7-2015

Automatic User Profile Construction for a Personalized News Recommender System Using Twitter

Shiva Theja Reddy Gopidi
University of Arkansas, Fayetteville

Automatic User Profile Construction for a Personalized News Recommender System Using Twitter

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

By

Shiva Theja Reddy Gopidi
Jawaharlal Nehru Technological University
Bachelor of Technology in Computer Science, 2011

July 2015
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

ABSTRACT

Modern society has now grown accustomed to reading online or digital news. However, the huge corpus of information available online poses a challenge to users when trying to find relevant articles. A hybrid system “Personalized News Recommender Using Twitter’ has been developed to recommend articles to a user based on the popularity of the articles and also the profile of the user. The hybrid system is a fusion of a collaborative recommender system developed using tweets from the “Twitter” public timeline and a content recommender system based the user’s past interests summarized in their conceptual user profile. In previous work [1], a user’s profile was built manually by asking the user to explicitly rate his/her interest in a category by entering a score for the corresponding category. This is not a reliable approach as the user may not be able to accurately specify their interest for a category with a number. In this work, an automatic profile builder was developed that uses an implicit approach to build the user’s profile. The specificity of the user profile was also increased to incorporate fifteen categories versus seven in the previous system. We concluded with an experiment to study the impact of automatic profile builder and the increased set of categories on the accuracy of the hybrid news recommender system.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my Advisor Dr.Susan Gauch for her invaluable guidance and encouragement throughout the pursuit of my Master's degree. I would also like to thank my thesis committee members, Dr. Brajendra Panda and Dr. David Andrews for their valuable participation. I would like to thank the faculty and staff of Computer Science department for their cooperation.

I thank my Mother for bringing me up and helping me get a good education. I would also like to thank my cousin Gautham Krishna Reddy and my brother Ravi for their encouragement in pursuit of my Master's degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation:

Due to the increase in pervasive use of technology, people have become increasingly accustomed to reading digital or online news. Users can access the news stories across different sources such as Google [25], Yahoo [26], and CNN [28]. However, the huge corpus of information available poses a challenge to users who may have a difficulty finding the most popular and interesting stories that appeal to him. A user can easily waste time wading through information that is of no use or interest to him. To solve this problem, many news recommender systems have been implemented.

A recommender system usually presents the items to a user in a specific order so that the user comes across the most popular or interesting items first. The popular movie streaming service Netflix [27] recommends content to the user based on the history of the user as well as the overall popularity of the corresponding content across its service. Popular music streaming service Pandora [29] recommends songs using the history or profile of the user. Amazon [24] uses the correlation concept to recommend the items to a user. It relies on the relationship between the primary driver items and the identification of affinity items. The recommender systems discussed above usually implement either one of the two basic approaches to recommendations, *content-based filtering* or *collaborative filtering*.

A news recommender system based on *content-based filtering* looks for similarity between the contents of the news articles to make its recommendations. An article 'A' whose content is similar to that of article ‘B’ will be rated more highly than article ‘C’ whose content is less similar. Content-based filtering is effective only when there are semantic features, in this case, words, that can be extracted from the items to be recommended. In contrast, collaborative filtering systems are employed where such features are unavailable and they recommend items to users based on the patterns of use of items across large collections of users. This approach typically requires very large collection of data sets. It makes use of the information or preferences across a large number of users to recommend news to a particular user. Both content-based recommender systems and collaborative recommender systems have drawbacks associated with them. A content-based recommender system may fail to assign a good score for a very popular article if the content of the article is different to that of the articles that the user had rated highly in the past. However, if there is a major new event unrelated to a user’s usual interests, e.g., a train crash in Philadelphia, the user would likely want to know about it. On the other hand, collaborative recommenders need to have data about a large number of users in order to recommend items accurately. If the data pool is too small, then the accuracy may suffer. In particular, they suffer from the *cold start* problem wherein a new collaborative recommender system has no data and therefore can make no recommendations until users have used the system enough to create data. To make use of the strengths of both approaches, many sites employ hybrid recommender systems that are fusions of collaborative and content-based recommender systems.

We are building on previous work that employs a hybrid recommender system in order to recommend news articles from CNN to users. “Personalized News Recommender Using Twitter” [1] is a fusion of a collaborative recommender system implemented using user data from the popular social media Twitter and a conceptual, content-based recommender system that recommends news articles based on the similarity of their categories to categories stored in the personal profile of the user. Experimental results showed that the hybrid system above outperformed both the collaborative and content-based recommender systems alone, as evaluated using the accuracy of the recommendations produced.

In the previous work, users had to create their own profiles by explicitly rating the categories in the profile. We extended that work in two ways: we allowed users to explicitly rate documents and, based on those document ratings, automatically built the user profile. We also extended the system to use a more detailed set of 15 categories to better identify the users' interests. We then evaluated the quality of the recommendations produced using this more detailed, automatically constructed user profile.

1.2 Objective

The objective of this thesis is to:

1. Design an automatic profile building system based on user feedback.
2. Implement a content-based recommender system that exploits the user profile.
3. Implement a collaborative recommender system based on the existing Hybrid system "Personalized News Recommender using Twitter".
4. Implement the new hybrid recommender system that fuses information from the content-based and collaborative recommender systems.

1.3 Organization of this Thesis

In Chapter 2 we discuss the literature related to content-based recommender systems and collaborative recommender system. We examine the different aspects related to building a Hybrid recommender system as well. In Chapter 3 we discuss about our system in detail. Chapter 4 covers the experimental set up for the system and also the evaluation of the results. In Chapter 5 we summarize our thesis and the future work that can be done to make further improvements.

2. BACKGROUND

2.1 Recommender Systems

Recommender systems are designed to predict the preferences of a user for a previously unseen item. There are various recommender systems that are currently in place that recommend various types of items such as music, books, movies, restaurants and news articles to a user. A recommender system needs to possess enough data on which to formulate its predictions. There are different ways in which a recommender system collects data from the user. Some of the recommender systems collect data implicitly based on a user's actions whereas the other recommender systems explicitly ask the user to input their feedback or rating about an item. Online giants such as Google and social media like Facebook [30] collect data implicitly from the user and use it to increase the accuracy of the advertisements that are recommended to the user. Popular music streaming service 'Pandora' collects the data explicitly from the user. It provides the user with two options 'like' and 'dislike' for each song and based on the user input it determines the songs to be recommended to the user.

A recommender system usually employs either a content-based approach or a collaborative approach. Content-based approaches depend on the similarity in the content between the items that users have liked or purchased in the past and other items in the database. In contrast, collaborative approaches use the opinions or preferences of a large number of people to predict the preferences of a user for a specific item. Hybrid systems combine these two approaches to overcome the drawbacks associated with the individual approaches.

As discussed previously, the recommender systems can be classified into three types:

1. Content-based recommender system
2. Collaborative recommender system
3. Hybrid recommender system

2.1.1 Content-based Recommender Systems

Content-based systems depend on the content of the item and the user's profile to recommend an item. This approach is most commonly employed when the items to be recommended have substantial textual description. In these cases, keywords are typically extracted from the descriptions and they are used to identify the content. For example in 'Pandora' the genre and the name of the artist for a song represent its contents. Each and every item is measured by the keywords present in it. The weight or the importance of the keywords in a document are measured in many ways. The most commonly employed approach is term frequency –inverse document frequency represented by tf*idf. The term frequency (tf) is measured by the number of times a word appears in a document divided by total number of words in the document. The inverse document frequency (idf) is measured by the log of the number of documents in which the word occurs divided by the number of documents in the collection. The rarer the word, the higher the idf. In a nutshell, tf is a measure of the word's importance in the document and idf is a measure of the word's importance in the collection.

A content-based recommender system is typically implemented in three phases. In Content Analyzer phase, the items are analyzed, words (features) are extracted, and the weight or importance of the features are calculated. In the Profile Creation phase, features are given weights according to their importance to specific user.   This mapping of features to users is typically stored in each user’s profile. It is built either implicitly from existing information about a user’s activity or explicitly using feedback from the user. In the third phase, the Recommendation phase, the similarity between the feature weights of all items in the database and the feature weights in the user’s profile is calculated to obtain a score for each item in the database. Items are then recommended to the user in decreasing order of similarity.

Initial work in content-based recommender systems extracted features that were simply keywords weighted with tf-idf. However, these approaches suffer from the problem of ambiguity, specifically, synonymy (where one idea may be expressed with several different words) and hononymy (where one word has many meanings). Thus more recent approaches extract more abstract features, i.e., they represent items and users with concept or category weights rather than keyword weights. Semerao and Lops [4] take this approach one step further. They propose a knowledge-based content recommender system that incorporates linguistic knowledge, domain dependent language, and also social knowledge as part of the background knowledge gathering process. They argue that a content-based recommender system which that culturally and linguistically expanded knowledge resources would perform better than a normal system which relies strictly on words.

Di Noia and Mirizzi [3] developed a content-based recommender system for movies. They used information extracted from the Linked Open Data (LOD) [7] cloud. They extracted information from DBPedia, Freebase, and LinkedMDB. The LOD contains items, in this case movies, semantically tagged with property-value pairs, for example, directors, actors, genres, etc. This approach enabled them to overcome the normal disadvantages associated with extracting the information from text since the data is already represented in a semantic way. They used the traditional vector-space model, representing each movie as a vector of features where each property was one the element of the feature vector.   They adapted the cosine similarity metric to calculate the semantic similarity between two movies. They evaluated their recommender system with large test collections and found that the semantic recommender outperformed structural and keyword based recommender systems.

Loh and Lorenzi [5] worked on using keywords with and without conceptual classes in a content-based recommender system for advertisements on a website. They evaluated the effect of using weighted features comprised of classes (categories) versus keywords versus combinations of the two as the basis of the recommendations. They used a taxonomy of classes representing properties, products, services, and tourism. They build the user profile using implicit feedback while the users performed everyday tasks. The factors used to build the profile were the length of the time spent by the user viewing the ads and the content of the ads seen by the user. .

The user profile was built four different ways. The first approach involved only the classes of the ads that were visited by the user. Ads belonging to the classes that the user had previously visited were recommended with this approach. The second approach used only the top keywords to build the profile where keywords were ranked based on the frequency of their occurrence within the ads visited by the user. Only the ads in which top keywords were present were recommended to the user. The third approach was a combination of the first two approaches. In this approach, a user profile is represented by the classes augmented with word frequency vectors. For an ad to be recommended to a user it should have at least one of the top five words in the class vector. The fourth approach is similar to the third one but it only uses one top word in each class vector.

