5-2013

Personalized News Recommender using Twitter

Satya Srinivasa Nirmal Jonnalagedda
University of Arkansas, Fayetteville

PERSONALIZED NEWS RECOMMENDER SYSTEM USING TWITTER

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

Satya Srinivasa Nirmal Jonnalagedda
Osmania University
Bachelor of Engineering in Computer Science, 2011

May 2013
University of Arkansas

ABSTRACT

Online news reading has become a widely popular way to read news articles from news sources around the globe. With the enormous amount of news articles available, users are easily swamped by information of little interest to them. News recommender systems are one approach to help users find interesting articles to read. News recommender systems present the articles to individual users based on their interests rather than presenting articles in order of their occurrence. In this thesis, we present our research on developing personalized news recommendation system with the help of a popular micro-blogging service "*Twitter*". The news articles are ranked based on the popularity of the article that is identified with the help of the tweets from the Twitter's public timeline. Also, user profiles are built based on the user's interests and the news articles are ranked by matching the characteristics of the user profile. With the help of these two approaches, we present a hybrid news recommendation model that recommends interesting news stories to the user based on their popularity and their relevance to the user profile.

This thesis is approved for recommendation
to the Graduate Council.

Thesis Director:

_____
Dr. Susan Gauch

Thesis Committee:

_____
Dr. Craig Thompson

_____
Dr. Bajendra Panda

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed _____
Satya Srinivasa Nirmal Jonnalagedda

Refused _____
Satya Srinivasa Nirmal Jonnalagedda

ACKNOWLEDGEMENTS

I thank the Lord, Balaji, for giving me opportunity and strength to apply myself in computer technology through out my graduate studies.

I sincerely thank my advisor Dr. Susan Gauch for her continuous dedicated support throughout my graduate research studies. I am very grateful to her for providing me the opportunity to work with her, and I will never forget her as she has been my inspiration as I hurdle all the obstacles in the completion of this research work. I appreciate Dr. Craig Thompson and Dr. Bajendra Panda for being on my thesis committte to advise and review my research work.

I also thank my parents, Vithal and Padma Jonnalagedda, and my brother, Kalyan Jonnalagedda, for their blessings and moral support that they have given me in my academic and other activities all the years of my life.

Last but not the least, I thank all my friends who made my Master's journey a memorable experience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Owing largely to the ever-increasing volume and sophistication of data on the web, we are able to access an enormous amount of information around the globe. The downside of this information explosion is that users are often swamped by information of little interest to them. The key challenge today is for the users is to find relevant information based on their interests. This problem has led to the evolution of the recommender systems that help users to find information they need based on their interests. Recommender systems proactively present users with information related to their interests rather than requiring the user to search for, and then filter through, information based on queries. Recommender systems can also be used to tailor the information presented by a website, e.g., a news site, to individual users. Thus, the stories at that site can be presented in an order based on the user's interests rather than using the same order of presentation for all users based on an editor's opinion.

Many organizations use recommender systems to recommend various types of products to the user. For example, Netflix recommends movies to its users based on their ratings of movies compared to other similar users' ratings. Amazon recommends various items such as gadgets, books, or movies and Pandora Radio recommends music based on a user’s past history and preferences. Additionally, news recommender systems that recommend news articles from around the globe have become popular. There are many online news reading services, such as Google News and Yahoo News. However, with so much news available, the driving problem is to identify and recommend the most interesting articles to each user so that they are not

presented with a flood of information to wade through. These articles should be related to each users interests but also include those news stories that are generating a lot of interest around the world.

