crucial step.

In [14], the authors proposed a novel methodology, SimCC, which was capable of computing the similarity of scientific papers in a more accurate and efficient way. SimCC could effectively represent both authority and context of a paper in computing similarities. In addition, they also introduced SimCC+A to take the recently-published papers into consideration. The authors argued that content and citations were two interrelated information, so they introduced a *contribution score* to measure the authority of a paper p on a single term t indicating how much a paper contributes to another paper q on

the term t and a *relevance score* which was equal to the TF/IDF vlaues of the paper terms. The SimCC score of term t to paper p is a combination of its *relevance score* and *contribution score* to paper q, which represents both the content and authority of a paper p on term t. The SimCC score of term t in paper p can be formulated as:

$$SimCC_t(p) = \lambda \times R_t(p) + (1 - \lambda) \times \Sigma_d \Sigma_{q \epsilon D(p,d)} C_t(p, q) \tag{3.1}$$

, where $R_t(p)$ is the relevance score of term t in paper p, $C_t(p, q)$ is the contribution score of term t in paper p to paper q, and $0 \leq \lambda \leq 1$ is the relative importance factor to combine the relevance score and contribution score of term t. Based on the experiments, the accuracy of similarity measures for scientific papers has been improved dramatically.

The way in [14] inspired us a lot in computing the similarity between scientific papers accurately and efficiently. In our PLUS, the input to the system comes from different sources. How to balance the importance of these various of sources and get the most accurate prediction is a challenge for us.

Scienstein [6] is another research paper recommender system, which was developed in 2009. It introduced the first hybrid research recommender system, which combines all the known methods including citation analysis, author analysis, source analysis, implicit ratings, and explicit ratings in the hybrid system to overcome the disadvantages. It also introduced two innovative methods 'Distance Similarity Index' (DSI) and the 'In-text Impact Factor' (ItIF) which were first used in developing paper recommender systems. The Scienstein was proved to be a powerful alternative to academic search engines by not only relying on keywords analysis but also relying on all the known analysis to improve the performance; however, the system couldn't solve the problems of privacy issues resulting implicit ratings and explicit rating.

In 2011, Kazunari Sugiyama and some researcher from Singapore [16] proposed a serendipitous recommendation for scholarly papers with the consideration of the relations

among researchers. The recommender system is more suitable for junior researchers to broaden their horizon and learn new areas, while senior researchers can discover interdisciplinary frontiers to apply integrative research. They adapted a state-of-art scholarly paper recommender system's user profile construction to make use of information drawn from dissimilar users and co-authors to specifically target serendipitous recommendations.

In 2012, Pinata Winoto and Tiffany Ya Tang [18] introduced collaborative filtering to develop a paper recommender system. They proposed a context-aware multidimensional paper recommender system that considers additional user and paper features. They argued that their methods would address two key issues: (1)how would the modified filtering perform when target users are inexperienced undergraduate students who have a different pedagogical background and contextual information-seeking goals such as task- and course-related goals from those of graduate students?; (2) should they combine graduates and undergraduates in the same pool, or should they separate them? Their system was only efficient for inexperienced users, and it didn't provide good performance for different levels of users.

Joonseok Lee, Kisung Lee [9] together with other researchers worked on a personalized academic research paper recommender system in 2013. They argued that most researchers rely on key-based search on browsing through proceedings of top conferences and journals to find their related work. In order to ease the difficulty, they proposed their web crawler to retrieve research papers from the web. The recommendation algorithm used in their system is collaborative filtering, and the results showed that their systems recommend good quality research papers.

In 2013, Siwipa Pruitikanee and other researchers [13] proposed a global and soft approach to a paper recommender system. They introduced a novel approach that embeds the whole process for selecting papers of interest given some keywords, and their approach

is based on a work-flow integrating fuzzy clustering of the papers, the computation of a representative summary paper per cluster using OWA operators, and ranking, in order to answer user queries adequately. Summarizing sets of papers into a single representative one promotes the procedure simplified when users interact with huge number of papers from the literature.

From the above existing works in recommender systems, we have realized that users' research interests are dynamically changing along with the time, and users' preferences can be analyzed from different angels. In order to develop a real personalized recommender system, users' opinions can not be ignored. In our PLUS, we distribute a comprehensive survey to users to on the one hand evaluate the performance of PLUS and on the other hand to help understand their real needs and interests better.

3.2 Existing Evaluation Metrics

There are some common metrics and methods to reflect whether a recommender system is "good" or "bad". These metrics are *accuracy*, *coverage*, *serendipity*, *computingtime*, and *robustness*. We will review how the metrics are measured in the following subsections.

Accuracy

Most of the early recommender systems concentrated on the accuracy of the recommendations. Generally speaking, the metric of *accuracy* measures how close the recommendations could be to the users' true interest. For collaborative filtering, *Root Mean Squared Error* (RMSE) is one of the most popular metrics used to compute the accuracy of ratings. Suppose that we have r'_{ui} to be the predicted rating for a data set S of user-item pairs (u, i) and r_{ui} to be the true rating that is known, the RMSE is given by:

$$RMSE = \sqrt{\frac{1}{|S|} \sum_{(u,i) \in S} (|r'_{ui} - r_{ui}|)^2} \tag{3.2}$$

Many recommender systems adopt machine learning knowledge to design the algorithms. In the machine learning domain, *precision* and *recall* are the most commonly used metrics to measure the prediction usage of the algorithms. We define the *precision* and *recall* based on the Table3.1.

