Conceptual, Impact-Based Publications Recommendations

Conceptual, Impact-Based Publications Recommendations

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

by

Ann Smittu Joseph
Cochin University of Science and Technology
Master of Science in Computer Applications, 2005
Mahatma Gandhi University
Bachelor of Science in Computer Applications, 2002

December 2013
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dr. Susan Gauch
Thesis Director

Dr. Brajendra Panda Committee Member	Dr. Dale R. Thompson Committee Member

ABSTRACT

CiteSeer^x is a digital library for scientific publications by computer science researchers. It also functions as a search engine with several features including autonomous citation indexing, automatic metadata extraction, full-text indexing and reference linking. Users are able to retrieve relevant documents from the CiteSeer^x database directly using search queries and will further benefit if the system suggests document recommendations to the user based on their preferences and search history. Therefore, recommender systems were initially developed and continue to evolve to recommend more relevant documents to the CiteSeer^x users. In this thesis, we introduce the Conceptual, Impact-Based Recommender (CIBR), a hybrid recommender system, derived from the previously implemented conceptual recommender system in CiteSeer^x. The Conceptual recommender system utilized the user's top weighted concepts to recommend relevant documents to the users. Our hybrid recommender system, CIBR, considers the impact factor in addition to the top weighted concepts for generating recommendations for the user. The impact factor of a document is determined by using the author's h-index of the publication. A survey was conducted to evaluate the efficiency of our hybrid system and this study shows that the CIBR system generates more relevant documents as compared to those recommended by the conceptual recommender system.

ACKNOWLEDGEMENTS

First of all, I would like to thank God for giving me strength to complete my Master's and for bestowing me with his abundant blessings, one of which is the opportunity to be able to do my Master's in the University of Arkansas.

I would like to express my profound gratitude to my advisor, Dr. Susan Gauch, Computer Science & Computer Engineering Department. She was very patient with my numerous queries and was of tremendous help & support. She has helped me expand my knowledge. I appreciate that Dr. Brajendra Panda and Dr. Dale R. Thompson had agreed to be part of my Thesis Committee and have taken time out to review my thesis.

I would also like to thank my parents and my husband for being supportive and for helping me make the right decisions. Last but not the least, I thank all my friends for their help and co-operation during my Master’s.

DEDICATION

This thesis is dedicated to my husband and my adorable baby Rhen.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation

One time or the other, all Internet users have come across recommender systems. If a user browses or purchases items from an e-commerce website such as Amazon, one might have encountered with listing “Customers who bought this item also bought the following items”. The system suggesting items to users is a recommender system. Recommender systems (RS) may be defined as software agents that provide suggestions or recommendations for items or documents to the user based on their interests and preferences. In brief, RS make recommendations of unknown items that users might prefer. The recommendations ease the information overload for the user by proactively suggesting relevant items to the users, moving the burden of discovery from the user to the system. Items and documents are used interchangeably throughout the thesis to define objects that are recommended by different RS. Different recommender systems have difference criteria for success that may vary based on the retrieval, recommendation, prediction, or interaction perspective. Recommender systems may be developed to retrieve accurate recommendations, to reduce the ‘cost’ of searching, to predict the ‘likeness’ to an item, to evaluate an item, to inform the users existence of certain items in the database or even to attract users to a domain. To improve the recommendations in future, the quality of the recommendations is evaluated by different methods classified as experimental, quasi-experimental, or non-experimental research designs [1-3].

The design of a recommender system can vary based on the domain characteristics, nature of user feedback and availability of usable data. Multiple techniques may be used to build recommender systems however there are primarily two different approaches - collaborative filtering (CF) and content-based recommender systems. The CF recommender approach is used by major e-commerce websites and is a prominent well-known method that could be adopted in different systems. Here, the community helps the user to obtain recommendations for items. This approach is particularly applicable when the items being recommended do not have much semantic information available. In that case, recommendations are made based on patterns of selection across a wide variety of users rather than based on features of the items themselves. The disadvantages of this system include requirement of integration with other informational database and the availability of a large, active user community.

In contrast, content-based recommender systems do not require a user community and they employ a much transparent approach. In these systems, features of the items themselves (keywords typically) are extracted and used to recommend items to users based on similarities between the items. However, content-based systems also have their disadvantages since the recommendations do not consider external features such as popularity among other customers.

It is always important to evaluate a recommender system to improve future recommendations to the user [3]. With the CiteSeer^x digital library, our objective was to improve the existing conceptual content-based recommender system to provide better recommendations to the users. For this, we developed a recommender system that recommended papers based on the paper authors' impact. We evaluated the conceptual recommender, impact factor recommender, and a hybrid system that combined the two sources of evidence in different proportions. Our specific objective was to determine the combination of the recommender systems that would provide the best recommendations to the user.

1.2. Organization of this Thesis

Chapter two of this thesis provides an overview of the literature review on the main premise of this work. Chapter three discusses the architecture and implementation of the newly designed recommender system. Chapter four discusses the experimental materials, procedures and analysis of our research project. Chapter five provides a brief conclusion discussing the scope of the research and possible avenues of research exploration in the future.

2. REVIEW OF RELATED LITERATURE

This section provides an overview of the different types of recommender systems and a description of the impact factor of authors.

2.1 Impact Factor

As there is a huge repository of publications by numerous authors in different fields, various measures have been developed to rate the importance of each author and their contributions to a particular field. The most commonly used measure is called the h-index, however there have been many variations of this measure developed since it was first introduced.

2.1.1 H-index

Jorge E. Hirsh proposed the h-index to measure the relevance of authors in a particular scientific field by taking into consideration the number of papers the author published and the number of times these papers were cited by other authors [4]. The h-index is not dependent on the amount of contribution the author contributes to a paper. The h-index of an author can be impacted positively by being the co-author of the paper. Every time the paper is cited, the co-authors h-index also increases. The h-index ignores self-citation, as these do not indicate a significant impact to the field. The h-index is a monotonically increasing measure; it never decreases. Even if the authors were to stop publishing, the h-index may continue to increase as previously published papers accrue more citations.

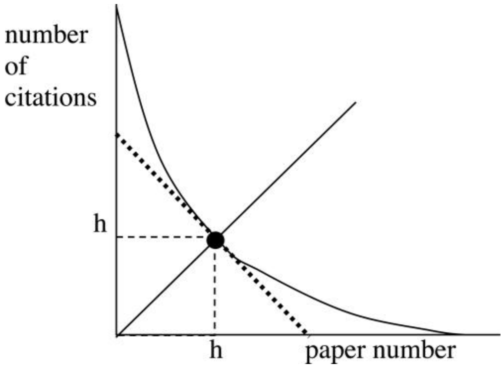


Fig 2.1.1: h-index representation [4]

Fig 2.1.1 is an example of the number of citations received versus the number of papers the author published. The intersection of the 45-degree line with the curve gives h. As per Jorge E. Hirsh, the h-index is explained as follows, ‘A scientist has an index **h** if **h** of his/her $\mathbf{N_p}$ papers have at least **h** citations each, and the other $\mathbf{(N_p - h)}$ papers have no more than **h** citations each [4]. Consider the following example: An h-index of 20 means the researcher has 20 papers each of which has been cited 20+ times. The typical h values can vary in different area of application. The factors that affect h are the number of authors on a typical paper, the number of publication venues for the field, and the typical number of references for a paper in that field. The H-index was intended to evaluate researchers in the same stage in their careers and it is not meant for historical comparison.

There are several drawbacks to the h-index. Since it never decreases, it does not distinguish between currently active authors and those whose contributions are essentially historical. It is also not applicable to new authors in a field who may be publishing excellent work but whose papers have not been around long enough to attract large numbers of citations. To address these issues there have been various variants of the h-index such as g-index, c-index, e-index [5-7]. However, in spite of these issues, the h-index is the most widely used measure of an author’s index and it is the feature that we extract and use in our impact based recommender system.

2.2 Recommender Systems

Today, the World Wide Web and the Internet make it possible for us to access unlimited amounts of information from nearly infinite sources just a click away. The deep web contains about 550 billion individual documents and an additional 2.5 billion documents are estimated to be on the surface web, growing every day at the rate of 7.5 million documents [8]. If the availability of infinite information is not managed effectively, it can lead to an information overload and ultimately result in a reduction of user productivity and decision-making ability [9]. Therefore, it becomes important that new systems are developed to retrieve relevant information with minimal burden on the user.

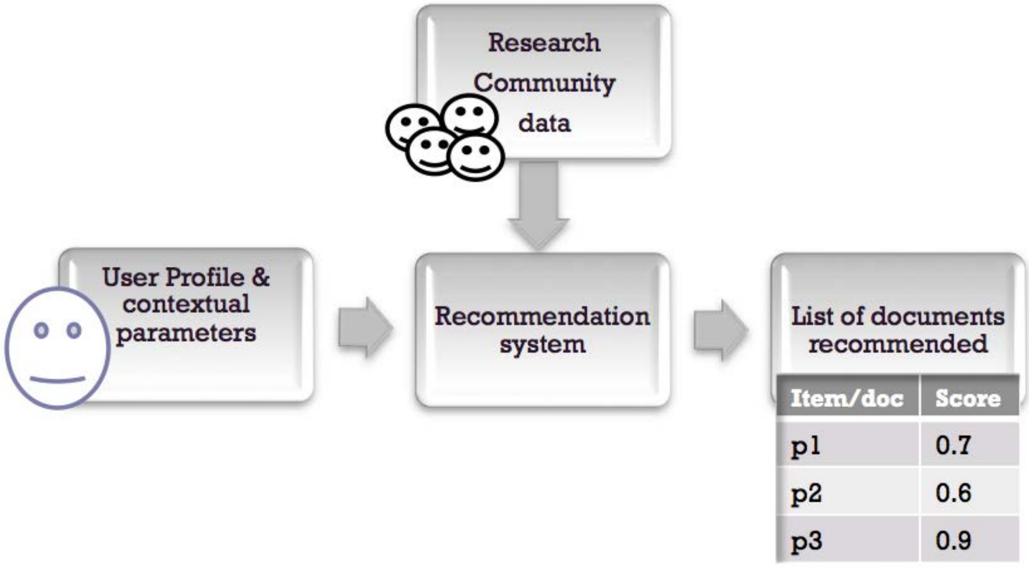
The desire to help users find relevant information from a sea of web pages led to the advent of major search engines such as Yahoo [10], Google [11], and Bing [12]. The algorithms used in these search engines helped the user to retrieve documents that are ranked based on keyword input. The first "all-text" crawler-based search engine, WebCrawler, was developed in 1994 to allow users to search for any word in any webpage [13]. This content-based approach has become the standard for many major search engines along with the speed of information retrieval.

With the continued growth of the Internet, a keyword query alone may not give the most appropriate result for the user. At times, the user may be unsure about the required keyword for yielding the interested results or the user may welcome reading suggestions based on the user's past queries. This privileged demand by the user has led to the development of recommender systems. Here, based on the user's search patterns or/and the search pattern similarity with other users, documents or products are recommended to the user. Thus, recommender systems can be defined as "software tools and techniques providing suggestions for items to be of use to a user" [14]. The different types of recommender systems include collaborative filtering (discussed further in Section 2.2.1), content-based recommendations (discussed further in Section 2.2.2), demographic, utility-based, knowledge-based, hybrid recommender system and of lately, there has been development in mobile recommender systems [14-16]. Recommender systems have become an active area of research since the first papers on collaborative filtering in the mid-1990s [17].

2.2.1 Collaborative Filtering Recommender System

In a collaborative filtering recommender system (CF recommender system), a user is given recommendations based on his/her interests in the past compared to others in the user community. For example, if users A and B have shown strong overlapping interests in publications in the past, then this system will make recommendations to user B based on the new publications chosen by user A. Online retailers such as Amazon, iTunes, Netflix use collaborative filtering as a method to provide recommendations to users; if a user purchases product A, B, C, then other users who purchased product B will also be shown products A and C as recommendations for future purchase.

These recommender systems have several challenges for their implementation: (1) selection of criteria that should be considered to determine overlap between the users; (2) identification of users with overlapping interests; (3) recommending new items that have not generated sufficient user interest in the past but might be very relevant to the user; (4) deciding whether or not overlapping interest on one topic indicates a similarity in interest on another. Therefore, a single method of recommender system might not provide the user with utmost utility to retrieve items of his/her interest with user's minimal effort [2]. Figure 2.2.1 shows the architecture of a prototypical CF recommender system for research articles.


Research Community data
User Profile & contextual parameters
Recommendation system
List of documents recommended

Item/doc	Score
p1	0.7
p2	0.6
p3	0.9

Fig 2.2.1. Collaborative filtering recommender system - “Tell me what is popular among my fellow researchers”. Figure modified from [3]

CF recommender system may use different types of inputs for evaluating user preferences. Ratings can be further classified as implicit and explicit ratings; for the latter, the users have to actively rate a document or item. Even though this burdens the user with the additional trouble of rating items, this may prove to be more accurate. For implicit ratings, the user’s action is simply taken into account and interpreted as rating. For example, if the user searches and observes a document, the system monitors and logs this activity as a positive response by the user. To provide active users with recommendations for documents or products, CF recommender systems use two different entities – users and items [2, 18].

There are multiple approaches for providing recommendations through the collaborative system. In the ‘user-based nearest neighbor recommendation system’, ratings of products or documents by a user is used to provide recommendations to peer or neighboring users. However, this approach assumes that if the users had overlapping preferences in the past, then they will have identical preferences in future, and that the user taste will remain stable overtime. This is a problem when the number of users is very large because this also increases the number of items that needs to be catalogued and the number of neighbors that needs to be monitored to obtain real time predictions for the active user. The selection of the neighbor for the user is also an important step of the process before suitable recommendations can be provided. Only those neighbors that exhibit a positive correlation with the active users past preferences and those who have rated the publications should be selected for the study. The selection is further refined by a threshold where a definite number of nearest neighbors are chosen, considering that the selection should not be too small or too large. Different problems associated with the limit thresholds are discussed in previous studies [19, 20].

Another approach for the collaborative recommender system is the ‘item-based nearest neighbor recommendation’ used in large-scale e-commerce websites such as Amazon. This is particularly suited for large databases and allows offline processing making it possible to provide real time recommendations even for large rating matrices [21]. Here, predictions are computed based on the overlap between the items and not the users. For this approach, an *item similarity matrix* is constructed with up to N^2 entries to describe pairwise similarity of the different catalogued items. Similar to the previous approach, a limit can be established for the ratings and neighbors. This approach may be used to make a prediction for an item ‘p’ for a user ‘u’ by ratings items that are similar to item ‘p’ and by computing the weighted sum of the user’s ratings for similar items.

$$pred(u,p) = \frac{\sum_{i \in ratedItems\,(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItems\,(u)} sim(i,p)} \qquad (1)$$

The above equation may be used to predict the rating for the user ‘u’ for an item ‘p’ [2, 21]. As explained here, the user-based and item-based approaches primarily use the neighborhood method, which emphasize on relationships between items and users.