Evaluation was carried out in order to determine the efficiency of the four approaches. The user was asked to rate either one of the 'like' which meant the recommendation was useful or 'dislike' which meant the recommendation was not useful. The results proved that the third and fourth approaches performed better. They found that the approach in which only the classes were used to build the profile performed the worst. The results proved that combining classes and keywords achieved better results rather than when they were used alone.
Gemmis and Lops [6] built a semantic content-based recommender system that applies machine learning techniques on both the items as well as the taxonomy attributed to the content by the user. Semantic indexing of the documents was implemented in this work to find the relevancy of the items. Every document is represented as a list of words [w1, w2, w3...wn] in the order of occurrence of the words in the documents. WordNet was then used to find the synsets [s1, s2, s3…sn] with each synset populated by the semantically similar words. The number of synsets could be less than the number of listed words in the document as some of the words may not be found in the WordNet. They applied the Leacock-Chodorow measure [2] to compute the similarity between the items.

The user profile was constructed using feedback from the users. The users were presented with the items and were asked to rate them. The documents with the rating in between a certain range were considered to have the probability of the user liking the items as positive and the rest of the documents with the probability of the user liking the documents as negative. The work also focused on utilizing the user generated content such as tags to build the content-based recommender system. Hence, along with the score the users were also asked to associate tags with the content. Art work is used as the item to be recommended in this work. The content of the item is represented by the author name, title and description. Tags were considered as the fourth part along with these three. Bayesian learning algorithm was then used to develop the classification. Experimental evaluation over 26 users proved that the instances where user generated taxonomy was used as part of the user profile yielded better results.

Lu and Yu [8] worked on developing a framework that uses a content-based approach to recommend tags to annotate webpages using the similarity between the content of the webpages as the basis. They based their work on the premise that similar articles tend to have similar tags associated with them. They represented each document with two vectors, the tag vector and the term vector. The tags that are present only in a fewer documents were attached with much more importance in a similar way as to the tf-idf approach where words that are present in fewer documents are given more weight. Each document or web page is represented by its URL. A URL is said to be well represented if there are many of tags associated with it. For the process of generating content they considered two URLs. One is the provider that already has the tags associated with it and the other one is the receiver whose tags are to be generated. The provider generates tags for the receiver if the similarity of their contents is high. The similarity is measured by the cosine similarity between the corresponding tag vectors and term vector pairs. They evaluated their system using over 1 million URLs from the Delicious homepage and preprocessed them. At this stage, they removed about one fourth of the tags and used the remaining tags to train the system. Experimental results demonstrated that about 32% of the tags given by the users rank in the top five generated by the system and about 69.45% of the tags recommended by their system were deemed to be relevant by the participants.

2.1.2 Collaborative Recommender Systems

Collaborative recommender systems use the collective opinion or preferences from a large set people or items to predict the response of a user for a given scenario. Collection of the training preference data can be either implicit or explicit. In the explicit case, the user will be asked to provide his opinion about a product in order to determine his interest. Popular online retail service Amazon occasionally use this approach [16]. In the implicit approach, a given user's behavior or actions are collected as they do their everyday tasks (browse, search, bookmark) and these actions are analyzed to generate likely preferences. Social media such as Facebook and Twitter follow this approach to recommend friends [16]. Regardless of how the information is collected, recommendation can also be dependent on either similarities between users or similarity between items.

In user-based recommendation, if user 'A' likes an item and the history of user 'B' is similar to that of the user 'A', then it is assumed that there is a greater probability that user 'B' will like the item too,. So, items are recommended to users who have similar tastes. For this to work, each user's history of past purchases or likes must be collected and then the user-user similarity is calculated. In contrast, item-based recommendations are generated using the correlation between different items. For this to work, each item's history of the users whom have purchased or liked it must be collected and then the item-item similarity is calculated. For example, the movie 'Avengers' may be recommended to a user who bought the 'Ironman' movie if many other users who purchased 'Avengers' also purchased 'Ironman'.

Collaborative recommender systems can also be classified into either memory-based or model-based systems. In memory-based systems, recommendation systems use the ratings given by the user as the metric to build the recommendation system. The ratings are obtained using either one of the user based or item based recommendation approaches.

There are three shortcomings associated with the collaborative recommender systems: 1) the cold start problem, which reflects the inability of the recommender system to make predictions for a new user due to the lack of enough data about the user; 2) the scalability problem, which represents the computational complexity of the algorithms that need to be computed to find out the most similar user-item pairs for each user; and 3) the sparsity problem which reflects the lack of user feedback on many or most items in a large collection of items.

All of the problems above are related to the fact that traditional collaborative recommender systems need a huge amount of a users' rating data. This problem is particularly acute when the system needs to be trained prior to making any recommendations. This can lead to problems when new training data arrives and the system has to be trained all over again. Retraining consumes both resources and time and can prevent the system from being able to make instantaneous recommendations. Wang and Hoi [9] addressed this problem by proposing a set of online multitask collaborative filtering algorithms based on online multitask learning. Traditional online collaborative filtering-based approaches employ the simple online gradient descent algorithms to solve the matrix factorization tasks and thus avoid the re-training cost associated with the traditional batch matrix factorization algorithms. Thus, they are able to make on-the-fly recommendations for a user without retraining. The algorithms proposed by Wang and Hoi further improved upon the traditional OCF by adaptively updating the vector weights for the user in question and other nearby users based on the user interaction matrix.

User-based collaborative recommender systems make use of the observed similarity between different users to predict the preferences of a user. Two users are considered to be similar and treated as neighbors if they had provided similar ratings for similar items in the past. This approach is used by Resnick and lacovou to implement GroupLens [17] that recommends news articles to a user. Because there are typically many items in the site's collection, e.g., consider the many millions of items that Amazon sells, these vectors are sparse, so it is difficult to accurately identify neighbors for a given user. Alejandro and Pablo [10] developed and discussed several different neighborhood weighting functions and the effect these functions have on recommender system performance. They reason that better metrics need to be developed to measure the neighborhood similarity. Their work showed that the correct neighborhood functions are an important component to providing accurate recommendations.

Popular items are bought by many people and thus they have a lot of usage data associated with them. This can bias a collaborative recommender systems toward recommending them and make it difficult for the system to recommend novel items that have less usage data available. Niemann and Wolpers [11] discussed a collaborative approach to increase the aggregate diversity of recommender systems. This work is used to recommend niche or novel items to a user that are usually ignored by recommender systems. They employed a context-based collaborative filtering approach where an items' relevance is evaluated by the items it frequently is associated with rather than depending on user data associated with the item.

In contrast to user-based systems, in item-based recommender systems, the association or similarity of items are measured by comparing vectors for the items in which each dimension is a user rating. The similarity between two items is measured by calculating the cosine similarity between their co-occurrence vectors. This approach is used by Sarwar and Karypis [12] to recommend movies to users.

As mentioned previously, because of the high-dimensionality of the user and item vectors data sparsity and scalability are the two issues that greatly affect the performance of collaborative recommender systems. To overcome these problems, some dimensionality reduction is needed to decrease the number of items in the user vectors by grouping items and/or reducing the number of users in the item vectors by grouping users. Gong [13] proposed a collaborative filtering approach that does both. He combines a user-based collaborative recommender systems and an item-based collaborative recommender systems. In his approach, users and items are clustered to reduce the dimensionality, and combine the data, in each vector.

User clusters are formed based on the similarity between the user vectors (vectors of ratings given by the users to items). User vectors are clustered into groups using the k-means algorithm in this approach. Once the user clusters are created, the values of each missing user-item pairs in the cluster are calculated by averaging the opinion of the rest of the users in the cluster. This allows the user vectors to receive values for items they have not liked explicitly, addressing the data sparsity problem. If a user belongs to several clusters, the value for a missing item can be measured by averaging the opinions of the users across the clusters weighted by the degree of participation of the user in the cluster.

Similarly, item clusters are formed based on the similarity between the item vectors (vectors of purchases/likes with dimensions for each user). Items are clustered into groups based on the k-nearest neighbors algorithm where the nearest neighbors are identified based on the value of the cosine similarity with a target item. The value of the rating of a user for an item is calculated by using the weighted average of the user's ratings in the neighbors' item vectors.

An experiment was carried out using the data set from Movie Lens, [14] a movie recommender system. Users were selected randomly to create a test set of 100,000 ratings from all users with each user contributing at least 20 ratings. The ratings for an item were given on a scale of 1-5. Their results over several different metrics demonstrated that this approach performed better than the traditional collaborative recommender systems that did not employ clustering to reduce dimensionality.

2.2 News Recommender Systems

News recommender systems are designed to recommend relevant news articles for a user. Their role is significant, as there is a huge corpus of information available that is constantly changing and the users may face difficulty wading through the flood of information to find those news items of most interest for him. They are generally classified into either the content-based or collaborative recommender systems. The content-based news recommender systems make use of the profile of the user and the content of the articles to recommend news articles. Collaborative news recommender systems depend on the crowdsourcing data to recommend news articles. In the following sections, we will discuss both the content-based and popularity-based recommender systems based on collaborative information.

2.2.1 Content-Based News Recommender Systems

These recommender systems use the profile of the user to recommend articles relevant to him. The construction of a given user's profile can either be explicit or implicit. In explicit approach, the user would be asked3 to express his interests and based on that his/her profile would be determined. In the implicit approach, the user's profile would be determined by the user's actions. The user's actions can range from clicking on a news article, rating news articles, the amount of time spent on accessing a news article, recommending of news articles to other users in social media, etc.

Bouras and Tsogkas [15] developed a content-based news recommender system that uses the content present in the news articles and profile of the user to recommend news articles to a user. The articles were processed and the keywords present in the article were weighted by the tf*idf approach. The profile of the user was built using an implicit approach based on the keywords present in articles viewed by the user, time spent by the user reading the articles, and the articles avoided by the user. Experiments conducted on 30 users with a dataset of 50 articles proved a significant increase in the precision and recall values.

2.2.2 Hybrid News Recommender Systems

Hybrid news recommender systems combine content-based and collaborative information approaches to recommend news articles to the user. Hybrid systems can differ from one other in the way they implement the fusion of the content-based and the collaborative information based components.