	TRUE	FALSE
TRUE	TP: true positive	FP: false positive
FALSE	TN: true negative	FN: false negative

Table 3.1: Confusion matrix

The $TRUE$ and $FALSE$ values in the first column represent the results in reality, while the $TRUE$ and $FALSE$ values in the first row represent the prediction results. TP, FP, FN,TN means the number of results in each permutation of the results. $Precision$ is used to answer the question that what percent of positive predictions were correct, and *recall* is used to answer the question that what percent of positive cases occurred. We use two equations listed below to demonstrate.

$$precision = \frac{TP}{TP + FN} \tag{3.3}$$

$$recall = \frac{TP}{TP + FP} \tag{3.4}$$

The metric of *accuracy* can also be defined using the table above. We give the equation below.

$$accuracy = \frac{TP + FN}{TP + FN + TN + FP} \tag{3.5}$$

Coverage

Most of the recommender system treat accuracy as the most metric of evaluating the quality of the recommendations; however, there are other factors that should be taken into account when the developers design the system. Among them, *coverage* is a vital metric for evaluating whether the recommendations are good or not. In [5], the authors explained that the coverage for a recommender system would be high if the recommendations were produced by considering more details of the items comprehensively. Refer to [5], it expressed the coverage in different formulas based on different definitions. One definition is called *predictioncoverage* , and the other one is called *catalogcoverage*. For example, if we use D to denote the set of available items and D_p to denote the set of items to be recommended by a system. In this case, the prediction coverage demonstrates the percentage of the items that the system could recommend; thus, we can formulate the prediction coverage as below:

$$prediction\_coverage = \frac{|D_p|}{|D|} \tag{3.6}$$

Catalog coverage is usually measured by conducting a number of recommendations during a predefined period of time. Take the same example as the above one. Additionally, suppose that N recommendations are produced during that period, and we use D_L^i to denote the set of items in the list L generated by the i^{th} recommendations, the catalog coverage can be expressed using the following equation.

$$catalog\_coverage = \frac{|\bigcup_{i=1...N} D_L^i|}{|D|} \tag{3.7}$$

Serendipity

People always expect new things from recommendations. Serendipity could measure the novelty of recommendations by determining if there are some unexpected items recommended to users, which exactly satisfy the users' interest [11].

Computing time

The computing time is another factor that developers need to consider when they design their recommendation algorithms. It usually reflects how fast the recommender system produce recommendation results. Originally a single machine is used to run the system. Since the scale of data is keeping increasing, more and more people tend to deploy a cluster of nodes or machines to distribute the burden of the computation [19]; however, the distribution introduces other problems such as data transmission. In this case, we cannot ignore the importance of the computing time of a recommendation algorithm because the system is useless even if it has high prediction quality but extremely low computing power.

Robustness

Nowadays not only the amount of data is increased at a very fast rate, the sources of data are also becoming more diversified. A good recommender system should be able to be robust enough when the input data is changed. In [4], the researchers defined the robustness of a recommender system to be the ability of the algorithm to make good recommendations regardless the existence of noisy or sparse data.

Besides the above metrics we have mentioned, users can define other metrics and evaluate their systems from various angels. The Table3.2 [4] [5] [11] summarizes most of the metrics that have been discussed before by other researchers.

Metrics	Description
Accuracy	How close the recommendations are to users' interest
Coverage	How broad the recommendations could be
Effectiveness	How useful the recommendations could be to users
Serendipity	How surprise the users could be by the recommendations
Computing time	How fast the system could run
Robustness	How robust the system could be with the noisy data and sparse data

Table 3.2: Evaluation metrics for recommender systems

CHAPTER 4

RESEARCH CONTRIBUTION

In this thesis, we presented a unique personalized literature recommender system - PLUS. It is different from other paper systems in providing more personalized recommendations by predicting users' research interests from various of sources including users' reading history, reading notes, and the authors and published sources that users are interested. The reasons for choosing these sources are understandable. First of all, researchers or students working on research projects have to read a lot of papers related with their research interests. Keeping track of the topics of the papers or articles that they have read before will help to find out their research interests. Secondly, reading notes can have several meanings. They can be the part containing the critical information of the article, they can be the point that the reader doesn't understand, or they can be the point that the reader has questions for. The readers' interests are also hidden in the content of the notes, and almost every student or researcher would take notes when he or she is reading a research paper. Last but not the least, researchers and students who are working on research projects always pay attention to the top researchers' work or the top groups' achievements; therefore, it will help to predict their research focus by understanding the topics of these top features.

PLUS utilities the predictions from a variety of sources to recommend the latest publications to users so that they are able to know the latest revolution in their research filed, and it is also designed to be capable of timely updating the recommendation results as long as the back-end database is updated. Another contribution of the thesis is to evaluate the system by surveying users' feedback. It makes PLUS more personalized.

In this chapter, we will review the design of PLUS following with the implementation details.

4.1 PLUS Design

4.1.1 Requirements

Data and components compose a complete system. In the system of PLUS, we need to establish a back-end database on the server side to guarantee the availability of the articles to be recommended. This back-end database is denoted as the resource pool, referred to the Figure 4.1, which stores all the articles to be recommended, and each record in the database contains basic information of a research article including the attributes of title, abstract, authors, publisher, published date, etc.

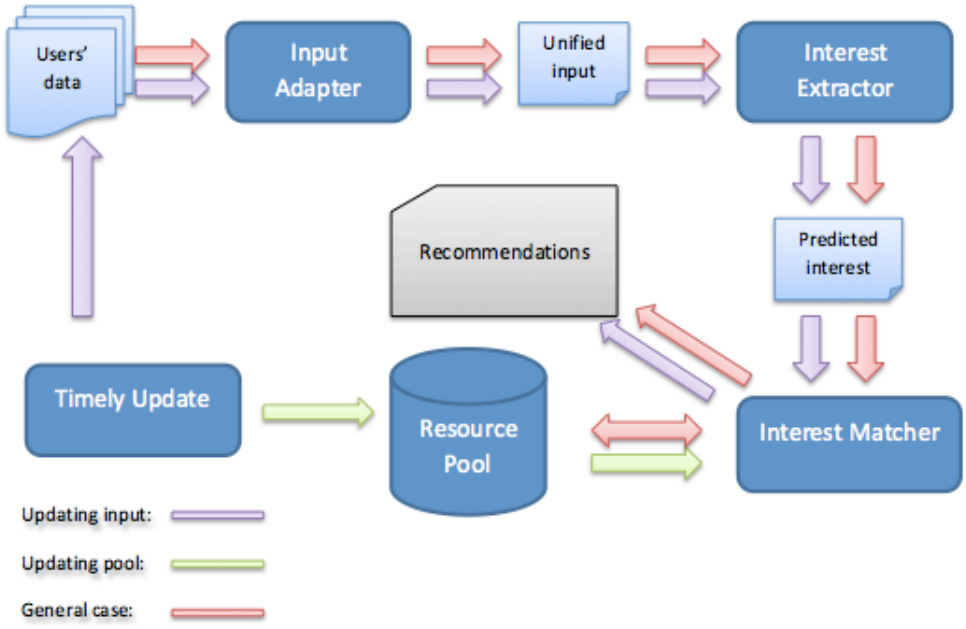

Figure 4.1: Overview of Dataflow

As aforementioned, PLUS is designed to take diverse data sources to predict users' interests. In this case, it requires the system to have the capability of processing different types of data. To simplify the process, we assume that all the input data are structured in text files.