Although CF systems were primarily developed to recommend items that did not have semantically-based features available, McNee et al. [22] has explored the CF system for recommending research papers by creating the ratings matrix using citation web between the research papers. The authors compared six different algorithms for obtaining additional references for citing in a target manuscript. In this project, the reference citations were selected from a database of 186,000 documents in ResearchIndex using offline and online experiments. The six approaches used included co-citation matching, user-item collaborative filtering, item-item collaborative filtering, naïve Bayesian classifier, localized citation graph search and keyword search. Results from the online study suggested that the users were enthusiastic about receiving recommendations from the domain and felt that the recommendations were of high quality. The offline experiment indicated that there were large differences in accuracy in recommending citations for the different algorithms, especially for citation coverage.

Recent approaches for the collaborative filtering recommendation systems include less mathematically complex methods such as the Slope One prediction scheme that provide recommendations with reasonable reliability [23]. Google news personalizes news for each user by a slightly different method; here, a combination of model and memory based approach is used [2]. Several examples of model based, memory based and hybrid recommenders used in CF recommender systems are discussed with their advantages and shortcomings in previous literature [24].

Overall, collaborative recommender systems are reasonably robust; however, this cannot be applied to every system. For example, CF recommender system may not be used when a system is recently developed because these systems do not have any history of user preferences and thus cannot provide reliable ratings. Specifically, without many users there are fewer ratings and with fewer ratings, recommendations are not generated effectively. Therefore, collaborative filtering is used with preexisting data is available to generate reliable ratings and nothing other than the ratings are required for the CF recommender systems. This is a well-known “cold-start” problem experienced by all CF recommender systems.

2.2.2 Content-based Recommender Systems

If we know that the user A prefers item ‘p’, then it would be easy for us to recommend items similar to ‘p’ to user A without requiring information about what other users are interested in. This, in a nutshell, is how a content-based recommender system differs from collaborative filtering recommender systems. A content-based recommender system recommends items by primarily tracking two specifics - the user profile based on his/her past preferences and the characteristics of the items that the user likes. There are different steps adopted to categorize items based on their characteristics and to automatically learn the user profile for making recommendations [1, 2].

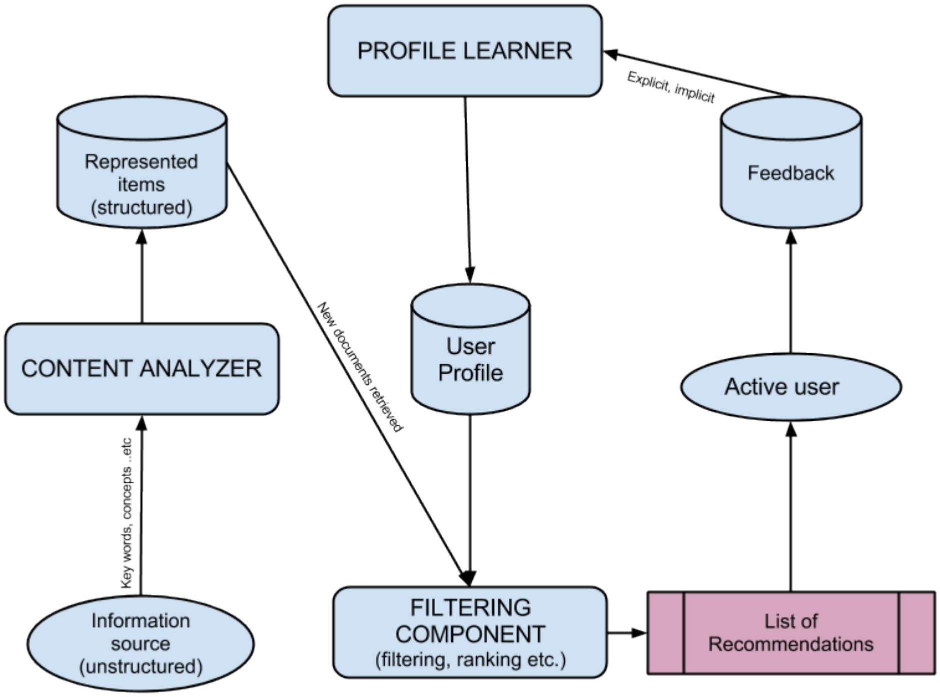

PROFILE LEARNER
Explicit, implicit
Represented items (structured)
Feedback
New documents retrieved
User Profile
CONTENT ANALYZER
Active user
Key words, concepts ..etc
Information source (unstructured)
FILTERING COMPONENT (filtering, ranking etc.)
List of Recommendations

Fig 2.2.2. A general architecture of a content-based recommender system - “Show me similar documents to what I have liked before”. Figure adapted from [25]

The general architecture of a content-based system, as shown in Figure 2.2.2, includes multiple components. The primary components are: (1) A *content analyzer* that extracts relevant keyword information from its information source, i.e., unstructured data in documents as a pre-processing step and prepares the document for subsequent steps such as learning and filtering; (2) a *Profile learner* module that extracts information from previous user preference data i.e., feedback from the user, and generalizes it through machine learning techniques to construct a user profile; (3) a *Filtering component* module that examines the user profile and compares the user preferences with the information from the document pool to make relevant recommendations to the active user. The feedback from the user about the recommendations can also be further used to improve the user profile to generate better recommendations in future.

The content of the represented items in a processed document may vary depending on how the attributes are assigned to unprocessed data and how information is retrieved. The unprocessed documents must be structured to reduce ambiguity caused by polysemy and synonymy, where a single word may have different meanings and multiple words may have similar meanings. These ambiguities may lead to the omission of relevant information or assigning relevance to non-relevant items. Using semantic analysis, these errors are mitigated to some extent by simply cataloging item characteristics in detail through content representation. For example, for a research paper recommender, publication characteristics such as author name, title, publisher, keywords, category etc. could be stored in the database to be used later to provide recommendations to the user. Keywords are assigned to represent documents using lexicons, ontologies or knowledge bases. Therefore, the *content analyzer* step processes the documents to a structured form that could be readily and efficiently accepted by the *filtering component*.

For the *filtering component* to provide recommendations it needs a robust and evolving user profile that is created after the *profile learner* assesses the feedback from the user based on their previous experiences. Initially, it is not essential that the user profile be created using the feedback; rather, it could be created simply by direct user input, i.e., if the user provides 'preferences' or 'areas of research interest' while setting up the profile. However, if this is not the case, the user may provide two types of feedback based on their previous choices– explicit or implicit feedback. Explicit feedback includes like/dislike statements, ratings provided by the user and/or text comments. Implicit feedback may include the monitoring of user behavior such as their search activity and/or clicking of documents. Based on this feedback, the *profile learner* creates new categories for the user and/or adds or reduces weight to keywords to develop a machine learned user profile. Feedback from the user may change over time, so this information is continuously updated to the *profile learner* and further allows understanding of the user preference dynamics.

If new items are available from the information source or document pool, the *filtering component* will compare the new documents with that of the available information in the user profile to assess if they should be recommended to the user. If there are numerous new items available, the *filtering component* uses an appropriate strategy to rank these items based on relevance. The *filtering component* assesses and categorizes the newly available documents using either basic keyword matching or by building vector space models (VSM) with TF-IDF (term frequency inverse document frequency) weighting.

For keyword matching, the user rates the items he/she likes and prepares a set of keywords based on this. Keywords used to compare the documents may not be specifically assigned ‘keywords’; they could be the document titles, contents of the documents, or any other characteristic. From the structured document pool, the system retrieves keywords to compare between the known and unknown documents. In the commonly used vector space approach, keywords extracted from the documents are weighted using the TF-IDF method (term frequency times inverse document frequency). The vector space model represents text documents as vectors of keyword weights in a multidimensional space, with one dimension for every unique keyword in the document collection. *Term frequency* assesses the importance of words in a document by measuring how often they appear in the document after considering appropriate normalizations to account for variability in document length. For example, the normalization for the frequency of terms is calculated through the equation as described in [26] as,

$$TF\ (i,j) = \frac{freq\ (i,j)}{maxOthers\ (i,j)}$$

Here, *freq(i,j)* is the total frequency of the keyword ‘i’ in document ‘j’ and *maxOthers* is the maximum frequency of the other keywords. Another parameter, *inverse document frequency* (IDF) is also calculated for the TF-IDF approach. This second section reduces weight on those keywords that commonly occur across several documents and this helps to remove the non-specific keywords for document retrieval. The IDF is calculated as,

$$IDF\ (i) = \log \frac{N}{n(i)}$$

Here, *N* is the number of all documents that could be recommended and *n(i)* is the number of documents among the *N* documents that has the keyword ‘i’. The final TF-IDF equation used to estimate the weight of a keyword ‘i’ in a document ‘j’ is as follows,

$$TF - IDF\ (i,j) = TF(i,j) * \ IDF(i)$$

After correctly representing the content and characterizing the keywords for assessing similarity in several documents or items, the system needs to retrieve or recommend documents based on this similarity. This is known as *Similarity Based Retrieval*. For the system to make recommendations, it should now evaluate how much an unknown document or item relates to the documents that the user has liked in the past. This similarity or likeness of the known document to the unknown document is estimated by different methods. For example, the Cosine similarity method measures the similarity and the cosine of two vectors can be estimated as follows:

$$The\ cosine\ similarity, Cos\ \theta = \frac{A\,.\,B}{\parallel A \parallel \parallel B \parallel}$$

Here, A and B are vectors of two attributes. A cosine similarity of -1 represents completely dissimilar value, whereas, 0 and 1 represents fully independent and completely similar values respectively. For information retrieval, the cosine similarity values may vary between 0 and 1, because the TF-IDF values are always positive.

Keyword-based methods for content-based recommendations

Once the user has provided implicit or explicit feedback in the form of a set of documents with indications of whether or not each document is interesting to them, the system must recommend new documents to the user based on the information provided. As mentioned previously, the user can directly provide sets of liked documents through a survey (explicit feedback) or they could be automatically deduced by the system by observing user behavior. From these documents, a user profile is created. Thus, the user's interests are represented as a set of keyword vectors, one per document in which the user has previously expressed interest. Several popular approaches to recommend documents based on the user profile are summarized below.

a) Nearest neighbors: In this approach, all document vectors in the collection are compared to all document vectors in the user profile vector using an appropriate similarity calculation function. [27]. The collection documents are then ranked by their maximum similarity to any document in the profile and the most similar *k* documents are recommended to the user. This is known as the *k nearest neighbor method* (*kNN*) and can be completed with different variations such as changing the size of *k*, weighting the profile documents differently based on user rating values, and considering thresholds for similarities. Based on the type of data, the similarity function used by *kNN* can differ. If the data is structured, a Euclidian distance metric is used and a cosine similarity measure is used if the data is unstructured [17, 28, 29]. The *kNN* method is used in Daily Learner [30] and Quickstep [31] systems. The Daily Learner is a learning agent for wireless news access devices that recommend relevant daily news stories to users based on user feedback. Quickstep uses an ontological approach to recommend academic research papers after creating a user profile and obtaining relevant user feedback.

b) Rocchio's algorithm: Rocchio's algorithm [32] is similar to kNN with the key difference being how the user profile is represented. In Rocchio's algorithm, the user profile keyword vectors are combined, typically by simple addition, to create a single profile that represents the user profile. Essentially, this profile vector represents the aggregate of all user interests. With this approach, each document in the collection need only be compared to one vector of user interests, so it is much more efficient. The documents are then ranked by similarity and the most similar documents are recommended to the user. This relevance feedback algorithm is used in several content-based recommender systems [1]. For example, Fab [33] is a recommendation system for the web and represents files with words having the greatest TF-IDF weights and with these words appearing frequently in a single file, but infrequently in the whole document pool.

The TF-IDF vector of the files with the greatest weights is detected by Fab using Rocchio's algorithm. Another example is YourNews [34], an adaptive personalized news delivery system that allows the users to view and edit their profiles for relevant news recommendations.

c) <u>Probabilistic methods and Naïve Bayes</u>: Some recommender systems base their recommendations on probabilistic, rather than vector space, models. This leads to a slight modification in both the term weighting scheme and the similarity calculation function. As in the vector space model, users interests and documents in the collection are represented by a set of weighted keywords, but the similarity function between these keyword sets is calculated using the Naïve Bayes model wherein the probability that a document 'd' belongs to class 'c', P (c|d) is calculated as,

$$P(c|d) = \frac{P(c)\,P(d|c)}{P(d)}$$

Here, P(c) is the probability of observing a document in class 'c'; P(d|c) is the probability of observing document 'd' when 'c' class is present; and P(d) is the probability of observing the document 'd'. The document 'd' is categorized in the class with the highest probability and is chosen by using the equation,

$$c = argmax_{cj} \frac{P\left(c_j\right) P\,(d|c_j)}{P\,(d)}$$

The naïve Bayes method is used in different content-based recommender systems [1]. For example, Syskill & Webert [35] is a software agent that rates websites and provides recommendations to the user based on the three point rating system by the user and the webpages the user clicks. A study evaluated six different algorithms for Syskill & Webert and determined that the naïve Bayesian classifier provides the best option for the system [35]. Another example is News Dude [36], an intelligent personal news agent that compiles daily news and tailors it to user preference based on their feedback. News Dude computes predictions for news stories for the user based on a long-term model that uses the probabilistic learning algorithm, the naïve Bayesian classifier.

Conceptual methods for content-based recommendations

In conceptual approaches, the user profile and the documents are represented as vectors of weighted concepts rather than vectors of raw keywords. This approach has the advantage of creating vectors with much lower dimensionality (the number of concepts in the ontology rather than the number of unique words in the collection). It also is able to handle problems with synonymy much better since multiple word forms all map to the same concept.