News recommender systems generally present news articles to the user in two different ways. One is content-based filtering and the other is collaborative filtering. Content-based filtering methods are based on the information and attributes of the item that are going to be recommended. This approach focuses on analysing previous user interests to make future recommendations. Essentially, it recommends news articles whose contents are similar to the contents of previously read articles that the user has rated highly. Content-based filtering has some limitations. It can be difficult for the system to learn and understand the user preferences from the user's actions for certain types of items. From some types of items, it is possible to extract features that have semantic meaning. For items with associated text, e.g., news articles, the text can be mined to extract keywords that represent the items that the user has read and liked in the past. These keywords can be used to identify other news articles. However, for other types of items, e.g., music, there may not be any semantically related features that can be extracted. Thus, we cannot recommend items to the users based only on the contents of the user's chosen items themselves.

Collaborative filtering is a method based on information about the actions of other users whose preferences are similar to the existing user. Thus, collaborative filtering is based on the actions users on a set of items rather than features extracted from the items themselves. This approach studies the patterns of other users in order to recommend similar and interesting articles

to the existing user. The key advantage of this recommendation method is that it does not require the computer to analyze the data and hence it is capable of recommending content without the understanding the current content. One major concern is that these systems need a large amount of existing data from many users in order to make the recommendations more precise. Also, a large amount of computation power is necessary to process these recommendations. A popular service that uses the collaborative filtering technique is Amazon that uses item to item (people who buy 'x' also buy 'y') collaborative filtering to recommend products based on other user purchases. Another popular example for this technique would be the social networking sites such as Facebook and Google Plus that recommend new friends, groups and other social connections to the user.

Many news recommender systems have been created that use collaborative filtering, content-based filtering, or a hybrid of the two methods to help users find interesting articles to read.

In this thesis, we develop a hybrid personalized news recommender system that recommends interesting news articles to the user using a micro-blogging service "Twitter". Our recommender system ranks the news articles in different ways. We consider the user's profile to recommend articles to the user. This is essentially a content-based recommender system. In addition, we also consider the article's popularity with the help of tweets from the Twitter's public timeline, essentially a type of collaborative recommendations. We present a novel approach to help users find interesting articles to read by merging the above two methods of ranking articles.

The following are the objectives of this thesis:

1. To develop a popularity based recommender system using Twitter
2. To develop a personalized news recommender system using conceptual user profiles
3. To develop a hybrid recommender system that combines the two approaches
4. To compare and analyze all the three strategies

1.2 Organization of this Thesis

Chapter 2 covers the literature review of this work and discusses the work of others in this field. Chapter 3 explains the design and implementation of the news recommender system. Chapter 4 describes the experiments we run on our modules and discusses the results and analysis of the experiments conducted. Finally, Chapter 5 summarizes the thesis and also talks about the future work.

2. LITERATURE REVIEW

2.1 Recommender Systems

In this section, we discuss previous research on recommender systems. Recommender systems are widely used to help readers filter through an ever-growing flood of information. Basically, recommender systems implement an information filtering method to select items from a stream of information that the users that are likely to find interesting. Recommender systems collect data from the users explicitly or implicitly and, based on the collected information, create user profiles. The user profiles are then used to generate recommendations. With explicit information collection, the user typically rates items in addition to his regular tasks. For example, in addition to purchasing an item, the user is asked to rate it with one or more stars. However, with implicit information collection, the recommender system monitors the user's behavior with items during their normal activities. No extra user effort is required. However, the system must infer the user's preferences from their actions.

Recommender systems have been considered as a remedy to overcome the information explosion problem and a lot of research effort has been focused on developing highly reliable recommendation techniques. Traditional recommender systems are classified based on what information they use and on how they use that information [11]. Recommender systems are usually classified into three categories, based on how the recommendations are made [12].

- Content-based recommender systems: These recommender systems recommend an item to the user similar to the ones the user preferred in the past. In other words, the system

basically compares the contents of candidate items to the contents of the items the user has shown interest in the past.