4.1.2 Architecture

The Figure 4.1 briefly shows how data flows in the system of PLUS. The system has totally five components, three of which are the processing units responsible for producing recommendation, and one of which is the storage component that stores all the articles to be recommended, and another component is functioning to refresh recommendations with the update of the resource pool or the input data. The red flow in the figure represents the general procedure before producing the final recommendations, which goes through initial preparation of input, interest prediction and articles matching from the Resource Pool. The green flow denotes the influence on the final results when updating the Resource Pool, and the purple flow denotes the impact on the final results when updating the users' input data.

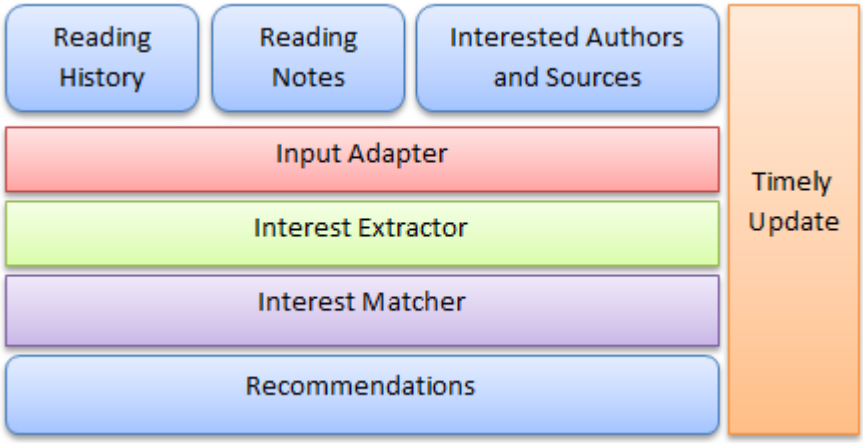

Figure 4.2: Hierarchical Design

Compared with other paper recommender systems, one of the advantages of PLUS is to take multiple sources of users' data to make predictions of their interests, and these sources are not only limited in the papers that users have read before. The first innovation is analyzing the notes that users have taken when they read papers, and the other innovation is considering the authors and publication sources like conferences and journals that users always follow up. These two sources surely can reflect users' interest

from some points. The other advantage of PLUS is to the timely update to both the users' data and the Resource Pool. This additional module allows the system to keep track of users' interests better. The Figure 4.2 shows the hierarchical design of PLUS as well.

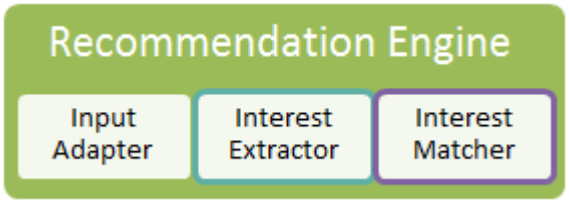

Figure 4.3: The components of Recommendation Engine

Recommendation engine plays an important role in a recommender system, which performs the core functions of making recommendations in the system. Refer to the Figure 4.3, the recommendation engine of PLUS is composed of three modules: (1)the Input Adapter module, (2)the Interest Extractor module, and (3)the Interest Matcher module. The Input Adapter is working to transforming different sources of users' data into a unified format of data which can be recognized by the following other modules. A unified format of data promotes the system to be generalized for different formats of data from different sources. It is convenient if we want to add more sources to predict users' interest in the future. It also allows other modules to be capable of providing a unified interface. The Interest Extractor is used to make predictions on users' personal interests. It analyzes users' input and extract users' interested topics. Combined with their interested authors and publication sources, the Interest Matcher will look through the Resource Pool and find out the articles that match the users' personal interest mostly.

The Timely Update module is designed to keep track of either the changes to the users' input data or the changes to the Resource Pool; therefore, the impact of the Timely Update module will be crossing the other modules. If there are new articles added to the Resource Pool, the Timely Update module will re-execute the Interest

Matcher module to refresh the recommendations for all the users. For a particular user, his or her reading history and reading notes are actually changing over time; therefore, the predicted research interest could also be changed as the time goes by. The Timely Update will be responsible for periodically updating the user's input data and making recommendations.

4.1.3 Prototype design

It is complex to implement the entire system at the beginning; therefore, we first designed a prototype to simplify the implementation. In the prototype design, we adopted the structured text file as the unified format of data. The Input Adapter should transform users' input into the structure text file and make it recognizable by the Interest Extractor module. For a particular user, the information of these papers that the user has read before can be stored in the database or come from the user's input from web application. After the transformation by the Input Adapter, these reading histories will be stored in a text file, where each record is formed using the .bib structure and contains a paper's title, authors, abstract, keywords, publisher, published date and so on. We deployed some note-taking applications such as the Microsoft OneNote and the EverNote to help manage the notes for each paper. The Input Adapter will read the notes from the these applications and transform them into sets of words in a text file. The text file simplifies the input format for the system.