Liu and Dolan [19] worked on developing a news recommender system that built the profile of a user on the basis of his click behavior and determined the popularity of the news stories based on the current news trends. They employed an implicit feedback approach to build the user's profile. Their proposed system had considered 22 different categories of articles, analyzed the information from the Google news log to find out how many times a user clicked on articles belonging to a certain category for each month and built the profile of the user from that information.. In addition to this, they also analyzed the effect of current news trends on the relevancy of the recommended articles to a user. They developed a Bayesian network that combines the profile of the user and the current trends in the news to form a hybrid system. The experiment was conducted on 100,000 people showed that the hybrid system performed better.

Morales and Gionis [20] implemented a hybrid recommender system using the contents of tweets sent by a given user to build their user profile. If no tweet content was available for a user, the tweet content from his social circle was used to create the personal profile for the user.   They also used the tweet contents sent by members of the user's social circles and Yahoo toolbar tags to gather popularity information about online news articles available from Yahoo.

2.2.3 Popularity-Based News Recommender Systems

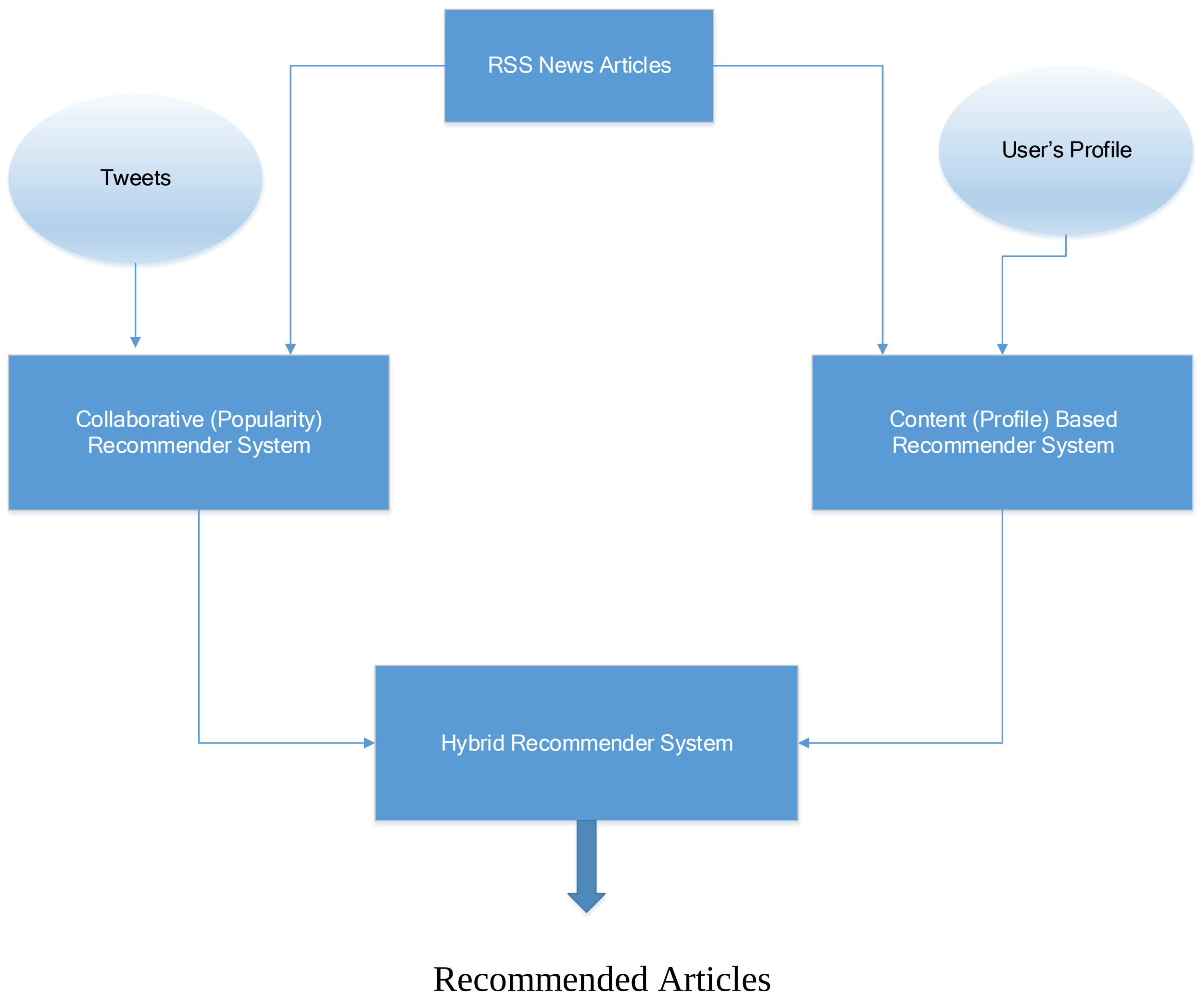
Popularity-based recommender systems rely mostly on the collaborative data to determine the order of the articles that are to be recommended to a user. Most of the existing recommender systems use data from social media such as Twitter, Facebook, and Tumblr to determine a news article's popularity. With the soaring popularity of social media, news events occurring around the world are discussed instantaneously by many users. Using trending data from sites such as Twitter, the popularity of the articles can be determined in a very reliable way.

Most of the popularity-based recommender systems are actually hybrid systems that combine the popularity data with user profiles. For example, Buzzer [18] is a news recommender system developed by Phelan and McCarthy that uses the current trending data in twitter and also the tweets generated by a user and his social circle to determine the order of news articles to be recommended to the user.

3. APPROACH

3.1 High Level Design

In this section, we present the high level view of our system that is comprised of the Collaborative Recommender System, the Content-Based Recommender System and the Hybrid Recommender System.



RSS News Articles
Tweets
User's Profile
Collaborative (Popularity) Recommender System
Content (Profile) Based Recommender System
Hybrid Recommender System
Recommended Articles

Figure 1. Architecture of the Recommender System [1]

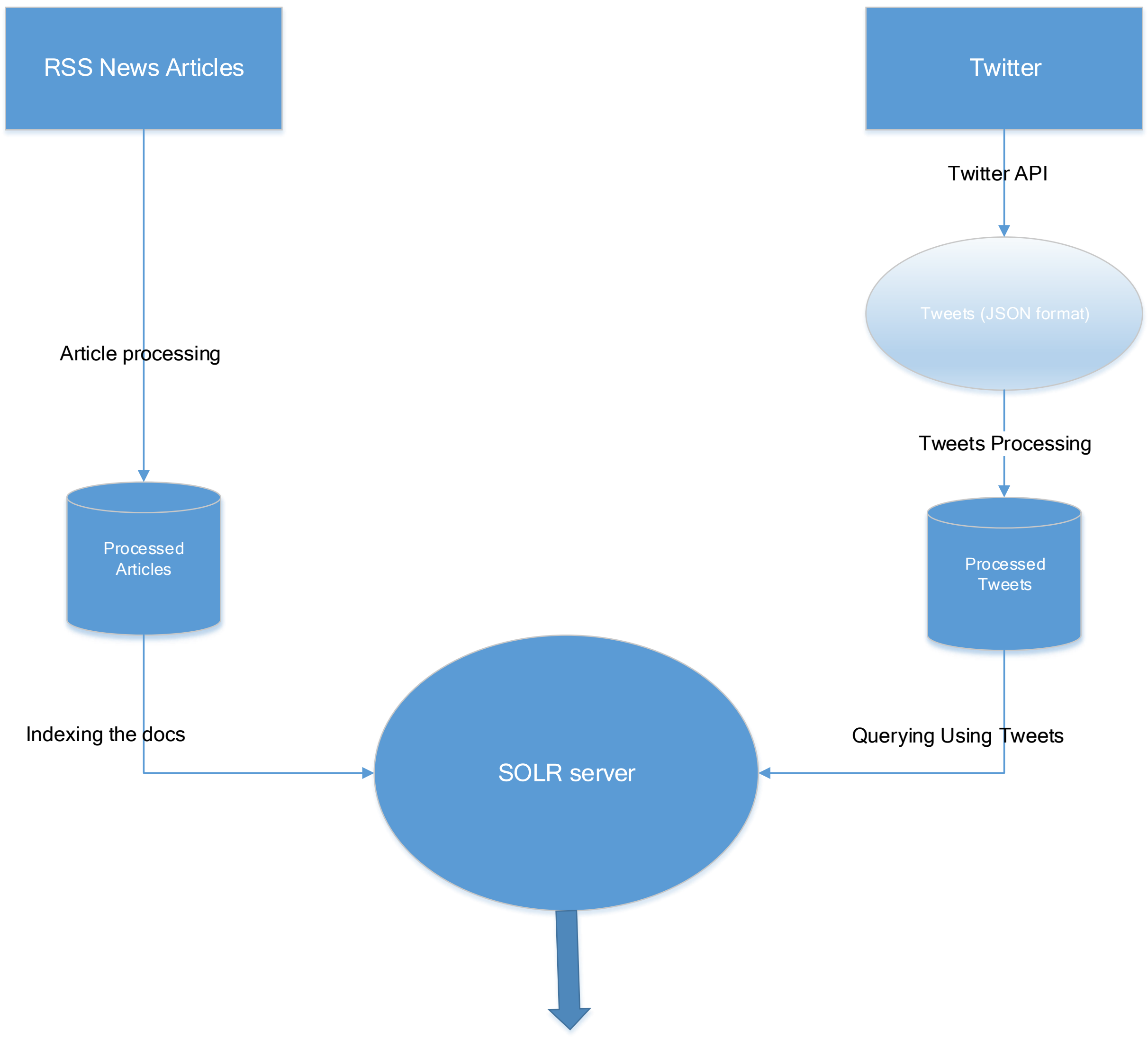
The proposed system consists of three modules.

1. Collaborative (Popularity) recommender system
2. Content-Based (Profile) Recommender system
3. Hybrid Recommender System

The collaborative information based popularity recommender system is implemented using user data from Twitter public timeline to recommend news articles from RSS news feeds from various sources such as Reuters, CNN, New York Times, and Yahoo. The popularity of an article is determined by measuring the similarity between the content present in the article and the collection of tweets. The content-based recommender system is implemented by allowing the user to explicitly rate sample documents, constructing the user's profile automatically based on these ratings, and then recommending future articles based on their similarity to the user profile. The Hybrid system is a fusion of the Collaborative recommender system and the Content-based recommender system. In the upcoming sections, we will discuss the implementation of each of these components in detail.