A conceptual content-based recommender system was recently developed to recommend research papers based on the user profile and the CiteSeer^x classified documents [37]. CiteSeer^x is a scientific digital library and search engine that has a collection of technical papers focused primarily on computer science. CiteSeer^x provides citation indexing and links using a method of autonomous citation indexing [38]. The current project is an extension of previous research by Puthiyaveetil [39] and is focused to improve the recommendations given to the user by considering an additional parameter, the author's h-index for documents. A brief summary of the conceptual content-based recommender system previously published is provided below.
The ACM (Association for Computer Machinery) classification tree, consisting of 369 categories in three levels, was used to represent the 1,834,852 documents in the University of Arkansas's CiteSeer^x collection. There were 55,526 documents that had been explicitly tagged by the authors with ACM concepts identifiers. These were used to train a kNN classifier for the ACM concepts and the remaining untagged documents were then automatically classified to identify the appropriate ACM concepts for those documents. The documents and their associated

concepts and concept weights are stored in the CiteSeer^x database to be used for browsing, building the user profiles, and making recommendations.

The user creates a profile by creating an account, logging in, and searching for papers to read. The ACM concepts associated with the user-viewed documents were accumulated to create the user profile. Essentially, this is a Rocchio approach but, instead of accumulating keyword vectors to build a user profile, we accumulate concept vectors to create a single, conceptual, user profile vector. Based on empirical results by Puthiyaveetil [39], the recommender system uses only the most highly weighted three concepts from each document's vector when calculating the profile/document similarity in order to make content-based recommendations. The rationale is that most documents are closely related to no more than three ACM concepts and that the other non-zero concepts in the document vector are more likely to introduce noise. The most similar documents to the user profile, based on the cosine similarity measure, are recommended to the user.

Chandrasekaran et al. [40] developed an algorithm to recommend documents for authors having publications in CiteSeer^x. A study conducted using eight of the authors suggested that majority of the preferred recommendations used 10 concepts from their user profiles. Another more recent study, focused on recommending documents to all CiteSeer^x users who were not published authors. This study by Puthiyaveetil et al. [37] conducted a series of experiments to determine the number of concepts from the user profile to use during the similarity calculation to produce the best recommendations. Document recommendations were generated for the top 3, 6, 9, and 12 concepts of the user profile, with the top 5 documents recommended for each approach presented to the user for evaluation. Using a subset of 1,000,000 documents and seven volunteer users, the results confirmed that conceptual content-based recommender system generated preferred results using only three concepts from the user profile and three concepts from the document vector [37].

2.2.3 Hybrid Recommender Systems

There are several advantages of using content-based recommender system rather than a collaborative filtering recommender system, where possible. The former has user independence, allowing the active user to develop their own user profile based on their preferences; whereas, CF recommender system require ratings from the community to generate recommendations. The CF recommender system has numerous 'black boxes' - the unknown users based on which recommendations are provided to the user. In contrast, a content-based system is much more transparent since it is clear which documents and attributes are used to generate the list of recommendations. Another limitation of the CF recommenders is that the system needs previous information from the community, but this is a problem if the item is new; content-based systems are able to overcome this limitation by the user's preference for similar items.

Content-based recommender systems also have certain disadvantages when compared to CF recommender systems. Content-based recommender systems can become over-specialized, recommending similar items over and over again, with no innate method of retrieving something that is completely unexpected; this is called the serendipity problem. Also, if the user is new and has not provided sufficient information to form a robust user profile, the content-based system may not be able to provide accurate recommendations.

To address the limitations of each of the two approaches, it is possible to combine them to form a hybrid recommender system. For example, in the *content-boosted collaborative filtering* hybrid approach developed by Melville et al., the content based recommendations are used to enhance the user profile and after the user data boost, collaborative filtering is used to create personalized recommendations [41]. REFEREE, developed by Cosley et al., is an open framework for building hybrid recommender systems and testing them using the ResearchIndex database [42]. Here, the content-based system is used to retrieve a set of documents from which recommendations are generated and ranked based on CF system. A third recommender system example is the use of Boltzmann machine proposed by Gunawardana and Meek, a probabilistic model that combine both content-based and CF information coherently [43]. Information from both systems are coherently encoded as features and uniformly used to assign weights to the features to learn how correctly these features predict user actions. They have applied this approach to recommending entertainment and shopping items with improved success over collaborative or content-based recommendation alone.

Several of these examples suggest that based on the study system under consideration, the information retrieval and recommendation system adopted can be customized as required. Our work can be considered a type of hybrid system. The original conceptual recommender system builds user profiles based on the contents of the documents; it is a content-based recommender system. However, the impact factor calculation is based on citations to other authors. One can consider citations to documents as a form of community feedback and, by exploiting that information; we are incorporating a collaborative filtering recommender, ultimately ending up with a hybrid system.

3. RESEARCH DESIGN

The Conceptual, Impact-Based Recommender System is a combination of the Conceptual Recommender System and the Impact-Based Recommender System. The Conceptual Recommender System is a model based system because it initially creates a user profile and then recommends papers to the user based on their user profile [37]. The documents recommended to the user are conceptually correct, but from the user's perspective the documents are not always relevant. Our aim is to consider the importance of the document and combine it with the conceptual recommendations to help improve the relevance of the documents recommended to the user.

The user can search for specific documents in CiteSeer^x and select the documents that are relevant to the user. This selection of documents is tracked and used to create a profile for the user. The user profile displays the concepts in a hierarchical structure based on the relevance of the document. The user can modify the profile to remove those concepts that are of no further interest to the user. This functionality aids to improve the user profile.

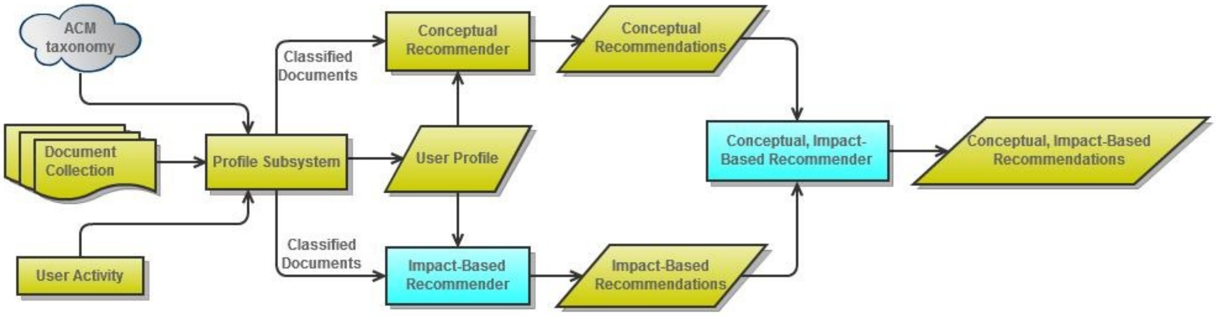


Figure 3: System Architecture of CiteSeer^x

The diagrammatic representation above is an extension of the Conceptual Recommender System, developed by Ajith Kodakateri Pudhiyaveetil, as part of his MS thesis [39]. The original system consists of two major components:

1) Profile Subsystem

2) Recommender

This work was extended to include the components of Impact-Based Recommender and Conceptual, Impact-Based Recommender.

3.1 Profile Subsystem

In this section we explain how the user profile is generated in CiteSeer^x by using the Classifier and Profiler components.

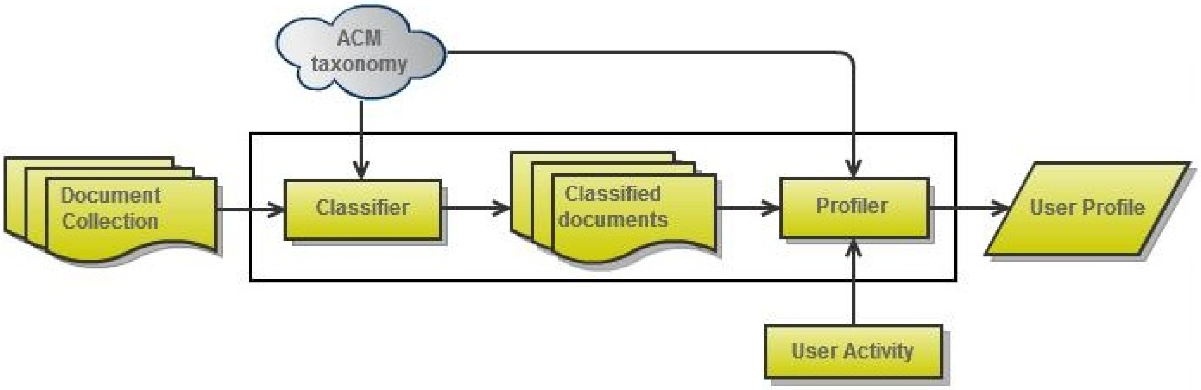


Figure 3.1: Profile Subsystem of CiteSeer^x

3.1.1 Classifier

Documents in the CiteSeerx database are classified into a set of predefined concepts. These concepts are obtained from the ACM's Computing Classification System (CCS). This 3-level deep hierarchical set of concepts contains a total of 369 concepts. In Figure 3.1.1, we show the ACM Taxonomy with a subset of the concepts. The concept 'Data' is further described, and used, in section 3.1.2.

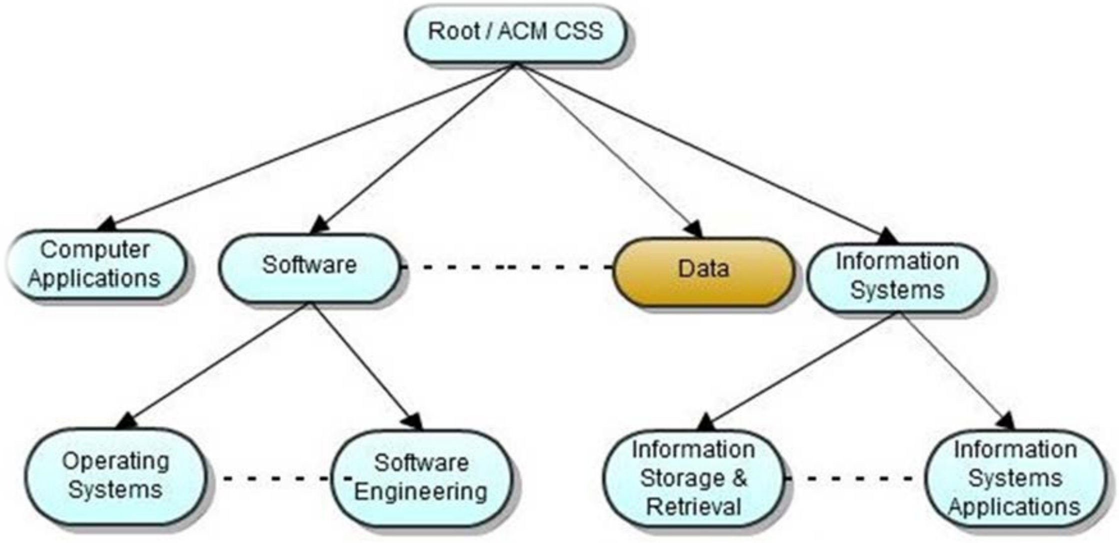


Figure 3.1.1: ACM Taxonomy

The classification of the documents is done in two stages.

1. Training stage: From our collection of documents, we parsed 1,834,852 text documents and found that 55,526 of these documents have author-assigned ACM tags. These documents were used as training data for the KNN classifier. For each concept in the CCS, we randomly selected 18 documents tagged by the authors as belonging to a concept. Concepts that had fewer than 18 candidate documents were ignored by the training algorithm and left us with a classifier that trained on 291 total concepts.

2. Classification stage: The non-tagged documents were then classified using the k-nearest neighbor algorithm. The top 10 concept matches and their similarity weights returned by the KNN classifier for each document in the collection are stored in the CiteSeer^x database. Both non-tagged and tagged documents are stored to the database. This database is used for the user profile and the recommender system.

3.1.2 Profiler

The main objective of the Profiler module is to create a user profile for the users in CiteSeer^x. The inputs to the Profiler module are the ACM taxonomy (refer to section 3.1.1), classified documents (refer to section 3.1.1) and the user activity. Each user's activity is represented by the documents the user viewed/clicked and the amount of time the user spent on the respective documents. These documents are displayed based on the queries that the user enters in the CiteSeer^x search engine. Consider the expansion of the "Data" concept from the ACM taxonomy in Figure 3.1.1 shown in Fig 3.1.2.a.

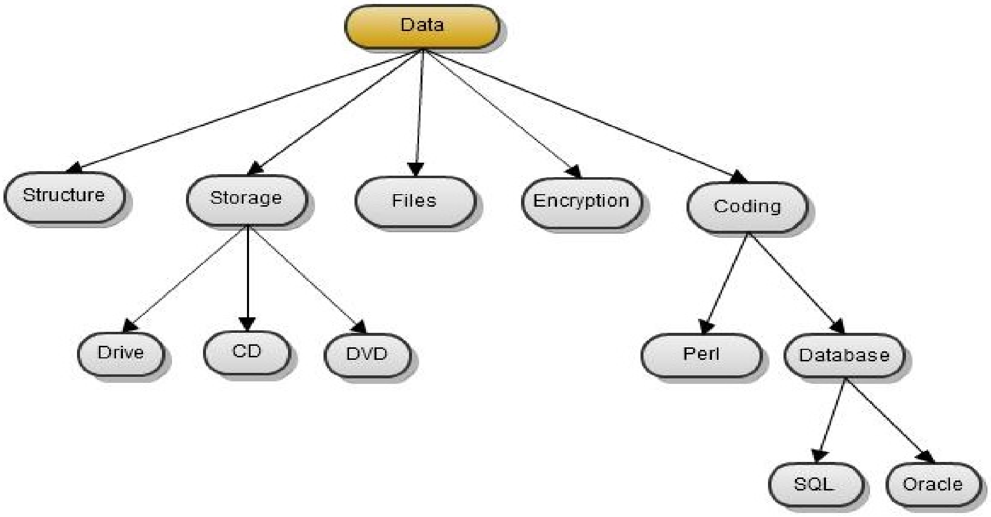


Figure 3.1.2.a: Concept 'Data' and sub levels

Assuming the user clicked on three documents, 'Ease of Coding', 'Types of Storage of Data' and 'Programming Languages pros and cons' respectively. The weights of the concepts for each of the documents are displayed in the table below (Figure 3.1.2.b) and the aggregated weights are derived as follows.