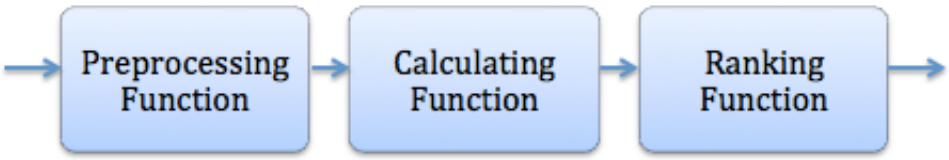

Figure 4.4: Core Functions of the Interest Extractor Module

As aforementioned, the Interest Extractor is functioning to predict personal interest

for a user. In the prototype, it is designed to have three core functions to complete the task: the preprocessing function, the calculating function, and the ranking function, seen from the Figure 4.4. The preprocessing function is used to accumulate the data from various of sources for a particular user and extract the key attributes that could uniquely represent each document. For example, if we use u to denote a random user, d_j to be the ith paper in the set of the user's reading history $\{p_1, p_2, ..., p_i, ..., p_N\}$ and n_i to be the notes for the ith paper, and we also use a_j and c_k to be the jth author and kth source respectively in the two sets of authors $\{a_1, a_2, ..., a_j, ..., a_M\}$ and sources $\{c_1, c_2, ..., c_k, ..., c_T\}$, then the job of the preprocessing function is to generate three sets of keywords, authors and sources denoted by $\{k_{i1}, k_{i2}, ...\}$, $\{a_{i1}, a_{i2}, ...\}$, and $\{c_{i1}, c_{i2}, ...\}$ with corresponding sets of frequencies $\{kf_{i1}, kf_{i2}, ...\}$, $\{af_{i1}, af_{i2}, ...\}$, and $\{cf_{i1}, cf_{i2}, ...\}$ for each document d_i, shown in the Figure 4.5. However, it is not easy to extract the keywords with some academic meanings that could represent one article because it is difficult for the system to understand the semantic meaning of each word. In the prototype, we decided to remove the stop words such as "a", "an", and "the" from the context and stem the remaining words so that the "noise" of the context can be reduced as much as possible, and the collection of the stemmed words could represent the document better.

Based on the preprocessing results, the calculating function will start to collect the data sets from each document. The function will calculate the union of the two sets belonging to the same attribute from any two different documents. If two same items belongs to the same attribute from different document, the function will sum up their frequencies. For the notes n_i of the document d_i, it is denoted as $\{n_{i1}, n_{i2}, ...\}$ with the frequencies $\{nf_{i1}, nf_{i2}, ...\}$, the system will also conduct the same process for the notes. Finally, it will generate three sets of items: $\{k_1, k_2, ...\}$, $\{a_1, a_2, ...\}$ and $\{c_1, c_2, ...\}$ with three sets of weights $\{kw_1, kw_2, ...\}$, $\{aw_1, aw_2, ...\}$ and $\{cw_1, cw_2, ...\}$.

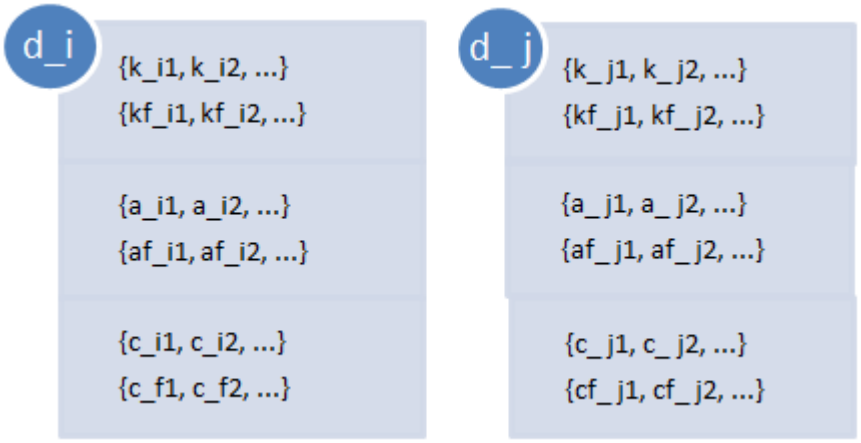

Figure 4.5: Representation for two arbitrary documents d_i and d_j

The ranking function in the Interest Extractor module is used to rank the items in the sets for the attributes generated by the calculating function based on the frequencies; then, the system will need some thresholds defined to be the size of the items in the targeted sets, which would be the sets that represent the user's personal interest. If we define S_1, S_2 and S_3, the targeted keywords would be $\{k_1, k_2, ..., k_{S_1}\}$, the targeted authors would be $\{a_1, a_2, ..., a_{S_2}\}$, and the targeted sources would be $\{c_1, c_2, ..., c_{S_3}\}$.

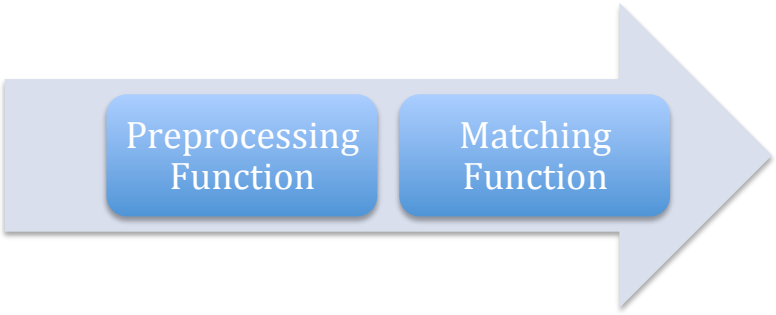

Figure 4.6: Core Functions of the Interest Matcher Module

In the Interest Matcher module, it contains two core functions: (1)the preprocessing function and (2)the matching function, shown in the Figure 4.6. The preprocessing function in this module is also used to prepare the input ready for the next function, but its input is the data stored in the Resource Pool. Each record i in the database will be

read and also represented by the three sets of keywords, authors and sources $< i$, $\{k_i1$, k_i2, ...$\}$, $\{a_i1$, a_i2, ...$\}$ and $\{c_i1$, c_i2, ...$\} >$. The matching function will find the papers from these records that match the user's predicted interest mostly. In the prototype, we will define a threshold S to be the size of the papers recommended to the user. The final results will be the sets of the id numbers of the records: $\{id_1$, id_2, ..., $id_S\}$.

The Timely Update module is responsible for periodically updating the recommendations to users due to the change of both users' input data and the resource database. In the prototype, the updates will be checked manually every once in a while to simulate the previous design. If the user's most recent reading histories are represented by $\{p_1$', p_2', ..., p_i', ..., p_N'$\}$ and n_i' is the notes for the ith article, then the new predicted interest for this user would be $\{k_1$', k_2', ...$\}$, $\{a_1$', a_2', ...$\}$ and $\{c_1$', c_2', ...$\}$ along with the sets of corresponding weights $\{kw_1$', kw_2', ...$\}$, $\{aw_1$', aw_2', ...$\}$ and $\{cw_1$', cw_2', ...$\}$, and finally the new recommendations can be denoted in the form of $\{id_1$', id_2', ..., id_S'$\}$.