- Collaborative recommender systems: These systems recommend an item to the user based on the people with similar tastes and preferences have liked in the past. These systems basically determine similarity based on collective user-item interactions rather than on any specific items contents. These systems are widely used and are generally used to recommend books, movies, music etc. They have the advantage that they can recommend items for which little or no semantic information is available (music, movies, products).
- Hybrid recommender systems: These systems combine both the collaborative and content-based recommendation techniques in order to improve the accuracy of the recommendation.

The information gathered from either content-based or collaborative filtering approaches can be used for either memory-based or model-based algorithms. Memory-based systems calculate recommendations on-the-go based on the previous user behavior. On the other hand, model-based systems are developed using data mining and machine learning algorithms to find patterns based on training data [13]. These systems incorporate algorithms such as Bayesian networks, clustering models, and semantic models to make predictions for real data from the training model to make recommendations. Memory-based systems are easy to implement, work well in real-time, and new data can be added easily and incrementally. However, this technique can become computationally expensive and the performance is affected when data is either

sparse or dense. Also these systems are dependent on human ratings and have limited scalability for large datasets.

Model-based systems better address the sparsity, scalability, and other problems faced by memory-based systems. These systems not only improve the prediction performance but also give and intuitive rationale for recommendations. Model-based systems have a more holistic goal to uncover latent factors that explain observed ratings [14]. However, the downsides of the model-based systems are expensive model-building and loss of useful information caused by dimensionality reduction techniques. Some applications implement a hybrid model that fuses both these models to overcome the limitations such as sparsity and loss of information. The goal of these hybrid systems is to improve the prediction performance and to overcome the limitations faced by the model-based and memory-based approaches. However, these systems have increased complexity and are expensive to implement [15].

Our recommender system, implements both content-based and collaborative filtering methods with both memory-based and model-based attributes. Sections 2.1.1 and 2.1.2 discusses the content-based and collaborative recommender systems respectively and section 2.2 discusses the different types of news recommender systems.

2.1.1 Content-Based Recommender Systems

In this section, we explore some of the applications that use content-based recommender systems. Content-based recommender systems are based on information about and attributes of the items that are going to be recommended. In other words, these systems recommend items that are similar to those items in which the user has shown interest in the past. The items are recommended based on the comparison between the item contents and user interests. These recommender systems are used in various domains such as web sites, web blogs, restaurants, news articles etc. The user profile is built based on his interests and this profile indicates the type of items that the user likes. Several techniques have been implemented to identify the items matching the user profile to make recommendations.

Most traditional content-based recommender systems depend on the content of the items themselves. The keywords associated with the items are used to match against the user profile to make recommendations. Hence, this approach only works for those applications whose items have contents in the form of text. Recommendations are done based on the comparison between the contents of the items and the keywords associated with the user profile that indicate his interests.

Content-based recommenders primarily use a weighting mechanism to rank the items by assigning weights to the keywords and to differentiate between the items. The keywords are extracted from the contents of the items that the user has shown interest in the past. These keywords form the basis for the user profile. If a user likes an item, the weights of the terms extracted from the item's content are updated to the weights of the corresponding terms in the

user profile. The items recommended in the future are primarily based on the user profile. There are several methods to calculate the weights of the keywords in the content. The most commonly used method is the TF-IDF (Term Frequency – Inverse Document Frequency) method.

Term frequency (tf) is defined as frequency that a word appears in a document divided by the number of words in the same document.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Figure 1: Term Frequency (tf)

Where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j.

The inverse document frequency (idf) is obtained by taking the logarithm of the value obtained by dividing the number of documents in the collection with the number of documents in which a term (t_i) was found.

$$\mathrm{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

Figure 2: Inverse Document Frequency (idf)

where,

|D| : Total number of documents in the collection.

$|\{d : t_i \in d\}|$: Number of documents where the term t_i appears.