3.2 Collaborative (Popularity-Based) Recommender System

The high-level view of the system is as below:



Recommended Stories

Figure 2. Architecture of the Collaborative (Popularity-Based) Recommender System

As discussed previously, the Popularity-Based Recommender System collects data from the popular social media platform Twitter. It matches the collected tweets to articles collected from an RSS news feed to identify which news stories are most tweeted about, i.e., the most popular. Building the Popularity based recommender system involved three steps.

1. Collecting news articles whose popularity is to be determined and processing them to remove unwanted noise.
2. Collecting tweets from the Twitter public timeline and processing them to remove the unwanted noise.
3. Calculating article popularity by matching the tweets to the news articles and accumulating scores to determine how much each news article is discussed on Twitter.

Collecting News Articles

We wrote a module in Java that is able to collect news articles from Reuters, CNN. It is given the URL of the categories in the website as a parameter. The collected articles are automatically stored in a file system that organizes the news article collection into different folders based on their respective categories. We also wrote a preprocessor in C++ and LEX that removes noise from the text. It downcases all letters, removes punctuation, html tags, and numbers.
Finally, the module then uploads preprocessed articles to a SOLR server [21]. Apache SOLR is a java platform that constructs an inverted index, called a *Lucene* index on a collection of text. It incorporates many features such as text highlighting, full-text search, dynamic clustering, and database integration. SOLR also provides REST services such as HTTP, XML and JSON API. Our system uses the Java library present at its core to create a Lucene index on the articles. This index maps from keywords to articles to provide fast searching.

Collecting Tweets

This module, implemented in Python scripting language, collects tweets from the social media service Twitter. The module uses Twitter’s streaming API to collect currently streaming tweets from the Twitter public timeline. The Twitter streaming API can be used to collect specific tweets based on a keyword given as a parameter or it can collect every tweet from the Twitter public timeline. In our case, it collects every live streaming tweet from the Twitter public timeline. Our module is given the number of tweets to be collected as a parameter. The Streaming API implemented in Python listens on the Twitter public timeline until the specified number of tweets mentioned in the given parameter are collected. All collected tweets are stored in a single file that must be parsed to extract individual tweets. The tweets collected from the Twitter API were in JSON format that contains several unwanted fields such as date, time zone and retweet score as well as the desired tweet content. Our module then parses the collection of tweets to split them into individual tweets and extract each tweet's textual contents. Each tweet is then preprocessed identically to the preprocessing done on the news articles and the resulting clean tweets are then stored one per line in the final file.

Calculating Article Popularity

The third module calculates each news article's popularity score by determining how much discussion about the news article there is on Twitter. This module is implemented in Python, LEX and C++. Given the tweets collected by the previous module, it submits each as a query to the SOLR server. Based on the Lucene index built for news article collection, SOLR returns a rank-ordered list of the top matching articles and a score that reflects the similarity between the query (tweet) and the article. For each tweet, the scores are extracted and the similarity score is added to the corresponding article's popularity score. Thus, after all the tweets are processed, each article has a popularity score that is the accumulated total of similarity scores across all tweets in the collection.

In reality, every article has a similarity score for each tweet (although those scores will be 0.0 if there are no words in common). For scalability, when we work with large article collections, we accumulate the scores of only the top 10 most similar articles for each tweet when calculating the popularity scores.

The calculation of the popularity score of an article using a collection of tweets is explained with the following example. Let us consider a collection of five processed tweets and five processed news articles. Tweets are represented by T1, T2, T3, T4, T5 and articles are represented by A1, A2, A3, A4 and A5. First, the documents are uploaded to, and indexed by, the SOLR server. After the SOLR server is queried with each tweet in the collection, similarity scores for each document, for each tweet, are extracted as shown in Table 1.

Table 1: Article-Tweet Similarity Scores from SOLR

Article/Tweet	T1	T2	T3	T4	T5	Total
A1	0.3		0.4		0.7	1.4
A2		0.8		0.9		1.7
A3	1.2	0.2	0.6		0.4	2.4
A4	0.2	0.7			0.3	1.2
A5	0.4		0.3	0.5		1.3

As is true for all vector space search engines, the similarity between a query and a document is calculated using the cosine similarity measure, essentially the inner product between query vectors and document vectors where each unique token in the document collection is a dimension in the vector space. By normalizing the document vectors with respect to length during indexing, SOLR calculates the inner product by simply doing a dot product between the query vector (tweet) and document vector (news article). In actuality, these vectors are all sparse (most words do not occur at all and thus have weight 0.0). SOLR exploits this fact to implement the similarity calculations efficiently.

Similarity Weight (Tweet, Article) =Tweet Vector. Article Vector

Figure 3. Similarity Weight Formula

The Popularity Weight for an article is simply the accumulation of the Similarity Weights for that article accumulated over all tweets in the collection. In our example, the Popularity Weight of the article A1 would thus be 0.3+0.4+0.7 =1.4. Table 2 shows the Popularity Weights for all the articles in our example.

Table 2: Articles & PopularityWeights

Article	PopularityWeight
A1	1.4
A2	1.7
A3	2.4
A4	1.2
A5	1.3

The Collaborative Recommender System then sorts the articles in decreasing order by Popularity Weight and recommends the articles to the user in the order shown in Table 3, i.e., article A3, then article A2, etc.

Table 3: PopularityWeight Based order of Articles [1]

Article	PopularityWeight
A3	2.4
A2	1.7
A1	1.4
A5	1.3
A4	1.2

As illustrated by the example, every article has a *PopularityWeight* (although some may be 0.0 if no tweets match the article). The number of recommendations to display to the user at any time is a parameter given to the recommender system.

3.3 Content (Profile)-Based Recommender System

Based on previous research [1], we use a conceptual approach for our Content-Based Recommender System. Thus, we build a user profile that is represented as a set of weighted categories rather than as a vector of weighted keywords. This higher level of abstraction has been shown to be effective by since it represents user interests at a higher level of abstraction and is less sensitive to the ambiguity present in natural language. An overview of our Content-Based Recommender System is shown in Figure 4.

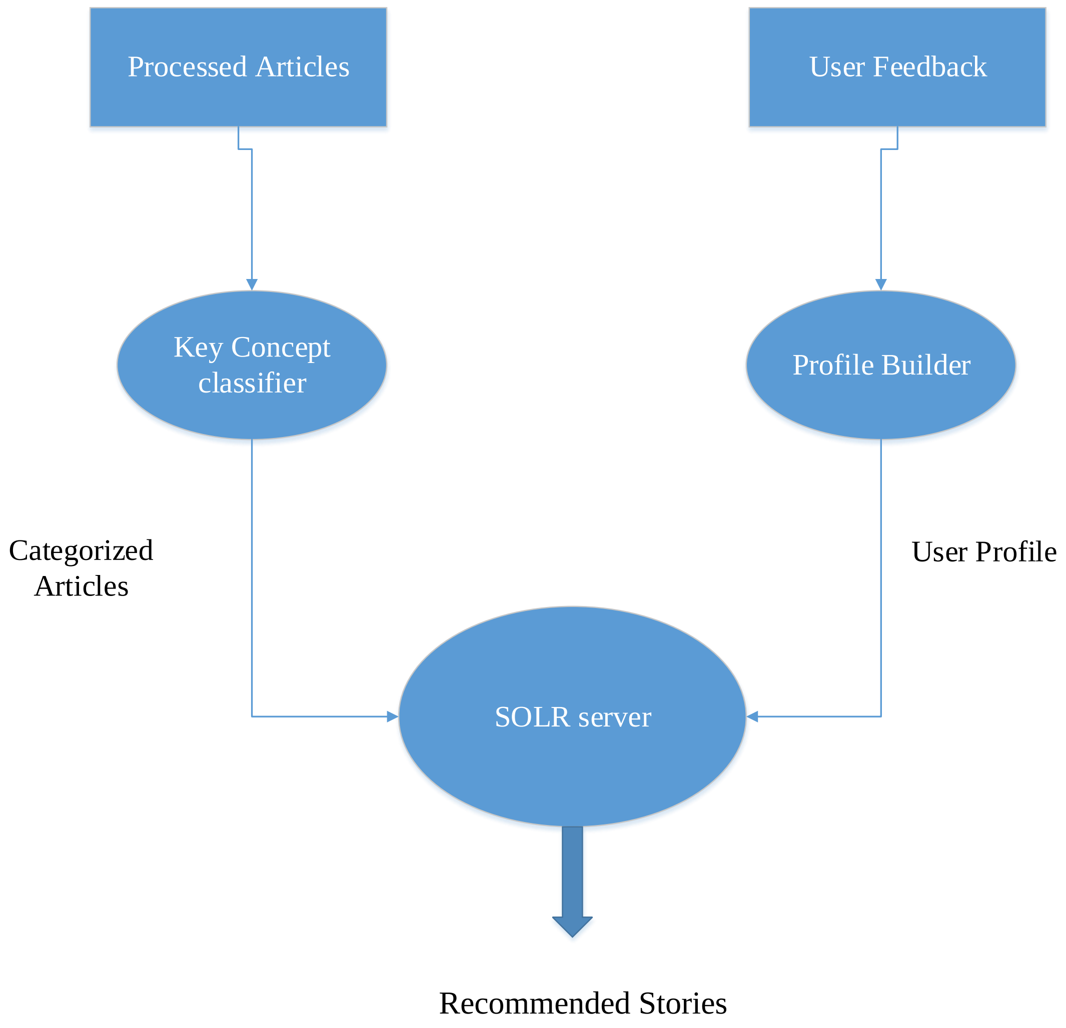


Figure 4. Architecture of the Content (Profile) Recommender System

The implementation of the content-based recommender system contains the following modules:

1. The KeyConcept text classifier trained for each of the news categories.
2. A Profile Builder module that collects user ratings on sample documents and automatically constructs the user’s profile from that feedback.
3. The SOLR server that is given the user’s profile and categorized documents and makes recommendations to the user based on their similarity.