Concepts	*Doc1: Ease of Coding*	*Doc2: Types and storage of Data*	*Doc3: Programming Languages pros and cons*	*Aggregated Weight*
Data		0.16		1.23

Structure	0.1			0.1
Storage		0.06		0.16
Drive		0.04		0.04
CD		0.05		0.05
DVD		0.01		0.01
Files		0.12		0.12
Encryption	0.11			0.11
Coding	0.09	0.03	0.06	0.48 (0.18+0.14+0.16)
Perl	0.04		0.1	0.14
Database			0.12	0.16 (0.12+0.03+0.01)
SQL			0.03	0.03
Oracle			0.01	0.01

Table 3.1.2.b: User's Aggregated Concept Weights

Here, the concept 'Coding' is in all three documents with different weights. These weights are accumulated to calculate the final weight associated with the concept 'Coding'. The above table shows that the user is interested in the category 'Data' and more specifically into the concept 'Coding' which has 48% of the total. Thus, the final output of the Profiler module is a weighted tree of the list of ACM concepts. These concepts represent the user's areas of interest. Figure 3.1.2.c shows a snippet of a user's profile.

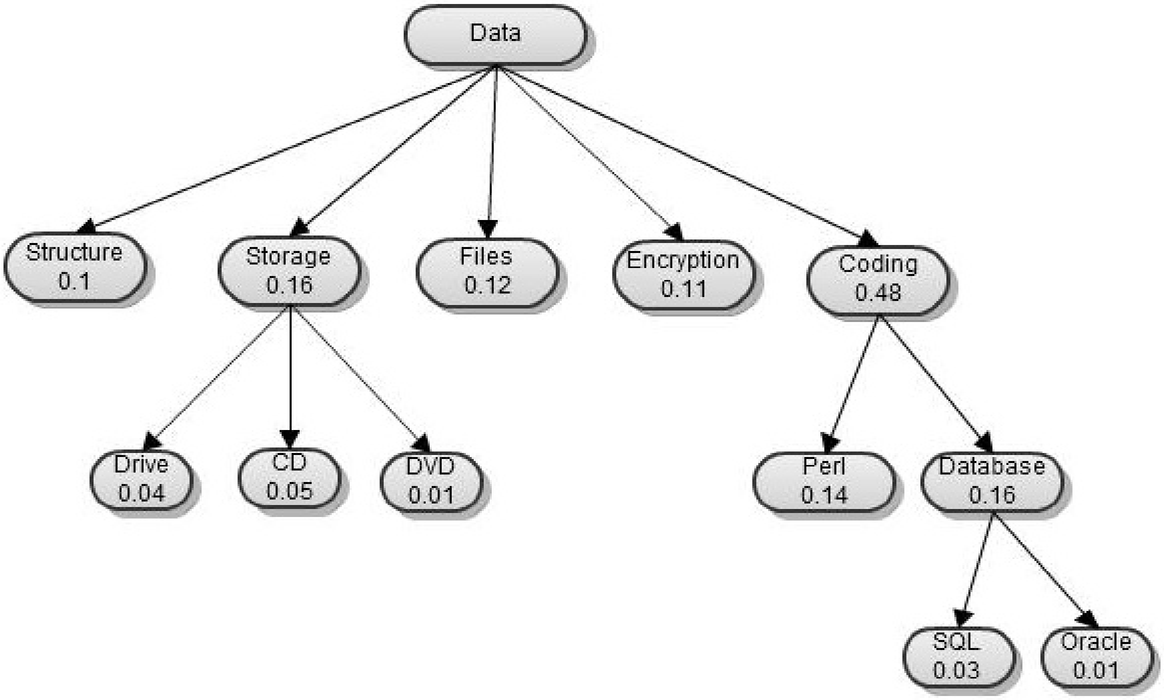


Figure 3.1.2.c: Conceptual User Profile

We recommend papers to the users using this profile. The user can improve their profile by modifying the relevance of the concepts and by deleting irrelevant concepts. The user can also view their profile in a hierarchical structure to view other related concepts in their area of interest.

3.2 Recommender Systems

This module recommends documents to the user using the Conceptual, Impact-Based Recommender System (CIBR). The CIBR system is developed by analyzing the data from both the Conceptual Recommender System and Impact-based Recommender System.

3.2.1 Conceptual Recommender System

The user profile and the classified documents are used to generate the Conceptual Recommender System. This recommender system categorizes documents based on the user's area of interest.

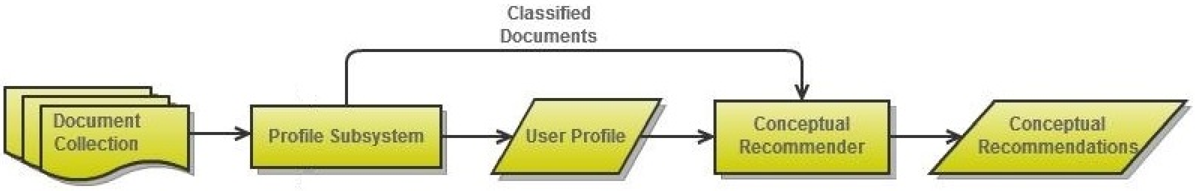


Figure 3.2.1: Conceptual Recommender System of CiteSeer^x

In the Profiler module (refer section 3.1.2), we calculated the user's areas of interest with respect to the ACM CCS concepts. In the recommender module, the user's top three concepts are used to retrieve relevant documents from the CiteSeer^x database. Figure 3.2.1.a shows the weight of the concepts 'Coding', 'Storage' and 'Files' for the documents 'Science Digital Library' 'Structures of video storage' and 'Tool for engineering privacy' in the CiteSeer^x database.

Documents	Coding	Storage	Files
Science Digital Library	0.54	-	0.45
Structure of video storage	0.23	0.12	-
Tools for software privacy	0.63	0.11	0.45

Table 3.2.1.a Document Concept Weights in the CiteSeer^x Database

The weights of the retrieved documents are multiplied with the weights of the user's profile concepts to get the weight *WtDoc* as per, Speretta, M. and S. Gauch (2005). The conceptual match between the document concepts and the user concepts are calculated by using the cosine similarity function [44]

$$\mathrm{WtDoc}(\mathrm{user}_i, \mathrm{doc}_j) = \mathrm{cwt}_{ik} * \mathrm{cwt}_{jk}$$

where,

$$\mathrm{cwt}_{ik} = \text{weight of concept}_k \text{ in userprofile}_i$$

$$\mathrm{cwt}_{jk} = \text{weight of concept}_k \text{ in document}_j$$

$$N = 3 \text{ (Number of concepts)}$$

$$\text{SumDoc} = \sum_{k=1}^{N} \text{WtDoc(user, doc)}$$

The retrieved documents and weight *WtDoc* are added to a collection set *RecList*. The weights of duplicate documents are aggregated to get *SumDoc*. The documents in *RecList* are sorted in descending order of weight. Figure 3.2.1.b shows the multiplied weights (*WtDoc*) and the aggregated weights (*SumDoc*) of the documents from the CiteSeer^x Database.

RecList → *Concepts*↓	Science Digital Library	Structure of video storage	Tools for software privacy
Coding	0.26 (0.54*0.48)	0.11 (0.23*0.48)	0.30 (0.63*0.48)
Storage	-	0.02 (0.12*0.16)	0.02 (0.11*0.16)
Files	0.05 (0.45*0.12)	-	-
SumDoc →	**0.31**	**0.13**	**0.32**

Table 3.2.1.b Document Weights *WtDoc* and *SumDoc*

The above documents will be displayed to the user in their decreasing weights. The document ‘Tools for software privacy’ would be the top recommended document to the user followed by the document ‘Science Digital Library’ and ‘Structure of video storage’ respectively.

3.2.2 Impact-Based Recommender System

The user profile, the classified documents, and the document’s impact factor are used by the Impact-Based Recommender System. This recommender system ranks documents based on the reputation of the document’s authors as measured by the authors’ h-index values. An author has an index h if h of his/her N_p documents have at least h citations each, and the other (N_p – h) documents have no more than h citations each [4]. The impact factor for a document is calculated by finding the h-index value for all the authors of the document and then choosing the highest h-index value. Thus, a document’s h-index is set to that of its most-cited author.

The CiteSeer^x database has a large set of static documents. The impact factor was pre-calculated for all the documents in the database and stored the values into the CiteSeer^x database.

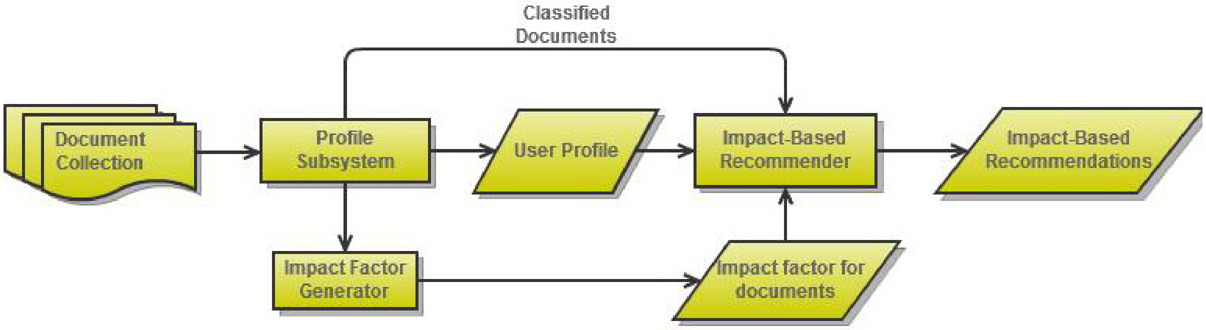


Figure 3.2.2: Impact-Based Recommender System of CiteSeer^x

Continuing with our example, in the Profiler module (refer section 3.1.2), we saw that the user's top 3 concepts were 'Coding', 'Storage' and 'Files'. Based on the top 3 concepts, we retrieve documents from the CiteSeer^x database and add them to the collection set *RecList*. We find the impact factor of all the documents in *RecList* and sort them in descending order. These documents are added to the collection set *ImpactList*. In the conceptual recommender module (refer section 3.2.1) we saw that the concepts returned documents 'Science Digital Library', 'Structures of video storage' and 'Tools for engineering privacy'. In this module, we find the impact factor of these 3 documents.

Science Digital Library			
Luke James		Maria N.	
Publications	#Cited	Publication	#Cited
PaperXYZ	40	PaperRST	0
PaperSDE	13		
PaperRST	0		
H-index	2	H-index	0

Table 3.2.2.a. Publications and Citations of the Authors of 'Science Digital Library'

In Figure 3.2.2.a, 'Luke James' has 3 publications (PaperXYZ, PaperSDE and PaperRST) . As per Jorge E. Hirsch [4], A scientist has index h if h of his/her N papers have at least h citations each, and the other (N - h) papers have no more than h citations each. Since ‘Luke James’ has 2 publications with citations more than and equal to 2, the h-index of “Luke James” is 2. Since Maria N. has a publication but no citation, her h-index is 0. Thus, we consider the highest h-index author, ‘Luke James’, and so the paper ‘Science Digital Library’ has an impact factor of 2.

Tools for software privacy					
Dianne L.		Sarah Tim Lee		Henry Tobit	
Publications	#Cited	Publication	#Cited	Publications	#Cited
PaperABC	2	PaperABC	2	PaperEE	54
PaperMNO	2			PaperHH	3
				PaperGG	4
H-index	2	H-index	1	H-index	3

Table 3.2.2.b Publications and Citations of the Authors of ‘Tools for software privacy’

In Figure 3.2.2.b. the h-index of ‘Dianne L.’ is 2 because the author has 2 publications with at least two citations each. The h-index of ‘Sarah Tim Lee’ is 1. The h-index of ‘Henry Tobit’ is 3. Thus, the impact factor for “Tools for software privacy” is 3.

Structure of video storage	
Timothy Prescent	
Publications	#Cited
PaperA1	30
PaperB2	27
PaperC3	15
PaperD4	14
PaperE5	14
PaperF6	10
PaperF7	9
PaperF8	3
H-index	7

Table 3.2.2.c Publications and Citations of the Author of ‘Structure of Video Storage’

In Figure 3.2.2.c. the impact factor of ‘Timothy Prescent’ is 7 because the author has 8 publications of which 7 publications have at least seven citations each.

	Coding	Storage	Files	Impact

				Factor
Science Digital Library	0.54	-	0.45	2
Structure of video storage	0.23	0.12	-	7
Tools for software privacy	0.63	0.11	0.45	3

Table 3.2.2.d Documents with Category 'coding' in the CiteSeer^x Database and the Impact Factor

In Figure 3.2.2.d shows the impact factor for the documents. The weights of the categories in the documents are not considered. The documents are sorted in descending order of impact factor and displayed to the user. Here the document "Structure of video storage" would be the top recommended document to the user followed by 'Tools for software privacy" and "Science Digital Library".

3.2.3 Conceptual, Impact-Based Recommender System

The Conceptual Recommender System and the Impact-Based Recommender System are combined together to generate the Conceptual, Impact-based Recommender System. In this system, the conceptual documents are generated and re-arranged as per the impact factor to get documents that are more relevant and that are from prominent authors.