Then, the weight of a term *"i"* in document *"j"* is calculated as follows:

$$wt_{i,j} = tf_{i,j} * idf_j$$

Examples of the earliest traditional content-based recommender systems include [17], [18] and [19]. In [17], Ken Lang implemented a Netnews filtering system, "Newsweeder", that addresses the user profile building problem by letting the user rates his or her interest level for each article being read. The goal of an information filtering system is to sort through large volumes of information and present to the user those documents that are likely to satisfy his or her information requirement. Ken Lang used two approaches TF-IDF and MDL (Minimum Description Length) for information filtering. Two metrics were used to evaluate the performance. One metric was precision that calculated the ratio of relevant documents retrieved to all documents retrieved. The other was the confusion matrix of error generated by text in which the column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. He found that MDL approach outperformed the traditional TF-IDF approach.

[18] and [19] conducted similar work by developing intelligent information agents InformationFinder, that recommended information such as news, bulletin boards, databases etc, and Syskill & Webert, that recommended movies, respectively. These intelligent agents

incorporate traditional content-based filtering approaches to make recommendations. These agents recommend items that match the contents of the user profiles. The user profile is built based on the user's interest and preferences in the past through explicit rankings, and this profile forms the basis for these agents to find items that are interesting to the user.

Mooney and Roy [16] developed a next-generation content-based recommender system that utilizes information extraction and a machine-learning algorithm for text categorization. Their prototype system, LIBRA (Learning Intelligent Book Recommending Agent), uses a database of book information extracted from web pages at Amazon.com. They performed a subject search on the Amazon website to obtain a list of book-description URLs that are published on subjects of broad interests. LIBRA then downloads each of these pages and uses a simple pattern-based information-extraction system to extract all the information pertaining to the book such as author, title, publications, related authors etc. Pre-processing is performed on the information gathered for the removal of stop-words and formatting to obtain unique tokens. The content associated with the synopses, published reviews and customer comments were clustered into a single attribute called description.

The system was trained by querying on specific authors and titles to retrieve relevant books. The books retrieved were presented to the users who were asked to rate their interest in the book. These user-book ratings form the explicit input on which to build the user profile. The system then learns a profile of the user using a Bayesian learning algorithm and produces a ranked list of the most recommended additional titles from the system's catalog. Next, the classifier was used to predict the user rankings for the remaining books. Finally, the top-scoring

books were recommended to the user. Also, the system adapts to the changing interests of the user and produces new recommendations based on the information collected from the user.

LIBRA was evaluated on several data sets. Experiments were conducted on the book recommendations and the books were rated by two different users. The performance of the system was analyzed using a 10-fold validation experiment for varying number of training documents. The results indicated that the top recommendations by LIBRA were found interesting to the users when compared to randomly selected documents. The results also implied that performance was still high despite considering very few training examples.

In more recent work, Chang et al [20] proposed a content-based recommender system to recommend tags that can be applied to web pages with or without prior tag information. Each web page was represented using two vectors. One is a tag vector that represents the prior tag information of the web page and the other is a term vector that represents the keywords in a webpage. The cosine similarity is calculated between the tag vector and term vector for all the web pages. The cosine similarity is also calculated between the tag vector for each web page and the term vectors of all the web pages. Finally, the similarity between all pairs of web pages is figured as a weighted sum four cosine similarity values calculated for each possible pair of vectors (tag-tag, tag-term, term-tag, and term-term). For each web page, the tags associated to the most similar web pages are considered for recommendation to the user.

They used a tag/term significance measure that is analogous to the TF-IDF measure for the purpose of evaluation. With the help of this measure, the terms that appear in fewer

documents are assigned with a higher weight in order to differentiate between the web pages. The propagation weight for each tag is calculated as it depends on its weight in the original webpage and the similarity between the sending and the receiving web pages. For the purpose of their study, they crawled over a million URLS and all the tags from Bookmarks on the delicious.com home page. The results of an experiment conducted for 31 web pages indicated that 69.45% of the tags recommended by their algorithm were relevant to the users.