KeyConcept Classifier

Our conceptual recommendation approach represents user profiles, and documents, as a set of weighted categories. A key component of this system is a classifier that can map documents in the collection to categories. The first step in this process is to select a set of categories that cover the major topics in the collection. The number of categories used must be sufficient to cover the breadth of the field without being so numerous as to artificially scatter related news stories. To guide us, we studied the categories used by human editors for the publication of online news. Thus, for our news recommender system, we focused on categories that are common across multiple news sites. We studied the categories used by Reuters and CNN and identified 15 categories that were in common on those sites. The categories selected were:

- Sports
- Politics
- Business
- Life Style
- Health
- Entertainment
- World
- Money
- Science
- Technology
- Arts
- People

- Crime
- Environment
- Economy

Since not all sites used the same set of categories, or indeed any categorization at all, and none of the tweets are categorized, we need a classifier to map news articles and/or tweets to these categories. To achieve this, we used the KeyConcept classifier module [1] that implements a k-nearest neighbor's algorithm. The K-nearest neighbor's algorithm first needs to be trained to identify features and weights for each category. It does this by extracting vocabulary and weights from a set of training documents, documents pre-labeled as belonging to one of the pre-selected categories. In our case, we trained the classifier with fifteen documents for each of the fifteen different categories. These documents were downloaded from Reuters from the appropriate categories as determined by the site's editor.

After the training, new textual content can be classified by comparing features extracted from the new content to features learned from the training documents. When a document is classified, the KeyConcept classifier returns a list of category identifiers and weights representing the match between the document and the category. This list is returned in decreasing order by match. The module takes as a parameter the number of matching categories to return.

In our system, we use the same processed news articles collected from an RSS feed as the Collaborative Recommender System. The processed articles are submitted to the KeyConcept classifier to identify the top three categories and their match values for each article. These categories and their match weights are then stored in the SOLR server for use when making recommendations. Essentially, in addition to representing articles as vectors of keywords and weights, we have SOLR also builds a second index that represents the articles as vectors of categories and weights.

Table 4: Articles & Category scores using K-neighbor's algorithm

Article	Sports	Crime	Entertainment	Politics	Science
A1	0.3		0.4		0.7
A2		0.8	0.1	0.9	
A3			0.6	0.8	
A4		0.7	0.2		0.3
A5	0.4		0.3		

Table 4 continues our example by showing representative categories and match weights from by the KeyConcept classifier for our five sample articles. To save space, the table only shows five categories, whereas the actual implementation makes use of fifteen categories. We can also observe from the table that match scores have been listed for only two categories for article A3. This indicates that the contents of the document A3 has been found similar to only two categories by the KeyConcept classifier.

Profile Builder

The previous content-based recommender system allowed a user to enter a value on a scale of 1-10 to indicate his level of interest in a category. The total weight of all categories was 10, and the user was asked to distribute numbers adding to 10 across the five categories used by that

system. For example, a user could enter 6 for sports and 4 for entertainment to indicate his preference for those categories. The 6 and 4 scores for the categories ‘Sports’ and ‘Entertainment’ respectively, represent the profile of the user.
We replaced this manually profile construction process with a module that automatically builds the user profile based on user preferences. We also extended the profile to be more accurate, by increasing the number of categories used from seven to fifteen.

Our Profile Builder is first tasked with collecting user feedback that captures their preferences. To do this, we collected five representative articles for each of the fifteen different categories. These articles were randomly selected from the fifteen training documents for the category. To avoid bias, the user was presented with this collection of 75 articles in a random order. The category labels for the documents were not shown to the user. We developed a web-based user interface that presents these feedback articles to the users and collects their ratings. Users may rate based on the article titles or click on the titles to view the entire article. As shown in Figure 5, the user then uses a radio button to rate the article as *Not Interesting*, *Interesting*, or *Very Interesting*. Once all articles are rated, the user clicks the *Submit* button to record their feedback.

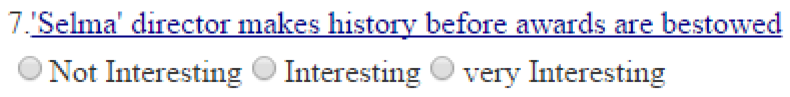

7.'Selma' director makes history before awards are bestowed
○ Not Interesting ○ Interesting ○ very Interesting

Figure 5. Article and Options in Profile Builder

The Profile Builder, upon receiving the user’s feedback, thus has five ratings for articles in each of the fifteen categories. It must next turn these ratings into category weights in the user's profile. As a first step, ratings are mapped to numbers as follows: the rating 'Not Interesting' is assigned the value 0, the rating 'Interesting' is assigned the value 1 and the rating 'Very Interesting' is assigned with the value 2.

The calculation behind the generation of a user's profile is as follows. The total sum of the ratings assigned by the user for all of 75 articles is calculated. This total will be used later to normalize the individual category scores. Let it be represented by *TotalRating*. For each category, the sum of the ratings given by the user for all articles within that category is also calculated. Let it be represented by *CategoryRating*. The level of interest of a user towards a particular category is represented by the *CategoryScore*, the *CategoryRating* normalized by the *TotalRating*. It is multiplied by 100 so that it is a percentage. The formula is shown in Figure 6.

*CategoryScore= (CategoryRating*100) / (TotalRating)*

Figure 6. CategoryScore Calculation

Consider the case of a user interested only in Sports. He would gives the highest rating 'Very Interesting' to each of the five articles from the 'Sports' category and the lowest rating 'Not Interested' to all of the articles from the rest of the categories. His user profile would be calculated as follows:

> *TotalRating* would be equal to 10 since there are 5 different 'Sports' articles and the user gave the highest rating of 'Very Interesting' (2) for each article.
>
> *CategoryRating* for the 'Sports' category would also be equal to 10.

CategoryScore for the 'Sports' category would be (10*100)/ (10) = 100. The *CategoryRating* for the remaining categories would be equal to 0 and hence their *CategoryScores* would also be 0.

A sample user profile built with the above mechanism is shown in Table 5. Note that the sum of all category scores is 100, reflecting the fact that the category scores are percentages of the total.

Table 5: Sample User Profile

Category	Category Score
Sports	0.00
Lifestyle	7.50
Business	10.00
Politics	5.00
Science	17.50
Health	7.50
Entertainment	5.00
World	20.00
Money	7.50
Technology	7.50
Economy	5.00
People	0.00
Arts	0.00
Crime	2.50
Environment	5.00

SOLR server

The third module in the Content-Based Recommender System is the same SOLR server used by the Collaborative Recommender System. In addition to building a Lucene index for the article contents for use by that system, the SOLR server is also used to build a Lucene index for the articles based on the category ids and weights provided by the KeyConcept classifier for use in this system.

To provide concept-based recommendations for a user, the user's profile is submitted to SOLR as a query and the top-matching documents are identified using the same cosine similarity measure used to match tweets to documents. In other words, SOLR's search mechanism calculates the dot product between the user's profile, treated as a vector of category weights, and each article, also represented as a vector of category weights stored in the Lucene index.   Figure 7 shows the formula used by SOLR to calculate the weight of an article with respect to a particular user profile. Finally, the articles are sorted and the articles are recommended to the user in decreasing order of the *PersonalWeight.*

$$\boxed{\textit{PersonalWeight} = \textit{Article Vector}.\ \textit{Profile Vector}}$$

Figure 7. PersonalWeight Formula

Let us consider an example of a user whose profile has a *CategoryScore* of 70 for ‘Entertainment’ and 30 for ‘Sports’. After applying the dot product, the *PersonalWeights* for the articles in our example are shown in Table 6.

Table 6: Articles and PersonalWeights

Article	PersonalWeight
A1	37
A2	7
A3	42
A4	14
A5	21

Based on these values, the conceptual Content-Based Recommender System would recommend the articles to the user in the order shown in Table 7.

Table 7: Concept-Based Recommendation Order

Article	PersonalWeight
A3	42
A1	37
A5	21
A4	14
A2	7

As illustrated by the example, every article has a *PersonalWeight* (although some may be 0.0 if all categories for an article match categories with weight 0.0 in the user profile). The number of recommendations to display to the user at any time is a parameter given to the recommender system.

3.4 Hybrid Recommender System:

The Hybrid recommender system fuses input from the Collaborative and the Content-based news recommender systems to recommend articles to the user. Both the *PopularityWeight* and the *PersonalWeight* of the article are used in determining the *HybridWeight*. This way, the hybrid system recommends articles that are of interest to the user that are also broadly popular. The relative contributions of the user's *PersonalWeight* and the global *PopularityWeight* are controlled by a tunable parameter, α.

$$HybridWeight=[\alpha*PopularWeight]+[(1-\alpha)*PersonalWeight]$$

Figure 8. HybridWeight Formula [1]

An α value of 1.0 causes the Hybrid recommender system to use only the *PopularityWeight* and thus causes it to operate as a purely Collaborative recommender system. Similarly, an α value of

0.0 causes the Hybrid recommender system to use only the *PersonalWeight* and thus causes it to operate as a purely Content-based recommender system.

Let us consider our example with five articles A1, A2, A3, A4 and A5 whose *PopularityWeights* and *PersonalWeights* are listed in the below table.

Table 8: Articles-PersonalWeights and PopularityWeights

Article	PersonalWeight	PopularityWeight
A1	37	1.4
A2	7	1.7
A3	42	2.4
A4	14	1.2
A5	21	1.3

After all article weights are calculated, we normalize the *PersonalWeights* by dividing by the maximum weight of any article. Thus, the top article receives a score of 1.0 and all other *PersonalWeights* are normalized relative to that value, between 0.0 and 1.0. In a similar way *PopularityWeights* of the articles are also normalized. Thus, normalization allows both the *PopularityWeights* and *PersonalWeights* of the articles to be represented on 0.0-1.0 scale. After normalization, the *PersonalWeights* and the *PopularityWeights* of articles from Table 8 are shown below in Table 9.

Table 9: Articles-Normalized PersonalWeights and PopularityWeights

Article	PersonalWeight	PopularityWeight
A1	0.8809	0.5833
A2	0.1666	0.7083
A3	1.00	1.00
A4	0.3333	0.50
A5	0.50	0.5416

The *HybridWeight* for each article is calculated by the formula mentioned previously. Table 10 shows the HybridWeights calculated using 0.7 for α, i.e., 70% of the *HybridWeight* is contributed by the *PopularityWeight* and 30% by the *PersonalWeight.* Applying the formula, the *HybridWeight* for the articles listed in Table 9 can be calculated as follows.

A1-*HybridWeight* = [(0.7*0.5833) + (0.3*0.8809)] =0.6725

A2-*HybridWeight*= [(0.7*0.7083) + (0.3*0.1666)] = 0.5457

The *HybridWeights* of the articles A1 through A5 from table 9, are listed below in table 10.