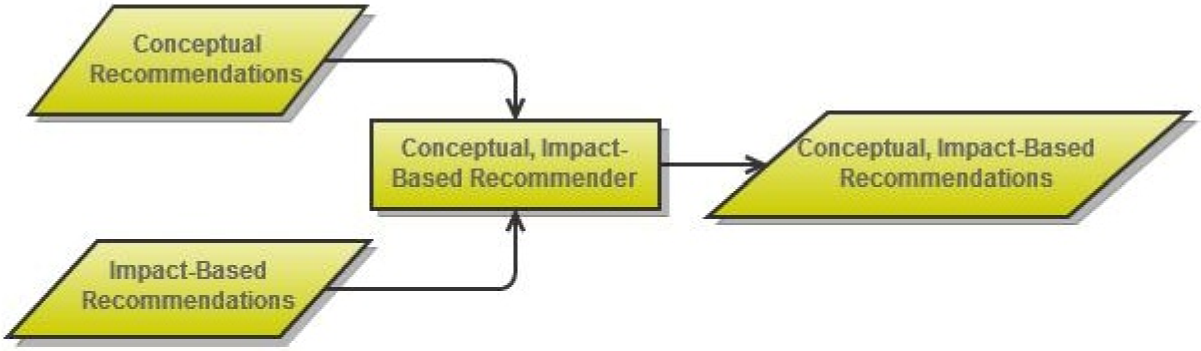

Conceptual Recommendations
Conceptual, Impact-Based Recommender
Conceptual, Impact-Based Recommendations
Impact-Based Recommendations

Figure 3.5: Conceptual, Impact-Based Recommender System of CiteSeer^x

In the evaluation section, we generate the conceptual and impact factor documents. We rank the conceptual documents as per concept weight and add them to a collection set *ConceptList*. We rank the impact-based documents as per impact factor and add them to a collection set *ImpactList*. Both *ConceptList* and *ImpactList* will have normalized values that vary from 0 to 1. The Conceptual, Impact-Based Recommender System (CIBR) uses the *ConceptList* and *ImpactList* and is determined by the formula below

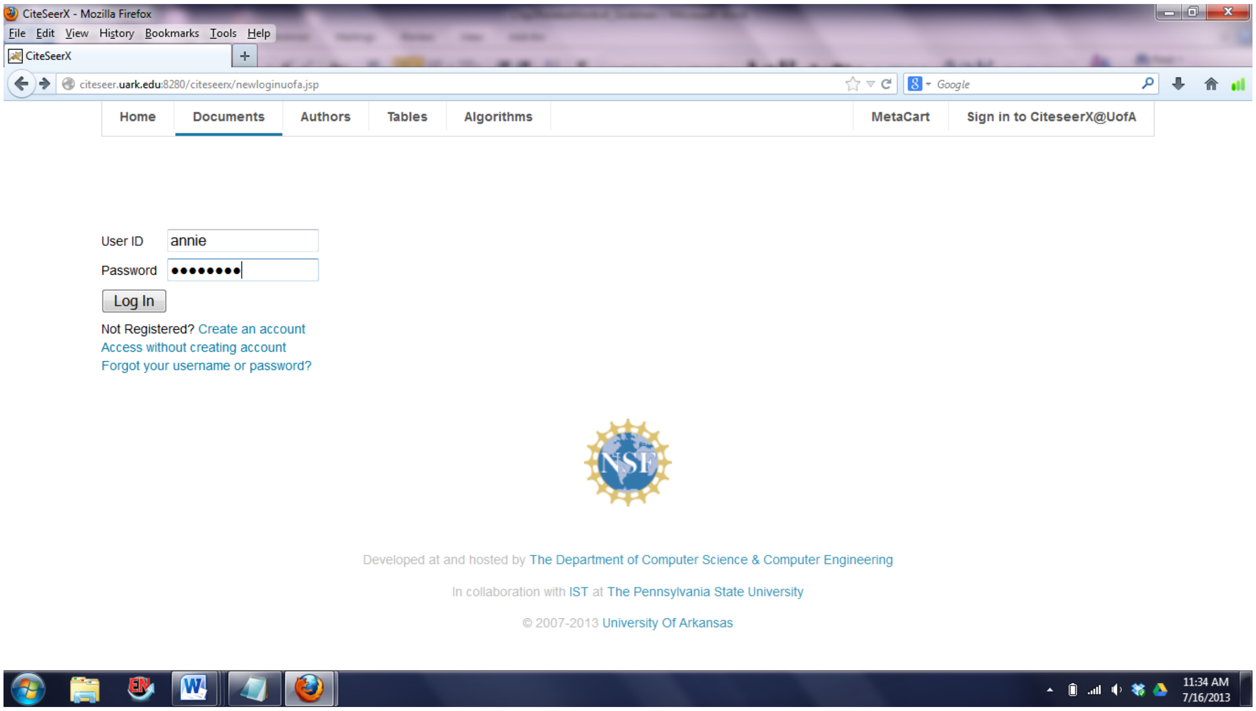
$$\text{CIBR} = \alpha * \textit{ConceptList} + (1- \alpha) * \textit{ImpactList}$$

We calculated CIBR, by varying α from 0 to 1. With α=0 being a purely impact based recommender system and α=1 being a purely Conceptual Recommender System. When α is 0.5, the concept match and the impact match count equally.

4. EXPERIMENTAL EVALUATION

The objective of the study was to compare the effectiveness of the impact-based recommendations versus the Conceptual Recommender System verus combining the two approaches. The survey included a total of 15 volunteers including student and faculty members from the computer science and computer engineering department of the University of Arkansas, Fayetteville. The survey participants interacted with the CiteSeer^x, conducting searches and reviewing results related to their research interests. Thes actions created a user profile for them that was then used to generate a mixture of impact-based and concept-based recommended documents. The participants rated the relavance of the recommended documents, presented in random order, based on their respective interests. The data collected from the participants was then analyzed to determine the best combination of impact-based versus concept-based factors in generating recommendations.

In order to complete the survey, users were requested to create a username and password. The user information was used to record the search history and track the users profile concepts such as Data, Storage, Files etc as explained in section 3.1.2,. Figure 4.a shows the login webpage of CiteSeer^x.


CiteSeerX - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CiteSeerX
citeseer.uark.edu:8280/citeseerx/newloginuofa.jsp
Google
Home
Documents
Authors
Tables
Algorithms
MetaCart
Sign in to CiteseerX@UofA
Log In
NSF
11:34 AM
7/16/2013

User ID annie

Password ••••••••

Not Registered? Create an account
Access without creating account
Forgot your username or password?

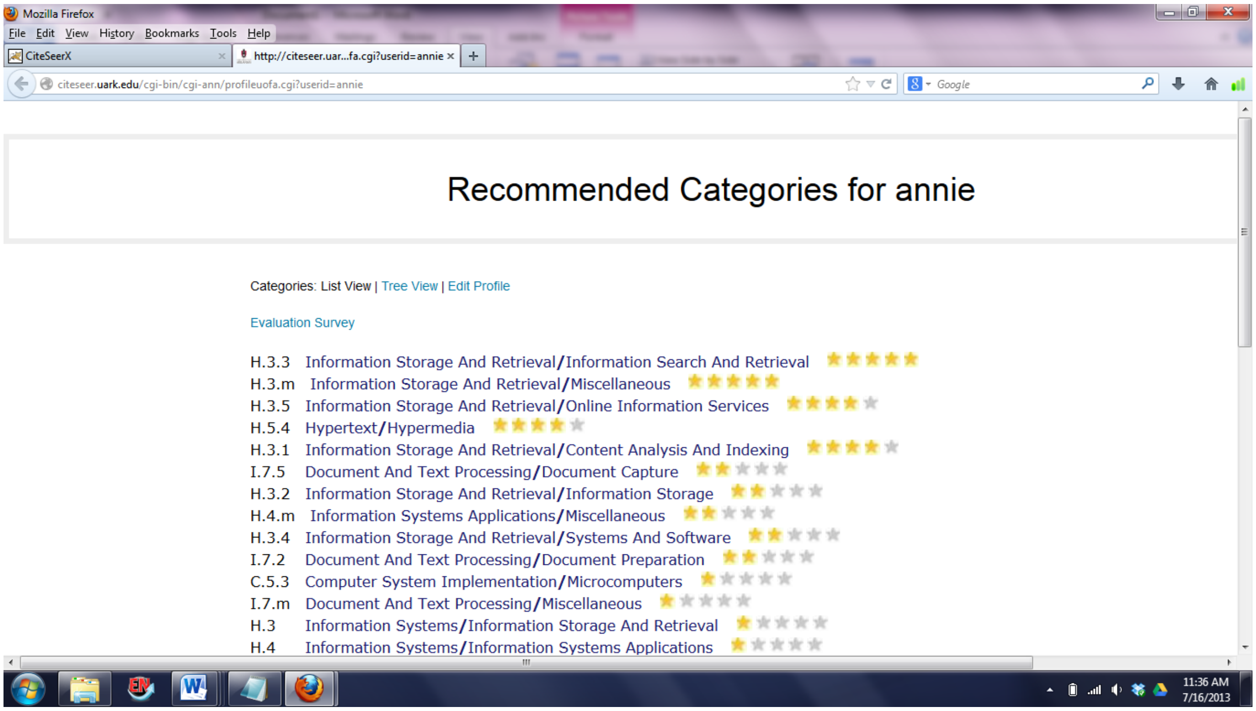
Developed at and hosted by The Department of Computer Science & Computer Engineering

In collaboration with IST at The Pennsylvania State University

© 2007-2013 University Of Arkansas

Figure 4.a: CiteSeer^x Registration and Login Webpage [45]

The survey participants were instructed to log in with their credentials and to search for topics that are of interest to them (single or multiple topics). The participants were requested to read 10 or more documents generated from the search. If the survey participant spends more than 10 sec on a specific document, then the recommender system assumes that the user is ‘reading’ the document. The time-limit is included for our recommender system to mark the concepts of these documents as relevant to the user and subsequently for the survey link to appear in the profile page. Fig 4.b shows the users top concepts and the link ‘Evaluation Survey’.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
CiteSeerX
http://citeseer.uar...fa.cgi?userid=annie
citeseer.uark.edu/cgi-bin/cgi-ann/profileuofa.cgi?userid=annie
Google
11:36 AM
7/16/2013

Recommended Categories for annie

Categories: List View | Tree View | Edit Profile

Evaluation Survey

H.3.3	Information Storage And Retrieval/Information Search And Retrieval	★★★★★
H.3.m	Information Storage And Retrieval/Miscellaneous	★★★★★
H.3.5	Information Storage And Retrieval/Online Information Services	★★★★☆
H.5.4	Hypertext/Hypermedia	★★★★☆
H.3.1	Information Storage And Retrieval/Content Analysis And Indexing	★★★★☆
I.7.5	Document And Text Processing/Document Capture	★★☆☆☆
H.3.2	Information Storage And Retrieval/Information Storage	★★☆☆☆
H.4.m	Information Systems Applications/Miscellaneous	★★☆☆☆
H.3.4	Information Storage And Retrieval/Systems And Software	★★☆☆☆
I.7.2	Document And Text Processing/Document Preparation	★★☆☆☆
C.5.3	Computer System Implementation/Microcomputers	★☆☆☆☆
I.7.m	Document And Text Processing/Miscellaneous	★☆☆☆☆
H.3	Information Systems/Information Storage And Retrieval	★☆☆☆☆
H.4	Information Systems/Information Systems Applications	★☆☆☆☆

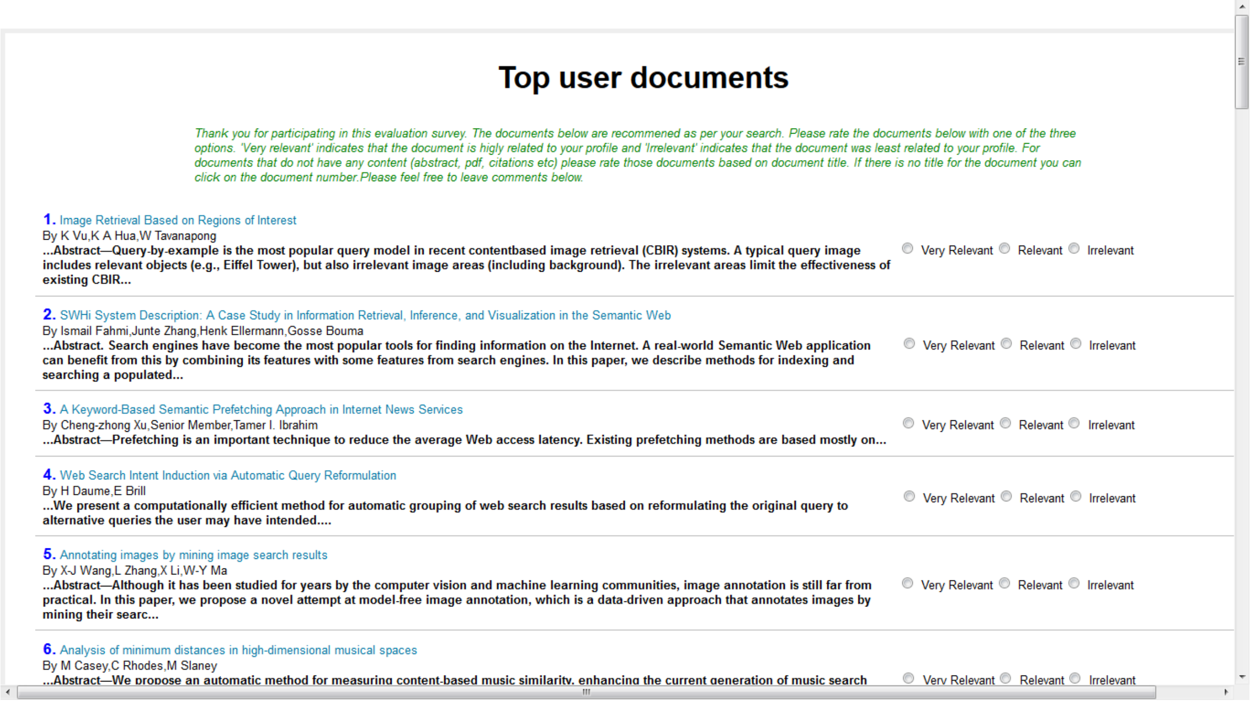
Figure 4.b: CiteSeer^x User Profile Webpage [45]

After the user clicks on the survey link, the participant is led to a webpage where all the recommended documents are displayed. The recommended documents is assumed to ‘portray’ the user’s research interests and is based on the recorded concepts from all of his/her previous search.

Using the CIBR formula described in section 3.2.3, each user evaluated 11 sets of documents generated by varying α from 0 to 1 in increments of 0.1. The results contained documents recommended based solely on the impact factor (CIBR calcualtions with α=0) as well as some recommended soley based on conceptual matches (CIBR calcualtions with α=1). The documents from all 11 results sets were merged and presented to the user in random order The maximum number of displayed documents possible for a user is 110 (11 sets * 10 documents per set). However, some documents in a set may overlap with other sets and therefore, in reality, each survey participant was asked to judge approximately 30 to 50 documents. The title, author and abstract were displayed for each of the recommended documents along with three rating options. The rating options were:

1. Very Relevant : The recommended document is very closely related to the user's search interests in CiteSeer^X.
2. Relevant : The recommended document is somewhat close to the user's search interest.
3. Irrelevant : The recommended document is not related to the user's search interest.

After reading the documents, the user is required to rate all the documents based on their relevance to user's research interest. The user selects one of three options for each document and concludes by submitting the survey. Figure 4.c shows the webpage with the list of recommended document list as displayed to the user.



Top user documents

Thank you for participating in this evaluation survey. The documents below are recommened as per your search. Please rate the documents below with one of the three options. 'Very relevant' indicates that the document is higly related to your profile and 'Irrelevant' indicates that the document was least related to your profile. For documents that do not have any content (abstract, pdf, citations etc) please rate those documents based on document title. If there is no title for the document you can click on the document number.Please feel free to leave comments below.