In [21], Clark et al present Engene, a web-based recommender system similar to Chang's work that incorporates a genetic algorithm-based classifier with TF-IDF weights to calculate the term weights in order recommend web pages to users. In this system, user profiles were built on implicit information.

2.1.2 Collaborative Recommender Systems

Collaborative recommender systems are the best known and most widely used recommenders. Examples of organizations that use collaborative recommender systems are Amazon, YouTube, and Reddit. A profile is created for each user (item) according to the similarity of other users (item) in the system. According to the profiles, collaborative filtering recommender systems recommend items to target users according to the preferences of their similar users. There are two major types of algorithms for collaborative filtering: user-based and the item-based. User-based algorithms find out the most similar neighbor users to a target user based on the similarity of ratings. The products having the highest ratings from the neighbors are recommended to the target user [24]. Essentially, if the target user and the neighbor both like

some items in common, then the target user is likely to like other items that their neighbor has liked that the target user has not yet purchased and/or rated.

For item-based algorithms, when user is interested in an item, similar items are also recommended to the user [22, 23]. Item similarity is based on items that are commonly purchased/liked together. If , in the past, people who like Star Wars also like Lord of the Rings, then a new user who has watched Star Wars should have Lord of the Rings recommended to them.

Traditional collaborative recommender systems incorporate similar steps in order to make recommendations to the user. First, the user-item ratings information is collected. Each user is provided with a collection of items for him to rate according to his interests. Each user is represented by item-rating pairs, which contains the ratings provided by the user to several items. Next, vectors are created that represent users or items and a similarity measure is chosen. There are several possible measures to calculate the similarity between two vectors. Pearson correlation, cosine vector similarity, mean-squared difference and Spearman correlation are some of the popularly used metrics to measure the similarity between two vectors. Then, the next task is to identify the neighbors who will serve as recommenders. There are two techniques to identify the neighbors selected to make recommendations. With threshold-based selection, vectors whose similarity exceeds a certain threshold value are considered as neighbors of the target user. In contrast, with the top-n technique, the n-closest neighbors are selected for a given value of n. Finally, predictions are produced by calculating the weighted average of the neighbors' ratings, weighted by their similarity to the target user.

Similar to content-based algorithms, collaborative filtering algorithms can be classified into two main categories, memory-based collaborative filtering algorithms and model-based collaborative filtering algorithms. Both these filtering algorithms have the same advantages and disadvantages as discussed in the earlier section.

Examples for the earliest traditional collaborative recommender systems include [25] and [26]. [25] developed GroupLens that recommended news articles to users based on user similarity. [26] developed the Bellcore Video recommender systems that recommended videos to users based on user similarity. Although successful first approaches, these traditional collaborative recommender systems often face challenges such as sparsity, scalability, and cold-start. Sparse data can be a problem because if you have a few thousand users but several million items, there may not be any users who have purchased the same items as a target user. Scalability is an issue because there may be millions of items and/or users and millions of transactions a day. Running machine learning algorithms may be intractable, leading to a need to reduce features and/or entities in the algorithm. Finally, the cold start problem is the most difficult. Initially, the system may have no rating data at all with which to begin. If it cannot make recommendations without rating data, and it cannot get rating data without a lot of involved users. How can you convince users to rate information if those ratings do not result in good recommendations?

To overcome the above mentioned problems, SongJie Gong [27] developed a collaborative recommender system based on user clustering and item clustering. Users are clustered based on their ratings on several items and each user's cluster is associated with a cluster center. Based on the similarity between a target user and the cluster centroids, the nearest neighbors of target user can be found to make recommendations and predictions where necessary. By clustering users and items, data can be aggregated across users or items, alleviating the sparse data problem and leading to greater accuracy. By comparing users to clusters rather than all other users, this approach is more scalable than the traditional collaborative recommendation.