Table 10: Articles and HybridWeights

Article	HybridWeight
A1	0.6725
A2	0.5457
A3	1.00
A4	0.4499
A5	0.5291

Thus, the Hybrid system would recommends articles to the user in order based on by their *HybridWeights* as listed in the below table.

Table 11: Hybrid-Based Recommendation Order [1]

Article	HybridWeight
A3	1.00
A1	0.6725
A2	0.5497
A5	0.5291
A4	0.4499

In this way, the Hybrid recommender system recommends the articles to the user by fusing the *PopularityWeight* of the articles determined by the user data from the Twitter public timeline and the *PersonalWeight* of the articles determined by the user's profile. As illustrated by the example, every article has a *HybridWeight* (although some may be 0.0 if both the *PersonalWeight* and the *PopularityWeight* are 0.0). The number of recommendations to display to the user at any time is a parameter given to the recommender system. For the experiments described in Chapter 4, we show the users the top 10 recommended articles.

4. METHODOLOGY, RESULTS AND ANALYSIS

We designed an experiment to evaluate the effectiveness of our Hybrid Recommender system that uses our expanded set of categories and our new module that automatically builds user profiles.

4.1 Experimental Set Up

4.1.1 Dataset

Our experiment was run on a dataset of news articles and tweets collected on the same day, February 23, 2015. The same dataset was used for all experiments. We collected 375 news articles from the RSS feed of the Reuters news site, http://www.reuters.com/ and the CNN news site, http://www.cnn.com/, 25 articles for each of the 15 categories. We also used the Twitter streaming API to collect approximately 400,000 tweets from the Twitter public time line on the same day on which the news articles were collected. This allows us to match tweet activity to specific news stories. Both the news articles and the tweets were preprocessed to remove the unwanted noise. As described in Chapter 3, these preprocessed articles and tweets were indexed by SOLR. The articles were also classified by the KeyConcept classifier and the top three categories for each article were also indexed by SOLR.

4.1.2 Subjects

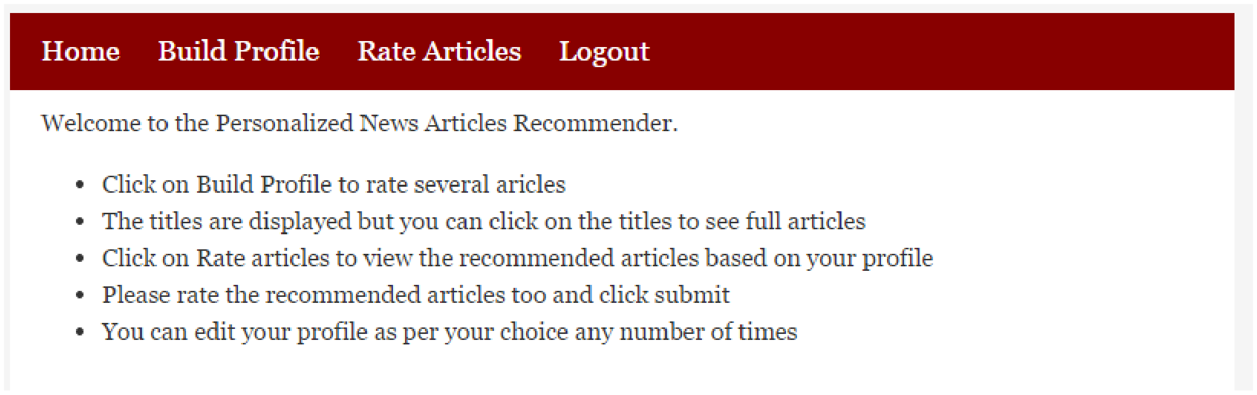
We employed 25 volunteers who participated in the experiment. These subjects were all graduate or undergraduate students at the University of Arkansas from various majors.

4.1.3 Method

The evaluation of the recommender systems by each subject involved the following steps:

1. User creates his/her individual account.
2. User explicitly rates the 5 evaluation articles for each of the 15 categories.
3. The Profile Builder builds the user's profile based on their feedback.
4. The Hybrid Recommender System generates results for α values from 0.0 to 1.0 in increments of 0.1, 11 sets of results in all. The set of results before are processed to remove duplicates and then presented in random order to the user for evaluation.
5. The user rates each article on the three-point scale ('Not Interesting', 'Interesting', or 'Very Interesting'.
6. After completing all ratings, the users submitted the results to the server for analysis.

The sample screen shot of the experimental user interface is shown in Figure 9.

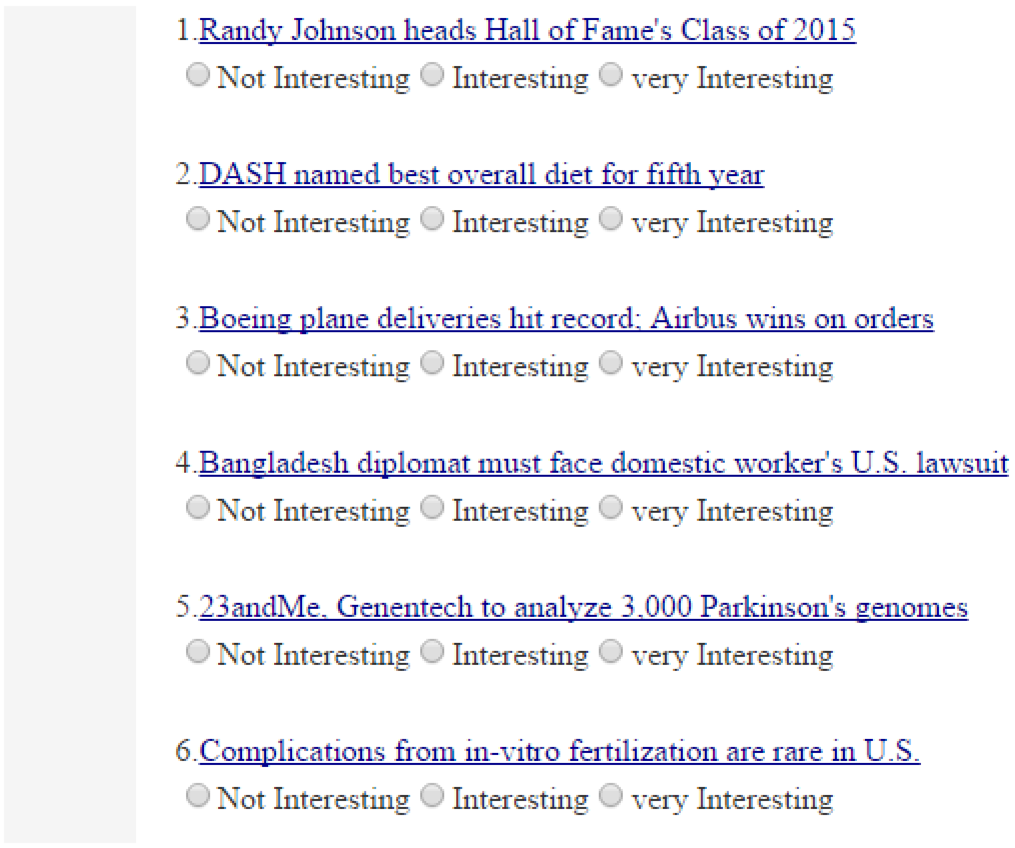


Welcome to the Personalized News Articles Recommender.

- Click on Build Profile to rate several aricles
- The titles are displayed but you can click on the titles to see full articles
- Click on Rate articles to view the recommended articles based on your profile
- Please rate the recommended articles too and click submit
- You can edit your profile as per your choice any number of times

Figure 9. Screen shot of the Experiment Interface

As can be seen from that figure, subjects were provided with links to build their profile and, after that, to rate the articles. Figure 10 show a sample screenshot of the articles presented to a user for rating. Users can read the title and also click on the title to read the entire article.



1.Randy Johnson heads Hall of Fame's Class of 2015
○ Not Interesting ○ Interesting ○ very Interesting

2.DASH named best overall diet for fifth year
○ Not Interesting ○ Interesting ○ very Interesting

3.Boeing plane deliveries hit record; Airbus wins on orders
○ Not Interesting ○ Interesting ○ very Interesting

4.Bangladesh diplomat must face domestic worker's U.S. lawsuit
○ Not Interesting ○ Interesting ○ very Interesting

5.23andMe, Genentech to analyze 3,000 Parkinson's genomes
○ Not Interesting ○ Interesting ○ very Interesting

6.Complications from in-vitro fertilization are rare in U.S.
○ Not Interesting ○ Interesting ○ very Interesting

Figure 10. Sample image of Articles Recommended to a User

For the experiment, we collected the top 10 articles for each of the 11 values of **α**, 110 results per user. After duplicate removal, each subject had to rate 24 unique articles on average. To remove any sort of bias from a user towards a particular approach, we presented these articles to the user in a random order.

The results from the experiment were stored in a database table with each row containing the Username, AlphaValue, Rank, DocumentId and the Rating given by the user for the article. Table 12 shows a subset of the results for a sample user.

Table 12: Sample Data of the User Evaluation

Username	alpha value	Rank	Document id	Rating
User1	0.4	1	703	0
User1	0.4	2	513	2
User1	0.4	3	1017	1
User1	0.4	4	209	1
User1	0.4	5	1224	1
User1	0.4	6	1012	2
User1	0.4	7	1018	2
User1	0.4	8	1008	2
User1	0.4	9	707	1
User1	0.4	10	1506	1

4.1.4 Metric

To analyze the results, we employed the *Mean Average Weighted Precision* (*MAWP*) metric. Precision measures the accuracy of recommendations, but does not take order into account:

Precision = (Number of Relevant Retrieved Results)/(Total Number of Retrieved Results)

Figure 11. Formula for Precision

In contrast, *Average Precision* (*AP*) places emphasis on the order in which highly-rated documents are recommended to the user. It is the average of precision calculated after each result. See Table 13

Table 13: Example demonstrating Evaluation of Average Precision

Article	Relevant	Precision
1	Yes	100%
2	No	50%
3	Yes	66.66%
4	No	50%
5	Yes	60%
	Average Precision	65.33%

The *Mean Average Precision* (*MAP*) is just the mean of the AP calculated over multiple queries and/or users. These metrics all work with Boolean user feedback (Relevant/Not Relevant). We extend these metrics to deal with trinary weights (0, 1, 2) as follows:

WeightedPrecision= (Relevant Results Weight)/(Total Weight)

Figure 12. Formula for Weighted Precision (WP)

Table 14 below demonstrates with an example, the calculation of Average Weighted Precision values.