1. Image Retrieval Based on Regions of Interest
By K Vu,K A Hua,W Tavanapong
...Abstract—Query-by-example is the most popular query model in recent contentbased image retrieval (CBIR) systems. A typical query image includes relevant objects (e.g., Eiffel Tower), but also irrelevant image areas (including background). The irrelevant areas limit the effectiveness of existing CBIR...
○ Very Relevant ○ Relevant ○ Irrelevant

2. SWHi System Description: A Case Study in Information Retrieval, Inference, and Visualization in the Semantic Web
By Ismail Fahmi,Junte Zhang,Henk Ellermann,Gosse Bouma
...Abstract. Search engines have become the most popular tools for finding information on the Internet. A real-world Semantic Web application can benefit from this by combining its features with some features from search engines. In this paper, we describe methods for indexing and searching a populated...
○ Very Relevant ○ Relevant ○ Irrelevant

3. A Keyword-Based Semantic Prefetching Approach in Internet News Services
By Cheng-zhong Xu,Senior Member,Tamer I. Ibrahim
...Abstract—Prefetching is an important technique to reduce the average Web access latency. Existing prefetching methods are based mostly on...
○ Very Relevant ○ Relevant ○ Irrelevant

4. Web Search Intent Induction via Automatic Query Reformulation
By H Daume,E Brill
...We present a computationally efficient method for automatic grouping of web search results based on reformulating the original query to alternative queries the user may have intended....
○ Very Relevant ○ Relevant ○ Irrelevant

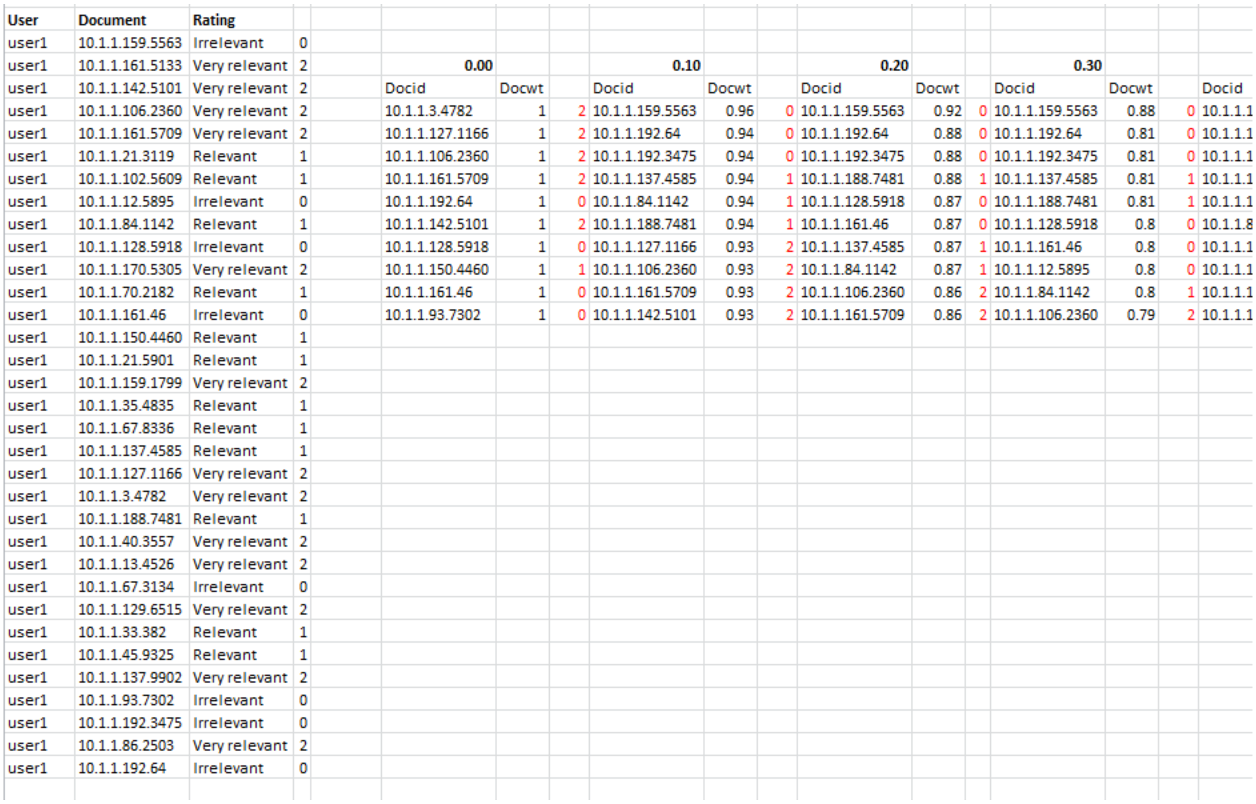
5. Annotating images by mining image search results
By X-J Wang,L Zhang,X Li,W-Y Ma
...Abstract—Although it has been studied for years by the computer vision and machine learning communities, image annotation is still far from practical. In this paper, we propose a novel attempt at model-free image annotation, which is a data-driven approach that annotates images by mining their searc...
○ Very Relevant ○ Relevant ○ Irrelevant

6. Analysis of minimum distances in high-dimensional musical spaces
By M Casey,C Rhodes,M Slaney
...Abstract—We propose an automatic method for measuring content-based music similarity, enhancing the current generation of music search
○ Very Relevant ○ Relevant ○ Irrelevant

Figure 4.c: CiteSeer^x Survey Webpage for Rating the Recommended List of Documents [45]

4.1 Data Analysis

Once the user ratings for all recommended documents was collected, we analyzed the data to determine which recommender system performed the best, i.e., impact-based, conceptual, or the hybrid recommender that combined inputs from the other two.



User	Document	Rating	
user1	10.1.1.159.5563	Irrelevant	0
user1	10.1.1.161.5133	Very relevant	2
user1	10.1.1.142.5101	Very relevant	2
user1	10.1.1.106.2360	Very relevant	2
user1	10.1.1.161.5709	Very relevant	2
user1	10.1.1.21.3119	Relevant	1
user1	10.1.1.102.5609	Relevant	1
user1	10.1.1.12.5895	Irrelevant	0
user1	10.1.1.84.1142	Relevant	1
user1	10.1.1.128.5918	Irrelevant	0
user1	10.1.1.170.5305	Very relevant	2
user1	10.1.1.70.2182	Relevant	1
user1	10.1.1.161.46	Irrelevant	0
user1	10.1.1.150.4460	Relevant	1
user1	10.1.1.21.5901	Relevant	1
user1	10.1.1.159.1799	Very relevant	2
user1	10.1.1.35.4835	Relevant	1
user1	10.1.1.67.8336	Relevant	1
user1	10.1.1.137.4585	Relevant	1
user1	10.1.1.127.1166	Very relevant	2
user1	10.1.1.3.4782	Very relevant	2
user1	10.1.1.188.7481	Relevant	1
user1	10.1.1.40.3557	Very relevant	2
user1	10.1.1.13.4526	Very relevant	2
user1	10.1.1.67.3134	Irrelevant	0
user1	10.1.1.129.6515	Very relevant	2
user1	10.1.1.33.382	Relevant	1
user1	10.1.1.45.9325	Relevant	1
user1	10.1.1.137.9902	Very relevant	2
user1	10.1.1.93.7302	Irrelevant	0
user1	10.1.1.192.3475	Irrelevant	0
user1	10.1.1.86.2503	Very relevant	2
user1	10.1.1.192.64	Irrelevant	0

0.00			0.10			0.20			0.30			
Docid	Docwt		Docid	Docwt		Docid	Docwt		Docid	Docwt		Docid
10.1.1.3.4782	1	2	10.1.1.159.5563	0.96	0	10.1.1.159.5563	0.92	0	10.1.1.159.5563	0.88	0	10.1.1.1
10.1.1.127.1166	1	2	10.1.1.192.64	0.94	0	10.1.1.192.64	0.88	0	10.1.1.192.64	0.81	0	10.1.1.1
10.1.1.106.2360	1	2	10.1.1.192.3475	0.94	0	10.1.1.192.3475	0.88	0	10.1.1.192.3475	0.81	0	10.1.1.1
10.1.1.161.5709	1	2	10.1.1.137.4585	0.94	1	10.1.1.188.7481	0.88	1	10.1.1.137.4585	0.81	1	10.1.1.1
10.1.1.192.64	1	0	10.1.1.84.1142	0.94	1	10.1.1.128.5918	0.87	0	10.1.1.188.7481	0.81	1	10.1.1.1
10.1.1.142.5101	1	2	10.1.1.188.7481	0.94	1	10.1.1.161.46	0.87	0	10.1.1.128.5918	0.8	0	10.1.1.8
10.1.1.128.5918	1	0	10.1.1.127.1166	0.93	2	10.1.1.137.4585	0.87	1	10.1.1.161.46	0.8	0	10.1.1.1
10.1.1.150.4460	1	1	10.1.1.106.2360	0.93	2	10.1.1.84.1142	0.87	1	10.1.1.12.5895	0.8	0	10.1.1.1
10.1.1.161.46	1	0	10.1.1.161.5709	0.93	2	10.1.1.106.2360	0.86	2	10.1.1.84.1142	0.8	1	10.1.1.1
10.1.1.93.7302	1	0	10.1.1.142.5101	0.93	2	10.1.1.161.5709	0.86	2	10.1.1.106.2360	0.79	2	10.1.1.1

Figure 4.1: Survey Feedback of User1

Figure 4.1 is a snapshot of a single user's feedback results for the sets of documents they rated and also that data broken out by various values of α. Values 0, 1, 2 are assigned to the options irrelevant, relevant and very relevant and subsequently, the recorded data are analyzed using the following four metrics:

4.1.1 Average Rating

We calculate *average ratings* of each α value for all the users. The *average ratings* is calculated first by accumulating the ratings for individual documents for each α value for each user and then by taking an average for all the documents in a set that belongs to each α value.

The ratings for individual documents are based on the user ratings of irrelevant, relevant and very relevant. Figure 4.1.1 shows the *average rating* of the documents for the various values of α. The graph shows that, averaged over all users, the highest average rating occurs with an α value of 0.1. This means that the users preferred recommended documents that were generated using a rating based 90% on the impact factor and 10% based on conceptual match with their profile. Since average rating does not take into consideration the rank order of the highly-rated documents within the set of 10 documents presented, we chose to explore better metrics for our analysis.

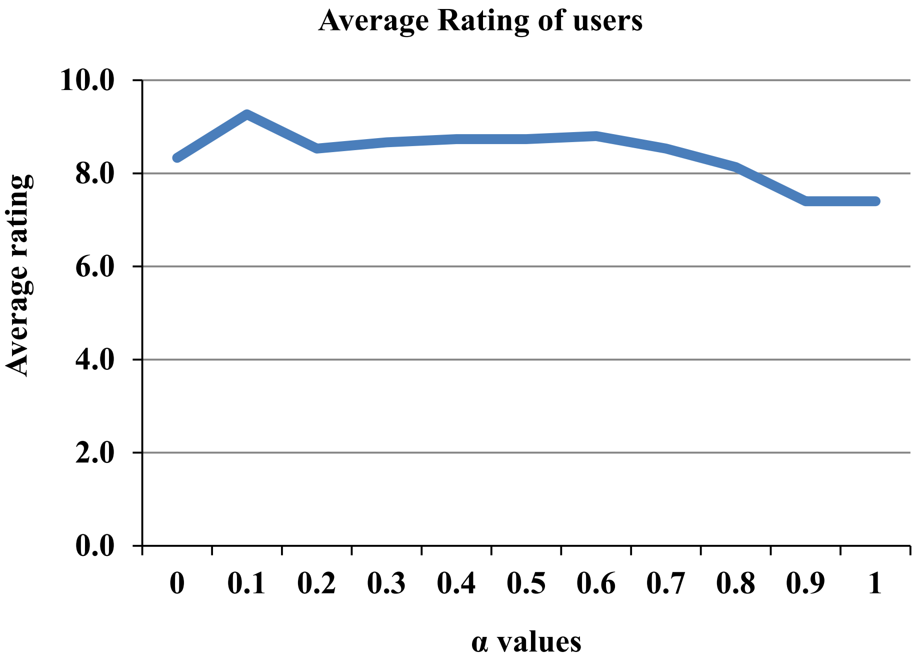


Figure 4.1.1: Average Rating of the Users for Different α Parameters.

4.1.2 Cumulative Rating

Cumulative rating is determined by the addition of previous ratings in a specific α value document set, where the documents are arranged in a particular rank order. The rank order of the documents are based on the weight of the documents, with documents arranged in the descending order of weights. The cumulative rating was calculated for each α value for individual users. For example, the cumultive sum for user 1 at α=0 is calculated as given in Table 4.1.2 a. The cumulative sum is determined for the different α values for all users and the average cumulative rating for each α value is estimated for all users as explained for α=0 in Table 4.1.2 b. In Table 4.1.2.a, we show an example of the cumulative calculation for α=0 for User1. The cumulative sum for User1 when α=0 is 81.

α=0 for User1			
Rank Order	*Documents*	*Ratings*	*Cumulative*
1	10.1.1.3.4782	2	2
2	10.1.1.127.1166	2	4
3	10.1.1.106.2360	2	6
4	10.1.1.161.5709	2	8
5	10.1.1.192.64	0	8
6	10.1.1.142.5101	2	10
7	10.1.1.128.5918	0	10
8	10.1.1.150.4460	1	11
9	10.1.1.161.46	0	11
10	10.1.1.93.7302	0	11
	Sum →	11	**81**

Table 4.1.2.a: Cumulative sum of α=0 for user1

Documents	*α = 0*
User1	**81**
User2	49
User3	34
User4	46
User5	51
User6	28
User7	52
User8	7
User9	6
User10	68
User11	92
User12	85
User13	23
User14	19

User15	74
Sum →	715
Cumulative Rating →	47.7

Table 4.1.2.b: Cumulative rating for α=0

In Table 4.1.2.b, we use the cumulative sum of different users at α=0 to find the cumulative rating. For the experiment, similar calculations were made for all α values. Figure 4.1.2 shows the cumulative rating of the documents for all the users. Based on this more sensitive metric, users perfomed best at α value= 0.6.