The approach proposed by Gong is explained in detailed below. First, users are clustered into groups using the K-means algorithm. The data sparsity problem faced by other collaborative recommender systems is overcome by the explicit use of the item clusters as prediction mechanisms. Based on the results from item clustering technique, predictions strategies are applied to the vacant data. Next, item clustering is implemented to identify the items with similar user ratings. The items are clustered in a manner similar to the user clustering technique. Next, the cluster centers and the neighbors are identified. From all these information gathered, the weighted average of the neighbors' ratings is calculated, weighted by their similarity to the target item to produce recommendation to the users.

Gong formed a dataset from enough users to obtain 100,000 ratings from1000 users on 1680 movies with every user having atleast 20 ratings. The ratings are on a numeric five-point scale with 1 and 2 representing negative ratings, 4 and 5 representing positive ratings, and 3

indicating ambivalence. Several metrics such as the mean absolute error, root mean square error and correlations between ratings and predictions are used for the purpose of evaluation and to deduce conclusions. The results clearly indicate that the proposed algorithm outperforms traditional collaborative recommendation algorithms.

In [28], Yang et al proposed a ranking-oriented approach to collaborative filtering technique similar to Gong's work that addresses the item ranking problem directly by modeling user interests inferred from the ratings. The similarity between the users is measured based on the correlation between their rankings of the items rather than rating values. They present new collaborative filtering techniques for ranking items based on the interests of similar users. They presented a ranking-oriented technique to collaborative filtering that directly deals with the item ranking problem without considering the predictions for ratings.

2.2 News Recommender Systems

News recommender systems are widely used and are a promising research direction. With so many information sources, the Internet provides fast access to the millions of news articles around the globe. However, users need recommendations to help them find the most interesting articles from this flood of information.

News recommender systems can be broadly classified into two types based on the type of recommendations made to the user. Some recommender systems take advantage of online social

networking sites to provide interesting news articles to the user. Such recommendations are called popularity-based news recommendations since the articles are ranked based on their popularity identified from the social networking websites. Other recommender systems recommend interesting news articles to the user solely based on user interests. Such recommendations are called profile based news recommendations since they rank the news articles based on the user's interests. The following two sections explore the applications based on the popularity based recommendation and profile based recommendation techniques.

2.2.1 Popularity Based News Recommender Systems

News recommender systems are widely used to help readers filter through an ever-growing flood of information. Many researchers focus on using real-time social networking sites such as Facebook, Google Plus, and Twitter to identify the most popular and most current news stories. Many news recommender systems make use of activity on micro-blogging services, such as Twitter, to identify news items that generate a lot of interest. Often, news stories break on informal micro-blogs before they appear on the traditional news service websites. Because they are instant, and widely available, they provide a massive source of information on current events. However, because they are unmoderated, the quality of the information is variable. Alan et al. discuss a method to determine which Twitter users are posting reliable information and which posts are interesting [2]. They identified future events based on the tweets from reliable Twitter users with the help of tense identification of their tweets.

Micro-blog posts can also be used as a way of identifying the popularity of certain events. Smyth et al. represent users and items based on micro-blogging review of movies and used this technique with various movie recommendation strategies on live-user data [1]. Their strategies combine the frequency of the tweets about a given movie with analysis of the tweets to identify positive and negative reactions.

Phelan et al. focus on using micro-blogging activity to recommend news stories [7]. Their recommender system, Buzzer, is applied to RSS feeds to which the users have subscribed. Buzzer mines the content terms from RSS and Twitter feeds and uses them to rank articles. Buzzer recommends news stories based on three different retrieval strategies namely, *Public-Rank*, for mining tweets from Twitter's public timeline, *Friend's-Rank*, for mining tweets from people the user follows and, *Content-Rank*, for ranking articles based on term frequency alone and scoring the articles based on the frequency of occurrence of the top RSS terms. It matches the mined terms with the index of RSS items. The overall score of each article is calculated by accumulating the TF-IDF (term frequency - inverse document frequency) scores across all the terms related within each article. They used two new recommendation strategies where the tweets are mined from Twitter's public timeline and from people the user follows and these mined terms are matched with the RSS terms that are gathered from all users' subscriptions in the Buzzer community. Also each user is allowed to sign up to a daily digest of email stories to store each user's response. In [8, 9], they extended their work by considering two additional strategies. They considered the public-rank and the friend's-rank strategy over all the news articles in the Buzzer system rather than just considering the articles from the users' index.