4.2 Implementation Details

In this section, we will demonstrate the implementation details based on the above design of the PLUS. First of all, we will introduce how to get the unique representation for a paper and how to formulate a random user's interest; then, we will show the details for the core functions in each individual module. At the beginning, we assume that there is a user with user id U_i and a list of history papers $\{P_0$, P_1, P_2, ..., $P_n\}$, each of which has a unique paper id P_j where j ranges from 0 to n. The recommendation engine will output a recommendation list, which contains the paper ids of the recommended papers.

4.2.1 Paper Representation

In the above prototype, the preprocessing function in both of the Interest Extractor and the Interest Matcher will generate a unique representation for a single paper. Remember that the preprocessing functions takes a paper's title, abstract, authors and

sources as part of the input together with the notes, we deployed a vector to represent a individual paper, $< paper\_id, keywords\_array[], authors\_array[], sources\_array[] >$. For example, if there is paper with paper id P_j, it can be defined as $< P_j, keywords[], authors[], sources[] >$. When it comes to the implementation, we defined a class called *Paper* to represent a single paper and the basic actions that could be performed on a paper. Below is the sample code of the class.

```
public class Paper {

   private String paperID;

   private String title;

   private HashMap<String, Double> keywords;

   private HashMap<String, Double> authors;

   private HashMap<String, Double> source;


   public Paper();  //Initialization

   public String getId(); // Return paper id

   public String getTitle(); //Return paper's title

   public HashMap<String, Double> getKeywords(); //Return paper's keywords

   public HashMap<String, Double> getAuthors();  //Return paper's authors

   public HashMap<String, Double> getSource(); //Return paper's source

   public void setId(String temp_id);  //Set paper id

   public boolean setTitle(String temp_title); //Set paper's title

   public boolean setKeywords(String keyword); //Set paper's keywords

   public boolean setAuthors(String author); //Set paper's authors

   public boolean setSource(String source); //Set paper's source
```

```
}
```

4.2.2 Interest Prediction

Generally speaking, the interests prediction is more common in predicting users' preferences on some specific items. For example, when CNN wants to recommend news to users, they can analyze the page visits to various of topics on their websites such as entertainment and sports then predict their interests for news. However, it is more complicated to find out a user's research interest based on the papers that they have read because it is hardly possible to use a single specific term to represent a user's research interest. In that case, it would be more difficult to make predictions with machine learning algorithms because all the algorithms are based on mathematics. That is why we need to formulate the users' research interests. In our design, we intended to deploy a vector as well to represent a user's research interest. With the same assumption above, the user U_i 's research interest can be formulated as $< targeted_keywords[], targeted_authors[], targeted_sources >$, where the array of the targeted keywords contains the most frequently stemmed keywords from the paper's title, abstract and notes, while the array of the targeted authors and the array of the targeted sources are the ones of the papers that the user has most frequently read before. In PLUS, we only analyze the reading history of a user in the past six months because a user's research interest can also be changed. If the time gap is too long, the accuracy will be not good. In addition, one element in the targeted keywords array is composed of the keyword itself and its weight, which can be denoted with the following structure. The definitions for the targeted authors and targeted sources are similar. However, sometimes we can see different formats of the names of the sources, so we decide to use a unified form of the name of the source in PLUS, and it will be the full name of the source.

```
/** The definitions of the classes Interest, Targeted_keywords,
    Targeted_authors, and Targeted_source **/
public class Interest{
   public String userID;
   public HashMap<Integer, Targeted_keywords> targeted_keywords;
   public HashMap<Integer, Targeted_authors> targeted_authors;
   public HashMap<Integer, Targeted_sources> targeted_sources;
}
public class Targeted_keywords{
   public String keyword;
   public double weight;
   //define member functions here
}
public class Targeted_authors{
   public String first_name;
   public String last_name;
   public double weight;
   //define member functions here
}
public class Targeted_sources{
   public String source;
   public double weight;
     //define member functions here
```

```
}
```

We count the frequency of a word that appears in a paper's title, abstract and notes. If the word happens to be in both of the paper and the notes, PLUS will consider it prior to other words that are not in the notes and increase its weight by the times that it appears in the notes. If the times that a word W_s appears in the paper P_j including the paper's title and abstract is X_s where s ranges from 1 to m, and m is the number of stemmed keywords in the paper P_j's title and abstract, and the times that the word appears in the notes N_j is Y_s, the weight or the frequency KW_s of the word W_s would be calculated using the Equation 4.1.

$$KW_s = \frac{X_s + Y_s}{m} \tag{4.1}$$

Therefore, the weights for all the keywords would be donated as $< KW_1,\ KW_2,\ KW_3,\ ...,\ KW_m >$, and the weights for the authors array and the sources can be represented with $< AW_1,\ AW_2,\ AW_3,\ ...,\ AW_p >$ and $< SW_1,\ SW_2,\ SW_3,\ ...,\ SW_q >$ where p and q are respectively the number of authors and sources. After finishing the weights calculation, we predefine a default number of keywords to be put in the targeted keywords array and a default maximum number of authors and sources that to be put into the targeted authors and sources arrays. Users can specify these values when using the PLUS. We use M, A and S to represent the size respectively for the targeted keywords, authors and sources arrays. In this case, the predicted research interest for the user U_i would be $< targeted\_keywords[M], targeted\_authors[A], targeted\_sources[S] >$, where $targeted\_keywords[M] = \{\ \{\ W_1,\ KW_1\ \},\ \{\ W_2,\ KW_2\ \},\ ...,\ \{\ W_M,\ KW_M\ \}\ \}$, $targeted\_authors[A] = \{\ \{\ F_1,\ L_1,\ AW_1\ \},\ \{\ F_2,\ L_2,\ AW_2\ \},\ ...,\ \{\ F_A,\ L_A,\ AW_A\ \}\ \}$, and $targeted\_sources[S] = \{\ \{\ S_1,\ SW_1\ \},\ \{\ S_2,\ SW_2\ \},\ ...,\ \{\ S_S,\ SW_S\ \}\ \}$.