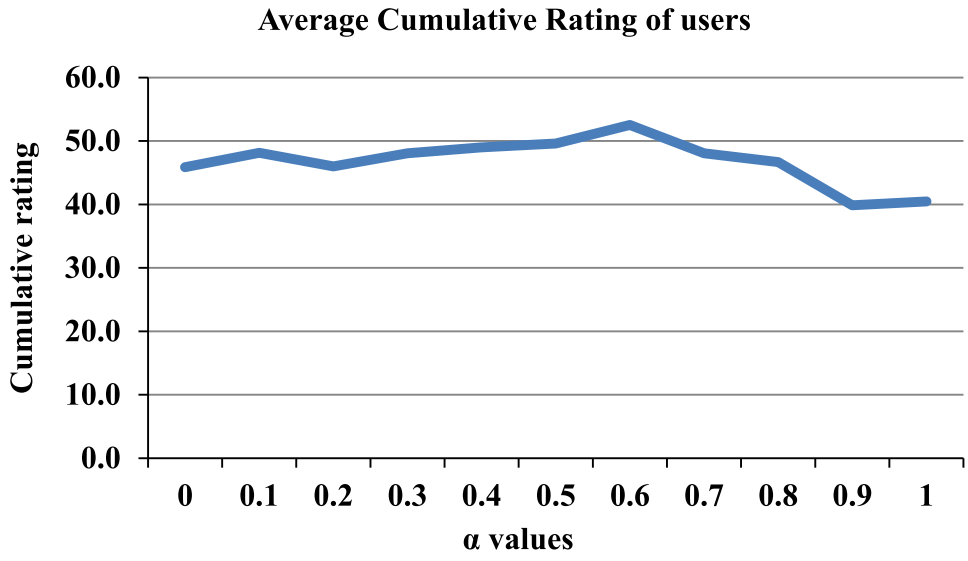


Figure 4.1.2: Cumulative Average Rating of the Users for Different α Parameters

4.1.3 Mean Average Precision

Mean Average precision (MAP) is the standard metric to evaluate search engines such as CiteSeer^x. Similar to Cumulative Rating, MAP takes the rank order of the recommended documents into account. MAP, however, only takes into account binary relevance judgments (relevant, non-relevant). To calculate it for our results, we treat both ‘relevant’ and ‘very relevant’ document ratings as relevant. As per [46], the MAP metric determines precision at each point when a new relevant document gets retrieved. After estimating the average for each query, the MAP then estimates average over queries as given in the equation,

$$MAP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{Q_j}\sum_{i=1}^{Q_j}P(doc_i)$$

where,

Q_j = number of relevant documents for query *j*

N = number of queries

$P(doc_i)$ = precision at *i*th relevant document.

Figure 4.1.3 shows the Mean Average Precision (MAP) of the recommender systems for all the users. Using this metric, we get very similar results to the Cumulative Rating and suggests that an α value of 0.6 provides the the best result.

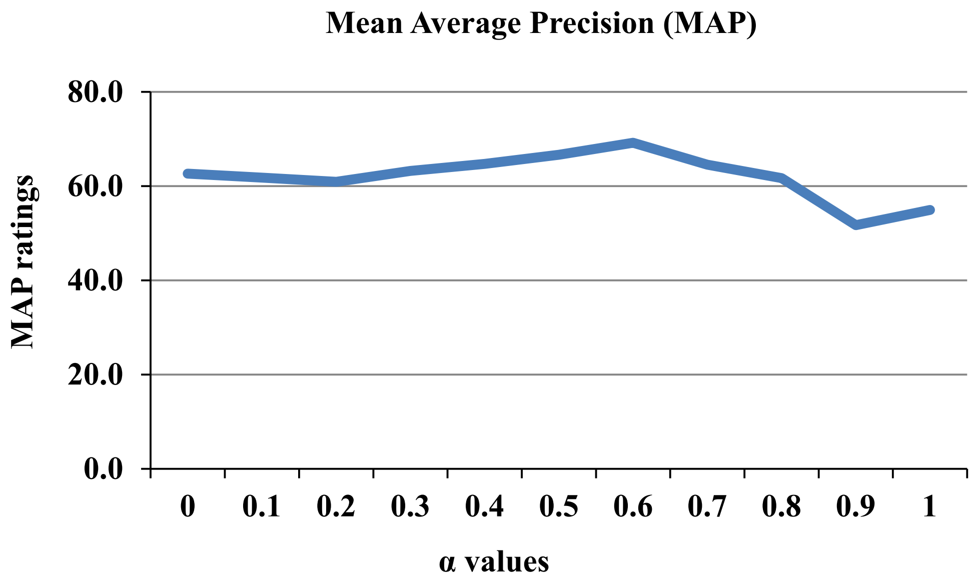


Figure 4.1.3: MAP for different α values for all users

4.1.4 Mean Average Weighted Precision

Mean Average Weighted Precision (MAWP) is essentially MAP modified to handle the distinction between 'relevant' and 'very relevant' documents. It takes relevance judgment weight into consideration rather than just counting the number and rankings of the relevant documents. Figure 4.1.4 shows the MAWP of the recommender systems for all the users. These results confirm those of the Cumulative Rating and MAP; the users best result is generated at an α value of 0.6. All three of the latter metrics reveal that a similar combined contribution of the impact and conceptual recommendations performs the best.

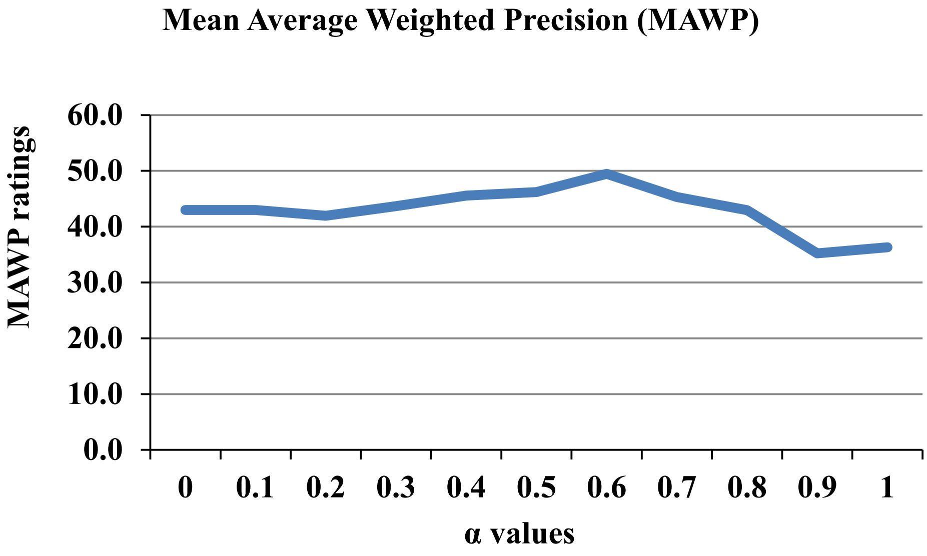


Figure 4.1.4: MAWP for Different α Parameters

4.2 Discussion

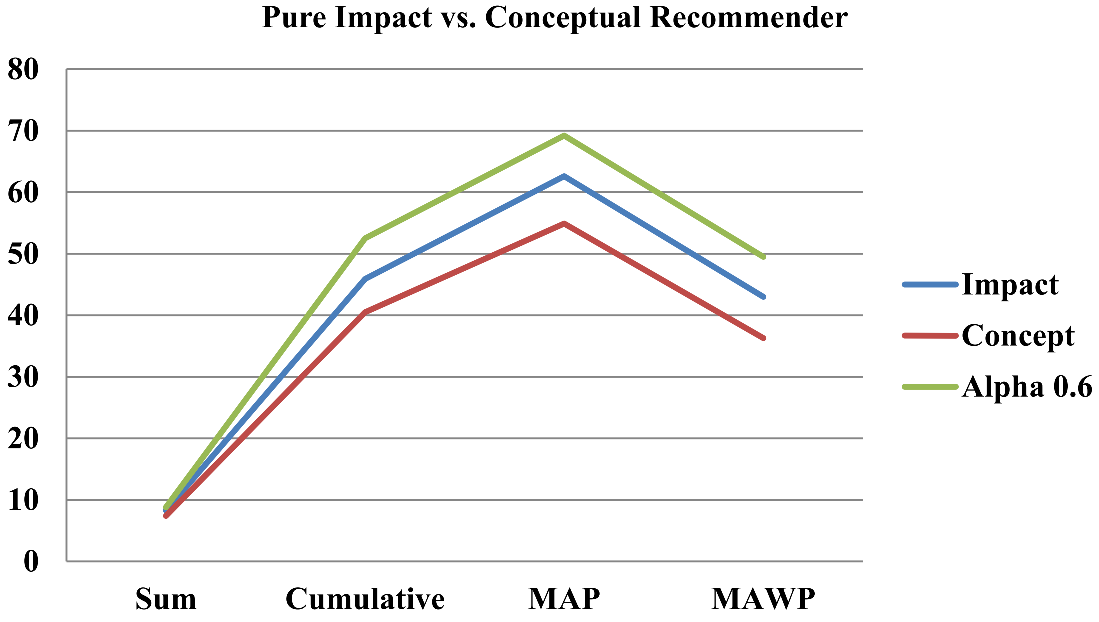


Figure 4.2.a: Comparison of the Pure Recommender Systems

Figure 4.2.a measures the users preference for a pure Impact-Based Recommender System (α=0) versus a pure Conceptual Recommender System (α=1). A comparison of pure Impact vs. Conceptual Recommender across all four metrics – Average, Cumulative Average, MAP and MAWP, suggest that the users preferred the documents returned by the pure Impact-Based Recommender System than the latter. The graph also suggests that the users preferred the hybrid recommender system of α = 0.6 than either of the pure recommender systems.

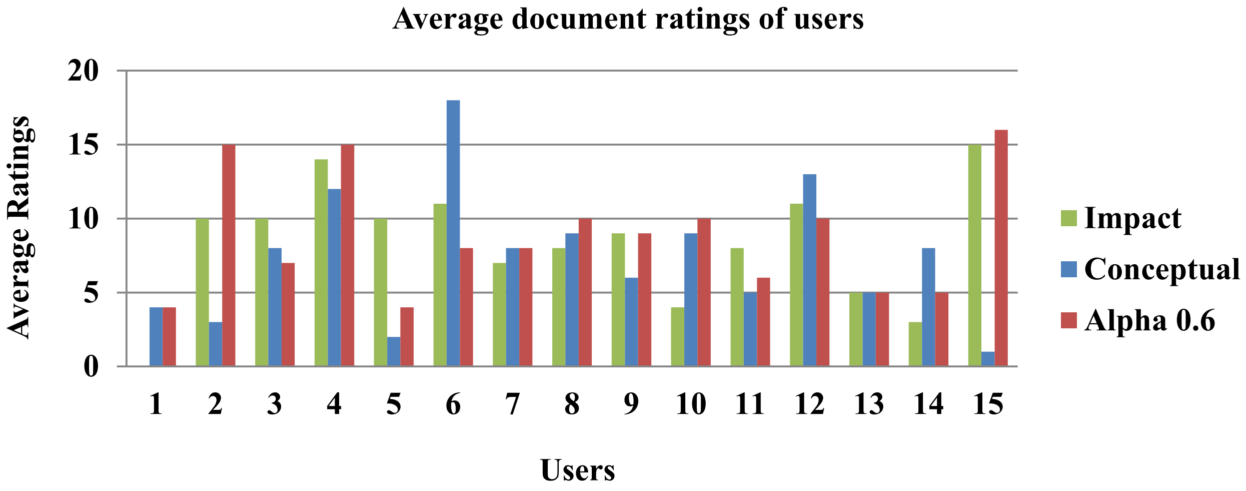


Figure 4.2.b: User Preference of the Pure Recommender Systems

Figure 4.2.b shows average ratings of both pure Impact vs. Conceptual Recommender Systems for individual users. A comparison of the individual user's average document rating suggest that an overall 50% of the users' preferred Impact-based recommended documents and the remaining 50% of the users preferred Conceptually recommended documents. The graph also shows that over 80% of the users preferred the hybrid recommender system when α is 0.6.

In conclusion, our results show that the Impact-Based Recommender outperformed the Conceptual Recommender, but that the hybrid system which combined the two approaches performed the best overall. In particular, the α value 0.6 which has a nearly equal combination of the two, produced the best overall performance.

5. CONCLUSIONS

5.1 Summary

The thesis discusses the development and evaluation of a new hybrid recommender system to recommend relevant documents to a CiteSeer^x user. The first part of the research included the development of an impact based recommender system that recommended papers to users based on the h-indexes of the authors of publications the users preferred. Subsequently, we developed a hybrid system that combined the impact-based recommender with an existing Conceptual Recommender System. The hybrid system was further evaluated using a survey experiment to determine the best ratio for generating the most relevant recommendations for the user. The final result is our Conceptual, Impact-Based Recommender System (CIBR).

Specifically, the CIBR system was implemented by combining the document weights produced by the Conceptual Recommender System with those produced by the Impact-Based Recommender System. Documents generated using the Conceptual Recommender System was represented by concept vectors containing non-zero weights for only the three highest-weighted concepts. The document-concept weights are calculated using a kNN classifier, trained on documents manually tagged with ACM CCS concepts by their authors. Authors are represented by user profiles automatically created as they examine search results and these profiles are also represented as weighted concept vectors and the top three concepts are selected to generate recommendations. The Conceptual Recommender System weights documents using the cosine similarity measure calculated on these abbreviated document and profile vectors.

In contrast, the Impact-Based Recommender System weights documents based on the impact factors of the document authors. The impact factor for a document was calculated using the highest h-index value of any author of the paper. The hybrid CBIR system normalizes these two scores so that each is in the range between 0 and 1 and further combines the weights in a relative contribution determined by a tunable parameter to produce a single weighting. After ranking the documents in decreasing order of their weight, the top 10 are presented to the user as recommendation for further reading.

The CIBR system for generating recommendations from the CiteSeer^x database was implemented and evaluated by a user study including 15 student and faculty participants from within the Computer Science and Computer Engineering department at the University of Arkansas. The survey participants each created a user profile and entered search queries to retrieve documents based on their research interest. The CIBR system generated recommendations for each user by providing a list of documents. Each document in the recommended list was retrieved either by using a pure conceptual based system, an impact based system or using different ratios of the two recommender systems. Results from the survey suggested that the users preferred documents returned by a combination of the Conceptual and Impact-Based Recommender Systems.

Specific contributions include:

1. The development of an Impact-Based Recommender System for CiteSeer^x based on the h-index values of authors.
2. The development of the CIBR hybrid system that uses a combination of impact and conceptual based systems to recommend documents.
3. Demonstration that Impact-Based Recommender System is effective than the previously implemented Conceptual Recommender System in CiteSeer^x.
4. Demonstration that our CIBR hybrid system is more effective than either, producing more accurate recommendations than either recommender system alone.

The documents generated from our study confirmed that our CIBR hybrid system were a more accurate match to the user profiles. However, future research should focus on refining the recommender system to improve reading suggestions for CiteSeer^x users.

5.2 Future work

The CIBR system is a definite improvement over the Conceptual Recommender System; however, further research should identify and resolve compromises in system components and provide better recommendations to users. In particular, the algorithm should be modified to reduce document ties in the CIBR system.