Table 14: Example Demonstrating Calculation of MAWP

Results	RelvantWeight	TotalWeight	WeightedPrecision
1	0	2	0
2	2	4	50%
3	1	6	50%
4	2	8	62.5
5	0	10	50%
		Average Weighted Precision=	43%

The *Mean Average Weighted Precision* (*MAWP*) is just the mean of the AWP calculated over multiple queries and/or users.

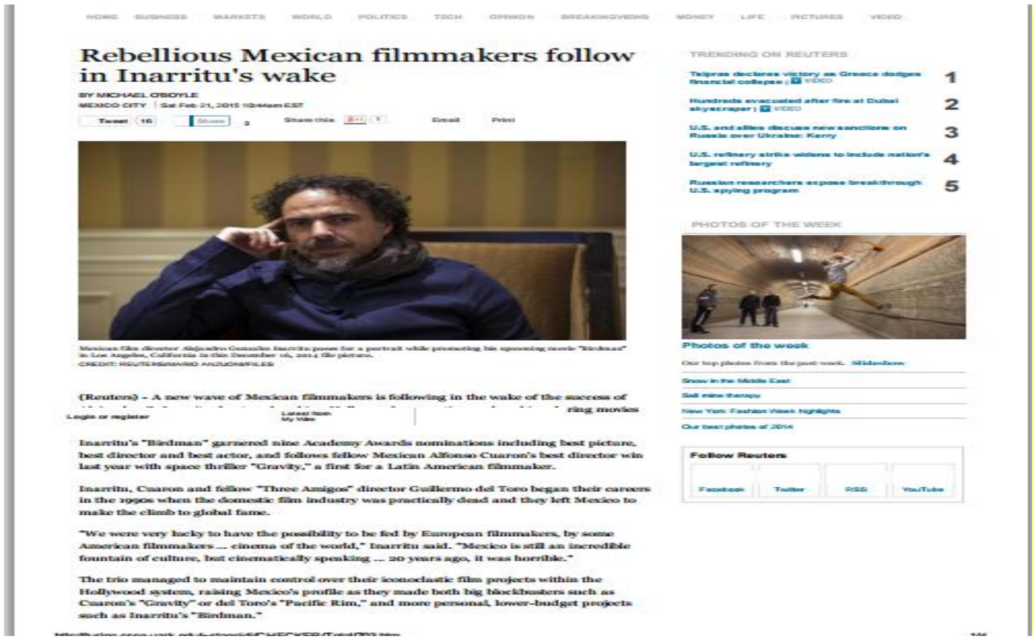
4.2 Experimental Results

4.2.1 Collaborative Recommender System

The results from the collaborative, popularity-based recommender system are those created when α is 1.0, i.e., the PersonalWeight is ignored. Thus, the recommendations from this system are the same for all users. Table 15 lists the top ten articles determined by the Collaborative Popularity-Based Recommender System along with their respective normalized scores.

Table 15: Top Articles and PopularityWeights

Articles	Normalized wt.
703	1
114	0.898199
1214	0.884259
1308	0.830523
513	0.789777
209	0.71566
1215	0.681528
313	0.499683
602	0.458851
109	0.453595
510	0.452886



Rebellious Mexican filmmakers follow in Inarritu's wake

BY MICHAEL O'BOYLE
MEXICO CITY | Sat Feb 21, 2015 10:44am EST

Mexican film director Alejandro Gonzalez Inarritu poses for a portrait while promoting his upcoming movie "Birdman" in Los Angeles, California in this December 16, 2014 file picture.
CREDIT: REUTERS/MARIO ANZUONI/FILES

(Reuters) - A new wave of Mexican filmmakers is following in the wake of the success of ... ring movies

Inarritu's "Birdman" garnered nine Academy Awards nominations including best picture, best director and best actor, and follows fellow Mexican Alfonso Cuaron's best director win last year with space thriller "Gravity," a first for a Latin American filmmaker.

Inarritu, Cuaron and fellow "Three Amigos" director Guillermo del Toro began their careers in the 1990s when the domestic film industry was practically dead and they left Mexico to make the climb to global fame.

"We were very lucky to have the possibility to be fed by European filmmakers, by some American filmmakers ... cinema of the world," Inarritu said. "Mexico is still an incredible fountain of culture, but cinematically speaking ... 20 years ago, it was horrible."

The trio managed to maintain control over their iconoclastic film projects within the Hollywood system, raising Mexico's profile as they made both big blockbusters such as Cuaron's "Gravity" or del Toro's "Pacific Rim," and more personal, lower-budget projects such as Inarritu's "Birdman."

TRENDING ON REUTERS

1. Tsipras declares victory as Greece dodges financial collapse | VIDEO
2. Hundreds evacuated after fire at Dubai skyscraper | VIDEO
3. U.S. and allies discuss new sanctions on Russia over Ukraine: Kerry
4. U.S. refinery strike widens to include nation's largest refinery
5. Russian researchers expose breakthrough U.S. spying program

PHOTOS OF THE WEEK

Photos of the week

Our top photos from the past week. Slideshow

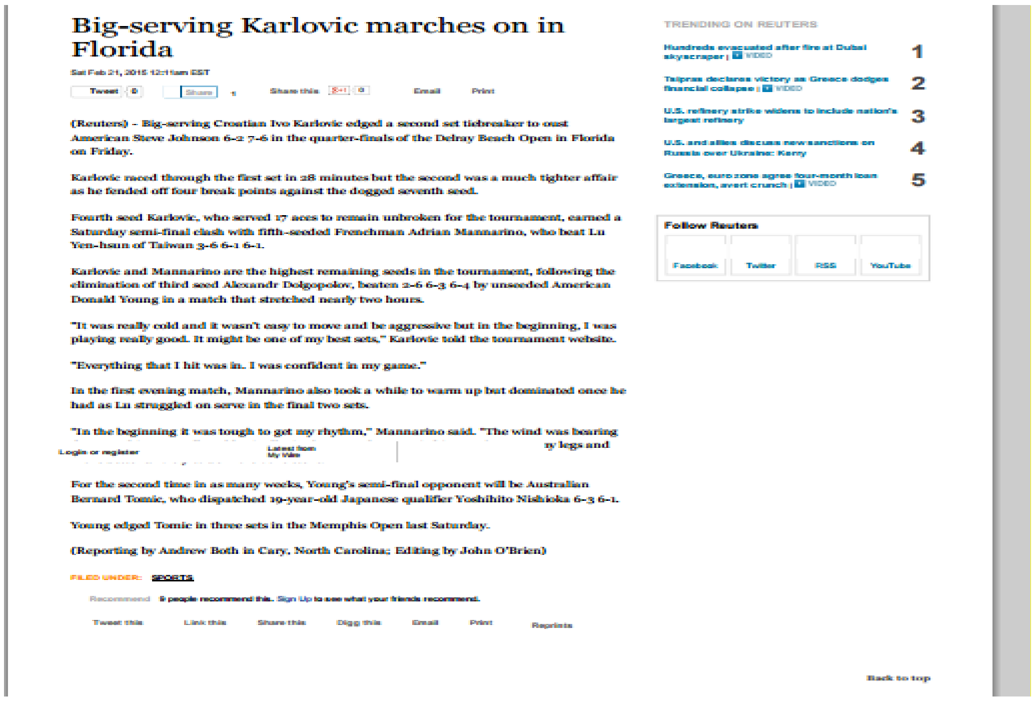
Snow in the Middle East

Salt mine therapy

New York Fashion Week highlights

Our best photos of 2014

Follow Reuters



Big-serving Karlovic marches on in Florida

Sat Feb 21, 2015 12:11am EST

Tweet 0 Share 1 Share this 0 Email Print

(Reuters) - Big-serving Croatian Ivo Karlovic edged a second set tiebreaker to oust American Steve Johnson 6-2 7-6 in the quarter-finals of the Delray Beach Open in Florida on Friday.

Karlovic raced through the first set in 28 minutes but the second was a much tighter affair as he fended off four break points against the dogged seventh seed.

Fourth seed Karlovic, who served 17 aces to remain unbroken for the tournament, earned a Saturday semi-final clash with fifth-seeded Frenchman Adrian Mannarino, who beat Lu Yen-hsun of Taiwan 3-6 6-1 6-1.

Karlovic and Mannarino are the highest remaining seeds in the tournament, following the elimination of third seed Alexandr Dolgopolov, beaten 2-6 6-3 6-4 by unseeded American Donald Young in a match that stretched nearly two hours.

"It was really cold and it wasn't easy to move and be aggressive but in the beginning, I was playing really good. It might be one of my best sets," Karlovic told the tournament website.

"Everything that I hit was in. I was confident in my game."

In the first evening match, Mannarino also took a while to warm up but dominated once he had as Lu struggled on serve in the final two sets.

"In the beginning it was tough to get my rhythm," Mannarino said. "The wind was bearing

Login or register Latest from My Wire

ny legs and

For the second time in as many weeks, Young's semi-final opponent will be Australian Bernard Tomic, who dispatched 19-year-old Japanese qualifier Yoshihito Nishioka 6-3 6-1.

Young edged Tomic in three sets in the Memphis Open last Saturday.

(Reporting by Andrew Both in Cary, North Carolina; Editing by John O'Brien)

FILED UNDER: SPORTS

Recommend 8 people recommend this. Sign Up to see what your friends recommend.

Tweet this Link this Share this Digg this Email Print Reprints

TRENDING ON REUTERS

Hundreds evacuated after fire at Dubai skyscraper | VIDEO 1

Tsipras declares victory as Greece dodges financial collapse | VIDEO 2

U.S. refinery strike widens to include nation's largest refinery 3

U.S. and allies discuss new sanctions on Russia over Ukraine: Kerry 4

Greece, euro zone agree four-month loan extension, avert crunch | VIDEO 5

Follow Reuters

Facebook Twitter RSS YouTube

Back to top

Figure 13. Contents of the Top 2 Articles Recommended by the Popularity-Based Recommender System [22]

Although there were obviously many tweets globally discussing these articles to cause them to be rated as the most popular, they would clearly not be of interest to all users. Thus, we expect that a Hybrid system that incorporates individual user preferences would be more effective.

4.2.2 Results and Analysis

Table 16 lists the results from the ratings given by our 25 subjects for each value of **α**.