4.2.3 Similarity Formulation

The traditional way to compare the similarity of two items is to calculate the cosine similarity between the two items. When an item can't be represented with a single feature, the vector space model will be an efficient method to represent an item with multiple features. In PLUS, we applied the vector space model to represent a paper which has a unique paper id, a set of keywords, authors and source. In order to guarantee the capability of calculating the cosine, the predicted interest is also represented by a vector with the same dimensions as the paper except the id field. Assume that an arbitrary paper P_j in the Resource Pool can be formulated as $< P_j, keywords[], authors[], sources[] >$ where j ranges from 1 to the maximum size of the Resource Pool and $keywords[] = \{ \{ W_1, KW_{j1} \}, \{ W_2, KW_{j2} \}, ... \} \}$, the cosine similarity is calculated as follows:

$$S_{ij} = \frac{(\Sigma(KW_r * KW_{jr}))}{\sqrt{\Sigma(KW_r * KW_r)} * \sqrt{\Sigma(KW_{jr} * KW_{jr})}} \tag{4.2}$$

where KW_r and KW_{jr} are the weights of the keyword W_r in the array of $targeted\_keywords[]$ and $keywords[]$.

In the Equation 4.2, D_{ij} represents the distance between the user U_i's predicted interest and the paper set $\{P_j\}$ to be recommended in the Resource Pool, and the distance will only be calculated when the keywords exist in both of the targeted keywords array and the keywords array of P_j, so for the keywords that does not exist in both of the arrays, the value will be zero. The Interest Matcher will rank the papers that has been calculated based on the distance, and the smaller the distance is, the better the paper matches the predicted interest.

The Interest Matcher will not only compare the similarity of the keywords between the predicted interest and the papers in the Resource Pool but also find papers which match the targeted authors and sources. The final recommendation list will merge the papers and be provided to the user.

CHAPTER 5

EXPERIMENTAL RESULTS

We evaluate PLUS from three different aspects. The accuracy is one of the most important metrics to determine the performance of a recommender system. The accuracy tells whether a recommender system recommends items to users that they are interested or not. Although the recommendations are produced based on the predicted interest, the prediction of a user's interest or preference could be not accurate. Even though the predicted interest might be of highly accuracy, the accuracy can't still be guaranteed. In the experiments, we define another two metrics, one is completeness, and the other one is timeliness. The completeness is used to measure that if PLUS could still recommend new papers when the Resource Pool is updated, and the timeliness is used to measure if the system could save the time that users have spent on searching papers.

5.1 Accuracy

		Satisfaction
2	Trends in energy-efficient computing: A perspective from the Green500	4
3	Energy-aware checkpointing of divisible tasks with soft or hard deadlines	3
4	HomeSim: Comprehensive, smart, residential electrical energy simulation and scheduling	1
5	Hypnos: Understanding and Treating Sleep Conflicts in Smartphones	5
6	Racing and Pacing to Idle: An Evaluation of Heuristics for Energy-aware Resource Allocation	4
7	Navigating Big Data with High-throughput, Energy-efficient Data Partitioning	3
8	Local power distribution with nanogrids	2
9	Virtualizing Power Distribution in Datacenters	2
10	Vector repacking algorithms for power-aware computing	1
11	Robust Architectural Support for Transactional Memory in the Power Architecture	1

Figure 5.1: Snapshot of the recommendations generated by PLUS from five users

The first experiment was conducted on campus among several graduate students. We built a GUI to collect their reading histories and their preference of authors and sources in their research field; then, PLUS would process the data and write the recommendations to an output file. Initially, we recommend 10 papers every time, and the results are shown in the Figure 5.1. It lists the ids of the papers that have been recommended to

five users. We evaluated the quality of the recommendations by distributing a survey to these users and let them to give a score ranging from 1 to 5 where 5 means that the paper recommended is satisfied mostly, and 1 means that the paper is unsatisfied mostly. The snapshot of the survey of a student is shown in the Figure 5.1. We can get 5 out of 10 papers of which the satisfaction score is equal to or greater than 3.

	U1	U2	U3	U4	U5
1	CHI07-p219-tang	HT10-p315-lunn	SIGMOD10-p567-fu	MICRO05-24400043	ANCS10-a21-nelms
2	CHI10-p183-unruh	HT04-p126-wardrip-fruin	CIKM01-p57-jamil	MM01-p251-nack	SPAA04-p93-guez
3	CSCW10-p321-paul	HT05-p46-walker	SIGAda03-p38-rosen	CIKM08-p883	ANCS08-p60-guo
4	CSCW06-p59-munkvold	HT09-p5-bernstein	CIKM07-p331	AAMAS10-p1527-ito	SPAA05-p1-andelman
5	CSCW08-p205-tang	HT01-p61-miles	ISEC09-p89-n	SIGIR00-p49-swan	SIGCOMM06-p243-katti
6	GROUP09-p301-sarcevic	HT04-p26-kolb	SIGIR07-p559-broder	NSPW00-p51-atallah	ANCS08-p99-congdon
7	ICIS09-p1330-yuan	HT05-p54-millard	ValueTools06-a54-avrache	SIGIR00-p128-petasis	SC07-a5-abts
8	CSCW08-p215-sarcevic	HT02-p23-furuta	SC00-a47-nikolopoulos	CIKM01-p490-ponceleon	ACMSE05-p20-kocak
9	ICUIMC08-p136-cho	HT01-p113-shipman	JCDL04-p160-dalal	CC02-p134-candy	SODA07-p199-li
10	CHI09-p2061-zhou	DocEng04-p124-revilla	ISMAR06-04079273	SAC10-p1718-bosc	PODC05-p227-gordon

Figure 5.2: Snapshot of the survey for one user

The second experiment was performed with the experimental dataset released by [17]. The dataset contains fifty researchers' research interests, which were generated from the researchers' published papers in DBLP, and it also contains about 100,000 candidate papers to be recommended. Each paper is represented by its keywords and the corresponding weights, and it is stored in a text file. We retrieved ten users' information and ran PLUS to generate a list of recommendations for each user against the list of papers that has been produced by [17]. The comparison of the results is displayed in the Figure 5.2.