A user inputs multiple search strings and the CIBR system uses keywords from the user-read documents to retrieve documents from the CiteSeer^x database. The documents are retrieved based on concept weight, h-index values or a combination of both. However, if there is a list of documents that have similar weights or h-index values, the CIBR system randomly picks from this list to provide recommendation that are displayed to the users. If this list of 'tied' documents is long, that in turn increases the randomness of the selected document. This introduces more noise to the process and ultimately generates less accurate recommendations to the user. Therefore, research should explore strategies to mitigate erroneous recommendations arising from ties and subsequently implement the algorithm to test the different methods to understand the best strategy.

Documents that have the same value can be differentiated using multiple methods to give priority for those documents that are more relevant. For example, one method to rank the tied documents is to increase the number of concepts (currently, three) considered from the user profile and from the CiteSeer^x database. A preliminary study was conducted using a single user to observe if varying the number of concepts reduced document ties. Results suggested that if 15 concepts were used from the user profile and were matched to the top six concepts from the documents in the CiteSeer^x database, the document ties reduced by ca.10 fold, decreasing the number of tied documents from 316 to 29 for the impact-based recommender system. The conceptual recommender system originally had lesser document ties (3 documents) and using the modified number of concepts, it further reduced (2 documents) by 1.5 fold with the current system that uses 3 concepts only from both the document and user profile vectors. For this case, the pure impact–based recommender system actually outperformed even the CIBR system, producing 7 very relevant documents in the top 10 versus 3 with the CIBR system with the more complete vectors and 5 with the CIBR system using only 3 concepts from each vector. With only one user being studied, these results are preliminary and merely illustrative. However, they indicate that we need further study on how to make the best use of the document and user profile vectors.

There is also information from previous literature that lists several other methods to overcome the document tie limitation. Rousseau (2008a) suggested that tied documents can further be ranked based on impact factor of the journals or by considering the year of publication, with the most recent publication having the highest rank among the tied documents [47]. Another proposal by Rousseau (2008b) suggests that if multiple researchers had the same h-index at a time duration T and the same number of citations, then the similar h-indices are re-ranked based on a measure of increase in productivity of the researchers [48]. Specifically, a ‘convex’ productivity increase is preferred over a ‘linear’ increase, which in turn is superior to a ‘concave’ h-function change in research productivity over time. This is based on the contents and size of the all publications in the h-index list, number of citations for each publication in the list, and finally the recent variations in h-index of the authors.

Another approach is to consider different variants of the h-index to address authors with the same h-index. For example, Zhang (2009) introduced the e-index and this variation complements h-index and handles noisy citation information and the low resolution of the h-index. This is specifically useful to increase ranking for highly cited scientists [49]. Garcia-Perez (2009) suggested the use of a multidimensional extension to h-index that handles authors with the same h-index, particularly, when the tied documents have low h-indices. Here, multiple components are used to calculate the h-index. When the h-indices calculated using one component is tied, other components help to differentiate documents with the same h-index [50]. Future research should focus to evaluate the above-mentioned strategies and determine an optimum method for generating more relevant recommendations.

REFERENCES

[1] Ricci F. and Shapira B. *"Recommender systems handbook"* Springer, 2011.

[2] Jannach D., Zanker M., Felfernig A., and Friedrich G. *"Recommender systems: an introduction"* Cambridge University Press, 2010.

[3] Jannach D. and Friedrich G. "Tutorial: Recommender Systems". in *Proceedings of the International Joint Conference on Artificial Intelligence, Barcelona*, 2011.

[4] Hirsch J. E. "An index to quantify an individual's scientific research output". in *Proceedings of the National Academy of Sciences of the United States of America* 102(46):16569-16572, 2005.

[5] Bras-Amorós M., Domingo-Ferrer J., and Torra V. "A bibliometric index based on the collaboration distance between cited and citing authors". in *Journal of Informetrics* 5(2):248-264, 2011.

[6] Zhang C.-T. "The e-index, complementing the h-index for excess citations". in *PLoS One* 4(5):e5429, 2009.

[7] Egghe L. "Theory and practise of the g-index". in *Scientometrics* 69(1):131-152, 2006.

[8] Bergman M. K. "White paper: the deep web: surfacing hidden value". in *journal of electronic publishing* 7(1), 2001.

[9] Lincoln A. "FYI: TMI: Toward a holistic social theory of information overload". in *First Monday* 16(3-7), 2011.

[10] Yang J. and Filo D. (1995) Yahoo. (http://search.yahoo.com/).

[11] Page L. and Brin S. (1998) Google. (https://http://www.google.com/).

[12] Ballmer S. (2009) Bing. (http://www.bing.com/).

[13] Pinkerton B. "Finding what people want: Experiences with the WebCrawler". in *Proceedings of the Second International World Wide Web Conference* 94:17-20, 1994.

[14] Ricci F., Rokach L., and Shapira B. Introduction to recommender systems handbook. *Book "Introduction to recommender systems handbook"*, Springer, pp 1-35, 2011.

[15] Burke R. Hybrid web recommender systems. *Book "Hybrid web recommender systems"*, Springer, pp 377-408, 2007.

[16] Ricci F. "Mobile recommender systems". in *Information Technology & Tourism* 12(3):205-231, 2010.

[17] Adomavicius G. and Tuzhilin A. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". in *Knowledge and Data Engineering, IEEE Transactions on* 17(6):734-749, 2005.

[18] Koren Y. and Bell R. Advances in collaborative filtering. *Book "Advances in collaborative filtering"*, Springer, pp 145-186, 2011.

[19] Herlocker J. L., Konstan J. A., Borchers A., and Riedl J. "An algorithmic framework for performing collaborative filtering". in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp 230-237, 1999.

[20] Anand S. S. and Mobasher B. "Intelligent techniques for web personalization". in *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*, Springer-Verlag, pp 1-36, 2003.

[21] Sarwar B., Karypis G., Konstan J., and Riedl J. "Analysis of recommendation algorithms for e-commerce". in *Proceedings of the 2nd ACM conference on Electronic commerce*, ACM, pp 158-167, 2000.

[22] McNee S. M.*, et al.* "On the recommending of citations for research papers". in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, ACM, pp 116-125, 2002.

[23] Lemire D. and Maclachlan A. "Slope one predictors for online rating-based collaborative filtering". in *Society for Industrial Mathematics* 5:471-480, 2005.

[24] Su X. and Khoshgoftaar T. M. "A survey of collaborative filtering techniques". in *Advances in artificial intelligence* 2009:4, 2009.

[25] Lops P., de Gemmis M., and Semeraro G. Content-based recommender systems: State of the art and trends. *Book "Content-based recommender systems: State of the art and trends"*, Springer, pp 73-105, 2011.

[26] Chakrabarti S. *"Mining the Web: Discovering knowledge from hypertext data"* Morgan Kaufmann, 2003.

[27] Allan J., Carbonell J. G., Doddington G., Yamron J., and Yang Y. "Topic detection and tracking pilot study final report". in, 1998.

[28] Salton G. "Automatic text processing". in *Science* 168(3929):335-343, 1970.

[29] Pazzani M. J. and Billsus D. Content-based recommendation systems. *Book "Content-based recommendation systems"*, Springer, pp 325-341, 2007.

[30] Billsus D., Pazzani M. J., and Chen J. "A learning agent for wireless news access". in *Proceedings of the 5th international conference on Intelligent user interfaces*, ACM, pp 33-36, 2000.

[31] Middleton S. E., Shadbolt N. R., and De Roure D. C. "Ontological user profiling in recommender systems". in *ACM Transactions on Information Systems (TOIS)* 22(1):54-88, 2004.

[32] Rocchio J. J. (Relevance feedback in information retrieval, in the SMART Retrieval System--Experiments in Automatic Document Processing, Englewood Cliffs, NJ, 1971. (Prentice Hall, Inc).

[33] Balabanović M. and Shoham Y. "Fab: content-based, collaborative recommendation". in *Communications of the ACM* 40(3):66-72, 1997.

[34] Ahn J., Brusilovsky P., Grady J., He D., and Syn S. Y. "Open user profiles for adaptive news systems: help or harm?". in *Proceedings of the 16th international conference on World Wide Web*, ACM, pp 11-20, 2007.

[35] Pazzani M. J., Muramatsu J., and Billsus D. "Syskill & Webert: Identifying interesting web sites". in *Proceedings of the Thirteen National Conference on Artificial Intelligence*, pp 54-61, 1996.

[36] Billsus D. and Pazzani M. J. "A hybrid user model for news story classification". In *Courses And Lectures-International Centre For Mechanical Sciences*:99-108, 1999.

[37] Pudhiyaveetil A. K., Gauch S., Luong H., and Eno J. "Conceptual Recommender System for CiteSeerx". in *Proceedings of the third ACM conference on Recommender systems*, ACM, pp 241-244, 2009.

[38] Councill I. G*., et al.* Towards next generation citeseer: A flexible architecture for digital library deployment. *Book "Towards next generation citeseer: A flexible architecture for digital library deployment"*, Springer, pp 111-122, 2006.

[39] Pudhiyaveetil A. K. "Conceptual Recommender system". Masters University of Arkansas, 2010.

[40] Chandrasekaran K., Gauch S., Lakkaraju P., and Luong H. P. "Concept-based document recommendations for citeseer authors". in *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, pp 83-92, 2008.

[41] Melville P., Mooney R. J., and Nagarajan R. "Content-boosted collaborative filtering for improved recommendations". in *Proceedings of the Eighteenth National Conference on Artificial Intelligence(AAAI)*, pp 187-192, 2002.

[42] Cosley D., Lawrence S., and Pennock D. M. "REFEREE: An open framework for practical testing of recommender systems using ResearchIndex". in *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB Endowment, pp 35-46, 2002.

[43] Gunawardana A. and Meek C. "A unified approach to building hybrid recommender systems". in *Proceedings of the third ACM conference on Recommender systems*, ACM, pp 117-124, 2009.

[44] Speretta M. and Gauch S. "Personalized search based on user search histories". in *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, IEEE, pp 622-628, 2005.

[45] Lawrence S., Giles C. L., and Bollacker K. (1997) CiteSeerX. http://citeseer.uark.edu/.

[46] Teufel S. An overview of evaluation methods in TREC ad hoc information retrieval and TREC question answering. *Book "An overview of evaluation methods in TREC ad hoc information retrieval and TREC question answering"*, Springer, pp 163-186, 2007.

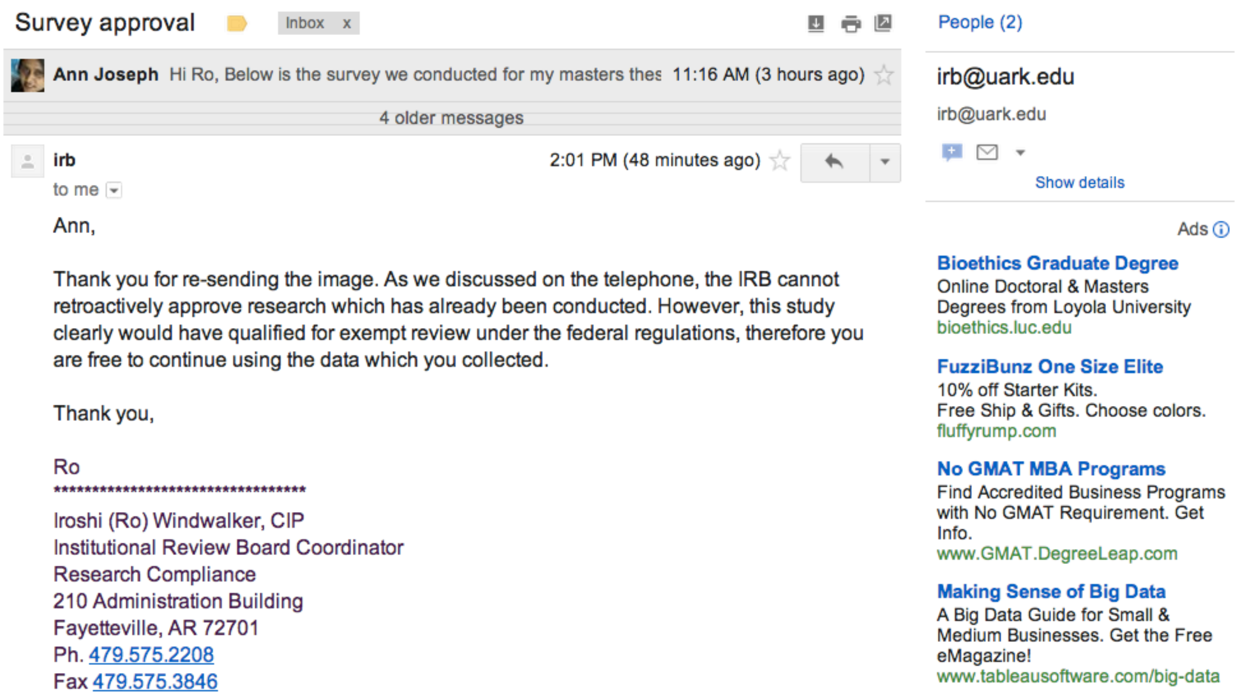
[47] Rousseau R. and Leuven K. "Reflections on recent developments of the h-index and h-type indices". in *COLLNET Journal of Scientometrics and Information Management* 2(1):1-8, 2008.

[48] Rousseau R. and Ye F. Y. "A proposal for a dynamic h-type index". in *Journal of the American Society for Information Science and Technology* 59(11):1853-1855, 2008.

[49] Zhang C.T. "The e-index, complementing the h-index for excess citations". in *PLoS One* 4(5), 2009.

[50] García-Pérez M. A. "A multidimensional extension to Hirsch's h-index". in *Scientometrics* 81(3):779-785, 2009.

6. APPENDIX



Survey approval

Ann Joseph Hi Ro, Below is the survey we conducted for my masters thes 11:16 AM (3 hours ago)

4 older messages

irb 2:01 PM (48 minutes ago)

to me

irb@uark.edu

irb@uark.edu

Ann,

Thank you for re-sending the image. As we discussed on the telephone, the IRB cannot retroactively approve research which has already been conducted. However, this study clearly would have qualified for exempt review under the federal regulations, therefore you are free to continue using the data which you collected.

Thank you,

Ro
****
Iroshi (Ro) Windwalker, CIP
Institutional Review Board Coordinator
Research Compliance
210 Administration Building
Fayetteville, AR 72701
Ph. 479.575.2208
Fax 479.575.3846