Table 16: Results Table

Alpha Values	MAWP	Recommender sys
0	49.67	Profile System
0.1	49.83	
0.2	50.63	
0.3	51.84	Hybrid System
0.4	49.06	
0.5	40.40	
0.6	32.53	
0.7	29.56	
0.8	27.74	
0.9	25.04	
1	23.57	Collaborative System

As we can see from the results, the Collaborative recommender system performed the worst, with a MAWP of 23.57. The Content-based personal recommender system was more than twice as effective, with a MAWP of 49.67 and the Hybrid recommender system fared the best with **α** at 0.3, producing a MAWP of 51.84.

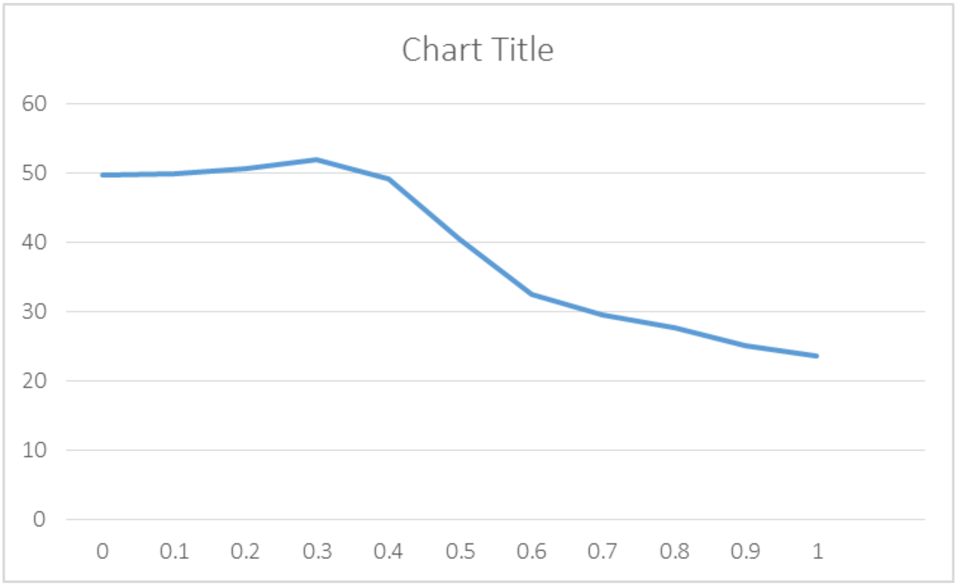


Figure 14. Performance of Recommender systems with respect to 'Alpha' value

Figure 14 presents the same data as Table 16, showing the results graphically. The α values are plotted along the X-axis and the corresponding MAWP values are plotted along the Y-axis. As we can see from the chart, accuracy of the articles was at the lowest when only popularity of the articles is considered (α=1.0). It rises steadily as the user's profile is also taken into account and peaks at α=0.3. As as the influence of an article's popularity is decreased further by decreasing α below 0.3, the MAWP also decreases. Thus, the performance is best when 70% of the contribution is from the user's profile and 30% is from the article's collaborative popularity.

Table 17: Recommender System Results Analysis

Rank	Popular System	Profile System	Hybrid System
1	26.00	46.00	52.00
2	22.00	47.00	49.00
3	20.64	47.77	50.66
4	20.72	48.58	50.87
5	22.08	48.94	51.02
6	23.34	49.23	51.29
7	23.72	49.22	51.64
8	23.58	49.31	51.84
9	23.48	49.46	51.80
10	23.55	49.66	51.78

Table 17 shows the results for the three systems, the Content-based recommender system, the Collaborative recommender system and the Hybrid recommender system with α=0.3 in more detail. Each row shows the MAWP values of the various approaches at each rank. We can see from the table that the Hybrid system performed better than the two component systems, particularly at the highest ranked documents.

We ran a student t-test on the results and found that the Content-based Profile system is statistically significantly better than the Collaborative Popularity-Based system ($p<0.000001$) and the Hybrid system with α=0.3 is statistically significantly better than the Popularity-Based system ($p<0.000001$) and statistically significantly better than the Profile-Based system ($p=0.000060$).

5. CONCLUSIONS

5.1 Summary

In this work, we created and then studied the ability of an automatically created user profile to produce accurate news recommendations delivered to each user. This Thesis also showed that the recommendations based on the user profile can be further improved by incorporating information about article popularity by including information from Twitter. The resulting News Recommender system is a fusion of a conceptual content-based recommender system and a collaborative popularity-based recommender system. This work is an extension of a previous news recommender that required uses to manually construction their profiles by explicitly entering a certain number to indicate his/her level of interest in a certain category. This approach is not completely reliable, as the user may not be able to accurately specify or represent his level of preference for a category with a number. We also extended the user profile to include 15 rather than just 5 categories of interest, to make it more accurate and broadly applicable.

In our work, we built the user's profile through feedback by allowing the user to explicitly rate articles from different categories. Based on the document ratings assigned by the user, our system automatically constructs their profile.

We ran an experiment with 25 users to evaluate our system. We measured the performance the conceptual, content-based recommender system, the collaborative recommender system, and the hybrid recommender system varying the contributions of the two subcomponents. We evaluated the results using Mean Average Weighted Precision. The Collaborative popularity-based recommender system achieved 24% MAWP, the conceptual, content-based recommender system achieved 50% MAWP and the Hybrid recommender achieved 52% MAWP in the recommendation of articles to the user. We were able to draw the following inferences from our experimental results.

1. The conceptual, content-based based recommender system performed significantly better than the collaborative popularity-based recommender system.
2. The hybrid recommender system significantly outperformed each of its subcomponent systems working alone.
3. The hybrid system was most accurate with α=0.3, i.e., 30% of the recommendation weight coming from popularity and 70% from matching the user's profile.

5.2 Future Work

Although our module automatically builds the user profile, users must still provide explicit feedback on news articles in order to provide input to the system. This module is an 46improvement of requiring users to enter numbers, but it is just the next step in the development of a system that removes this burden from the user. The next step in the development of this system is to develop a system that collects implicit feedback from the user as they read news. This module would "look over the user's shoulder" as they interacted with news websites through their browser or plug into their RSS feed newsreaders. By collecting information about which articles are read, bookmarked, saved, or shared, user's interest in various news articles could be implicitly determined. These articles could then be used as input to the profile builder, allowing the recommender to provide personalized recommendations without requiring any extra effort by the user.

In addition to implicit user feedback, we could explore ways of enhancing the user profile itself. Non-content-based features could be included such as the user’s location and changes in their areas of interest over time.

REFERENCES

[1] N.Jonnalagedda and S.Gauch, “Personalized News Recommendation Using Twitter”, in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Web Technologies*, Washington, DC, USA, 2013

[2] C.Leacock and M.Chodorow, “Combining local context and WordNet similarity for word sense identification,” in Fellbaum, pp. 265–283, 1998

[3] T.De Noia, R.Mirizzi,V.C. Ostuni, D.Romito and M.Zanker, “ Linked Open Data to Support Content-based Recommender Systems,” in *Proceedings of the 8th International Conference on Semantic Systems*, New York, New York,USA, 2012

[4] G.Semeraro, P.Lops, P.Basile and M.de Gemmins, “Knowledge Infusion into Content-based Recommender Systems,” in *Proceedings of the Third ACM Conference on Recommender Systems,* New York, New York,USA, 2009

[5] S.Loh, F.Lorenzi, G.Simoes, L.K.Wives and J.P.M.de oliviera, “Comparing Keywords and Taxonomies in the Representation of User Profiles in a Content-based Recommender System,” in *Proceedings of the 2008 ACM Symposium on Applied Computing,* New York, New York,USA, January, 2009.

[6] M.de Gemmis, P. Lops, G.Semeraro and P. Basile, “Integrating Tags in a Semantic Content-based Recommender,” in *Proceedings of the 2008 ACM Conference on Recommender Systems,* New York, New York,USA, 2008

[7] LOD website http://linkeddata.org/

[8] Y.Lu, S.Yu, T.Chang and J.Y.Hsu, “A Content-based Method to Enhance Tag Recommendation,” in *Proceedings of the 21st International Joint Conference on Artificial Intelligence,* San Francisco, CA, USA, 2009

[9] J.Wang, S.C.H.Hoi, P.Zhao and Z.Liu, “Online multi-task collaborative filtering for on-the-fly recommender systems,” in In *Proceedings of the 7th ACM conference on Recommender systems*, New York, NY, USA, 2013

[11] K.Niemann, and M.Wolpers, “A new collaborative filtering approach for increasing the aggregate diversity of recommender systems,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining,* New York, New York, USA, 2013

[12] B.Sarwar, G.Karypis, J.Konstan, and J.Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th international conference on World Wide Web,* New York, NY, USA, 2001

[13] S. Gong, “A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering,” in *Journal of Software, pages 745–752,* 2010.

[14] Movielens website https://movielens.org/

[15] C.Bouras and V.Tsogkas, “Personalization Mechanism for Delivering News Articles on the User's Desktop,” in *Fourth International Conference on Internet and Web Applications and Services, pages 157-162,* 2009

[16] X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” in *Advances in Artificial Intelligence,* New York, New York, USA, January 2009.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work,* Chapel Hill, NC, 1994

[18] O.Phelan, K.McCarthy, M.Bennett and B.Smyth, “On using the real-time web for news recommendation & discovery,” in *Proceedings of the 20th international conference companion on World Wide Web,* New York, New York, USA, 2011

[19] J.Liu, P.Dolan, and E.R.Pedersen, “Personalized news recommendation based on click behavior,” in *Proceedings of the 15th international conference on intelligent user interfaces,* New York, New York, USA, 2010

[20] G.D.F.Morales, A.Gionis, and C.Lucchese, “From chatter to headlines: harnessing the real-time web for personalized news recommendation,” in *Proceedings of the fifth ACM international conference on Web search and data mining,* New York, New York, USA, 2012

[21] Apache SOLR website http://lucene.apache.org/solr/

[22] Reuters website http://www.reuters.com/

[23] Twitter website, https://twitter.com/

[24] Amazon website, http://www.amazon.com/

[25] Google website, https://www.google.com/

[26] Yahoo website, https://www.yahoo.com/

[27] Netflix website, https://www.netflix.com/

[28] CNN website, http://www.cnn.com/

[29] Pandora website, http://www.pandora.com/

[30] Facebook website, https://www.facebook.com/