5.2 Completeness

The completeness measures that if PLUS could recommend new papers to users when adding new papers to the Resource Pool. In the experiment, we predicted the user's interest from the result produced by PLUS, then we ran PLUS again based on the user's new interest. By comparing the new result with the old one, the ideal completeness degree is 100% if the two results perfectly matched. In reality, the completeness degree

of PLUS can reach 60%. We evaluated the completeness of the results produced by both PLUS and PLUS without taking notes into consideration in the Figure 5.3.

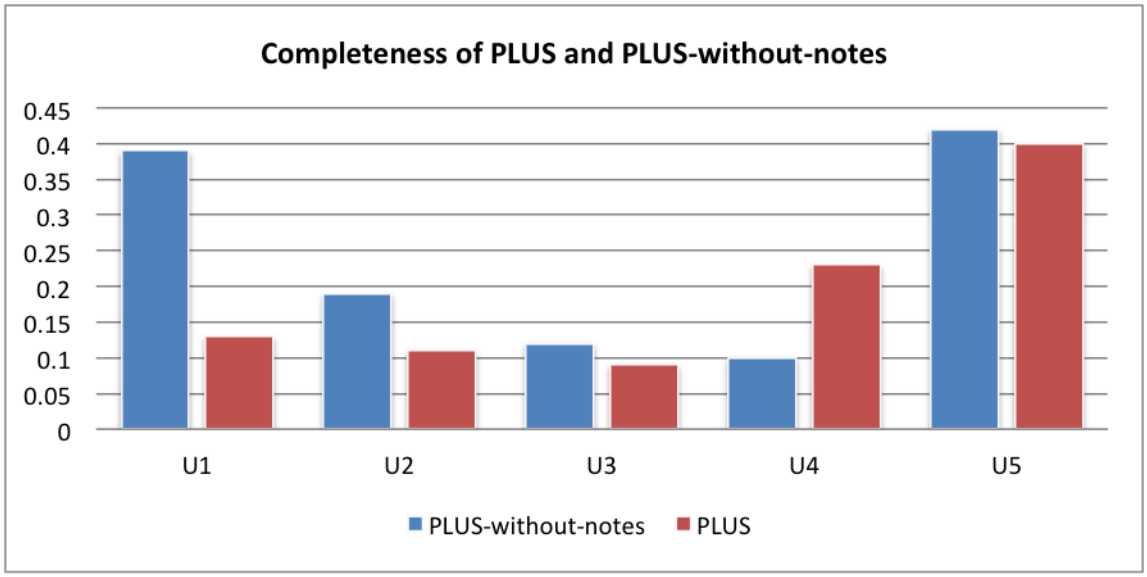

Figure 5.3: Completeness of PLUS and PLUS-without-notes

5.3 Timeliness

Researchers' research interests could change with the time going. We intended to make PLUS to update the researchers' interests when the changes occurred and update the recommendations as well. In this experiment, we kept tracked the interest changes of a specific user and ran PLUS to see the differences of the recommendations. You can see the comparison in the Figure 5.4

5.4 PLUS vs. PLUS-without-notes

PLUS is featured with taking notes as an elementary source for predicting users' research interests. Therefore, we conducted one experiment to compare the performance of both PLUS and the PLUS without the notes. We conducted a survey among five users and let them to give a score on both results of PLUS and PLUS without taking notes into consideration. The comparison can be seen in the following Figure 5.5. From the

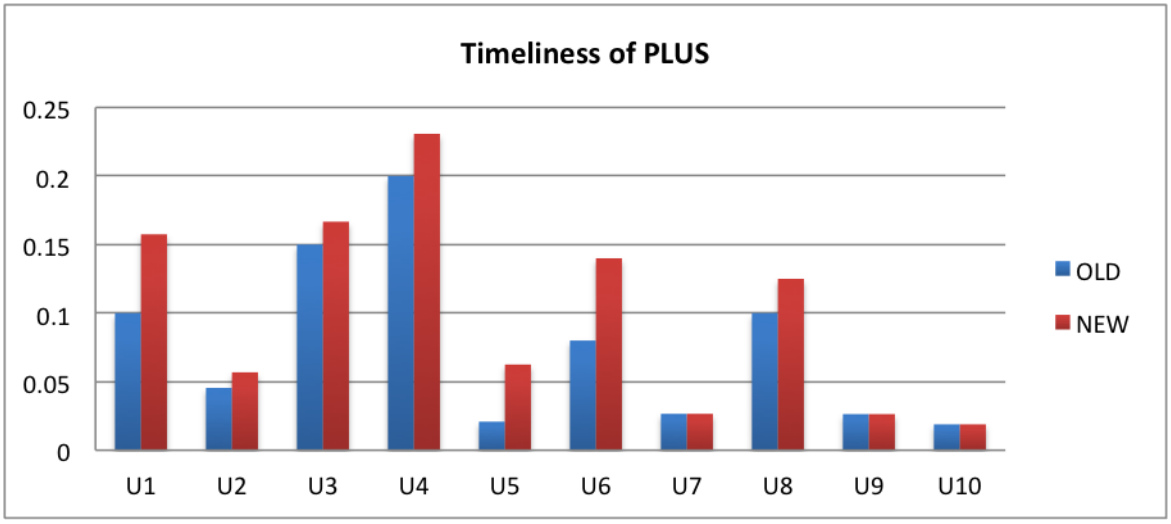

Figure 5.4: Timeliness of PLUS from ten users

results, we can predict that PLUS could make better recommendations when taking the notes as a source for predicting users' research interests.

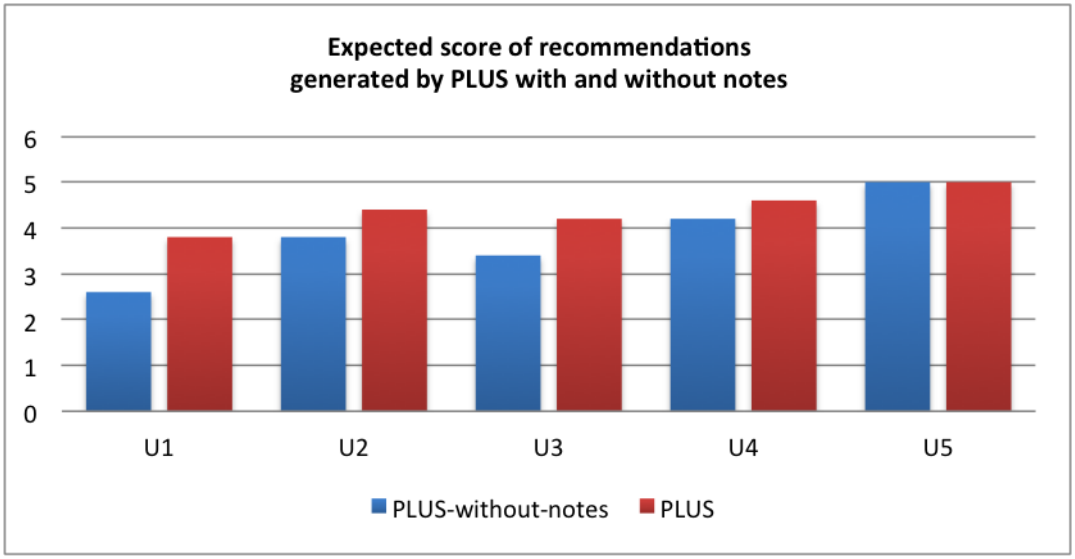

Figure 5.5: Comparsion of the results between PLUS and PLUS-without-notes

CHAPTER 6

CONCLUSION AND FUTURE WORK

In the thesis, we proposed a unique personalized literature recommender system, PLUS, to make searching papers more efficient. PLUS is innovative in the following aspects: (1) it takes multiple sources that could reflect a user's personal research interest as the input; (2) it prevents the recommendations from going outdated due to the timely update to the user's interest and the Resource Pool; (3) it is targeted to recommend the latest papers to the users; and (4) it is using a comprehensive survey to evaluate users' satisfaction to the system, which is more approaching users' real interests.

In this article, we first introduced some background knowledge of recommendation algorithms; then, we mainly concentrated on the big picture of the system and the implementation details of PLUS. Finally, we presented how the experiments were prepared to simulate and measure the performance of PLUS, and we listed different forms of results of PLUS.

By implementing the whole system and analyzing the experimental results, we found that there were still some challenges that need to be further studied. First, it is still difficult to analyze the semantics of the keywords for a paper or context. A recommendation system could benefit a lot from the accurate semantic analysis. Second, it is difficult to grab the complete information of a paper because of the copyright issues, which would make the system less accurate. Third, how to make the execution time of the system less enough to guarantee the performance is getting more difficult because the base size of the Resource Pool will get larger and larger.

In addition to solving the challenges, more work needs to be done in the future to make the system more user-friendly and intelligent, and achieve good performance at the same time.

BIBLIOGRAPHY

[1] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[2] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[3] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[4] Franois Fouss and Marco Saerens. Evaluating performance of recommender systems: An experimental comparison. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '08, pages 735–738, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM.

[6] Bela Gipp, Jöran Beel, and Christian Hentschel. Scienstein: A research paper recommender system.

[7] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IN PROCEEDINGS OF THE SIXTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 688–693, 1999.

[8] Jon Kleinberg and Mark Sandler. Using mixture models for collaborative filtering. *Journal of Computer and System Sciences*, 74(1):49–69, February 2008.

[9] Joonseok Lee, Kisung Lee, and Jennifer G. Kim. Personalized academic research paper recommendation system. *CoRR*, abs/1304.5457, 2013.

[10] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[11] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[12] Vladimir Nikulin. *Hybrid Recommender System for Prediction of the Yelp Users Preferences*, volume 8557 of *Lecture Notes in Computer Science*, pages 85–99. Springer International Publishing, 2014.

[13] Siwipa Pruitikanee, Lisa Di Jorio, Anne Laurent, and Michel Sala. Paper recommendation system: A global and soft approach. In *FUTURE COMPUTING 2012, The Fourth International Conference on Future Computational Technologies and Applications*, pages 21–27, 2012.

[14] Masoud Reyhani Hamedani, Sang-Wook Kim, Sang-Chul Lee, and Dong-Jin Kim. On exploiting content and citations together to compute similarity of scientific papers. In *Proceedings of the 22nd ACM international conference on Conference on*

information & knowledge management, CIKM '13, pages 1553–1556, New York, NY, USA, 2013. ACM.

[15] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[16] Kazunari Sugiyama and Min-Yen Kan. Serendipitous recommendation for scholarly papers considering relations among researchers. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, JCDL '11, pages 307–310, New York, NY, USA, 2011. ACM.

[17] Kazunari Sugiyama and Min-Yen Kan. http://www.comp.nus.edu.sg/ sugiyama/schpaperrecdata.html, .

[18] Pinata Winoto, Tiffany Tang, and Gordon McCalla. Contexts in a paper recommendation system with collaborative filtering. *The International Review of Research in Open and Distance Learning*, 13(5), 2012.

[19] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.

ABSTRACT

PLUS: A Unique Personalized Literature Recommender System

by

Jingwen Zhang

May 2015

Advisor: Dr. Weisong Shi

Major: Computer Science

Degree: Master of Science

There are massive research papers published from various of disciplines every year, and people who are engaged in scientific research usually have to spend a large amount of time on searching and finding the papers that they are interested in. In this thesis, we illustrated a unique personalized literature recommender system (PLUS) which was proposed to predict users' personal research interests and recommend the latest papers to them as much as possible. The system shows advantages in four aspects: (1) it takes multiple sources that could reflect a user's personal research interest as the input; (2) it prevents the recommendations from going outdated due to the timely update to the user's interest and the Resource Pool; (3) it is targeted to recommend the latest papers to the users; and (4) it is using a comprehensive survey to evaluate users' satisfaction to the system, which is more approaching users' real interests. The experimental results showed that PLUS was capable of discovering the papers that users are indeed interested, and the time that the users have consumed on searching papers was reduced significantly with the help of this recommender system.

AUTOBIOGRAPHICAL STATEMENT

Jingwen Zhang

5057 Woodward Ave,

Detroit, MI 48201

(313)460-3858

Education:

M.S.	Computer Science
	Wayne State University, 2011-2015
	Advisor: Weisong Shi
B.S.	Computer Science
	Xidian University, China, 2008-2011