A user trial was conducted on 35 active Buzzer users for over a month. During this period, they collected a total of 56 million tweets from the public timeline and 537,307 tweets from the social graphs of the 35 registered Buzzer users. They observed the user click behavior for all the strategies. The results indicated that the public-rank and friend's-rank strategy outperformed the other strategies and is considered better when compared to the traditional keyword (TF-IDF) based strategy.

2.2.2 Profile-Based News Recommender Systems

Profile based, or personalized, news recommender systems recommend articles to the user based solely on his/her interests. A user profile is built based on the preferences or interests of the user. In one of the earliest news recommendation systems, Pazzani et al. used, *News Dude*, a personal news recommending agent that uses TF-IDF in combination with Nearest Neighbor algorithm in order to recommend news stories to users [4]. They built their user profile implicitly and divided the user interests into short-term interests and long-term interests. They developed a hybrid user model that considers both long-term and short-term interests and found out that this model outperformed the models that consider either of these interests.

Similarly, Michael et al [3] describe a content-based recommendation system that recommends a story to a user based upon a description of the item and a profile of user's interests. More recently, Cantador et al. present a system, News@Hand, that has similar goals but which incorporates semantic web technologies [6] News@Hand uses semantic annotation

based on ontologies to group news items and then recommends news stories to users that match their semantically-based profile.

Wouter et al. described ontology based methods to recommend news articles to the users depending on their interests [5]. The user profile is based on the news articles browsed recently and the ontology is provided by, *Athena*, a framework built for news personalization service which is an extension of their framework, *Hermes*, mentioned in their earlier work [29]. The ontology is developed to store the concepts and their relations to the news items. Athena uses the traditional keyword-based recommendation technique along with the semantic-based recommendation algorithms to compare the unread news articles with the user profile. The news items that most match with the user profile are recommended to the user.

The semantic-based recommendation algorithms consider the concepts and the semantics of the text to determine the relation between various keywords. Various techniques are used to identify the semantic relatedness between the keywords in the articles and explained in detail below. The first technique is a simple mechanism called concept equivalence. The user profile is considered as a set of concepts based on the interests of the user. The news article is also considered as a set of concepts pertaining to the domain of the article. For a new article, the interestingness of the articles is calculated by the intersection of the above two sets. The other techniques binary cosine and the Jaccard similarity coefficient compute the cardinality of intersection of above two sets relative to the cross product and union of the above two sets respectively. The semantic relatedness of two keywords is calculated by creating a vector that represents the keywords and calculating the cosine similarity between the two vectors. The

advantage of this approach over the other three techniques is that it takes the related concepts of a concept into account that occurs in a text.

For the evaluation of this approach, Wouter et al. considered 5 users with different news interests and each user is provided with 300 different news articles and is asked to rate all the articles by skimming through the summaries. The articles rated by the user are divided randomly into two sets, training set and validation set. The training set is used to create the user profile. The validation set is used to determine the similarity of the news article to the user profile. To determine the performance of this system, measures like accuracy, precision, recall and specificity are used. The results indicate that the semantic-based news recommender performs better than the traditional recommender systems based on TF-IDF.

Our news recommender system incorporates both the strategies explained above, popularity and profiles, to present a novel hybrid approach to recommend interesting news articles to the user.

3. APPROACH

In this section, we present an overview of our hybrid news recommendation system. Section 3.1 shows a schematic view of the system architecture and provides an overview of the overall system. Sections 3.2 - 3.4 describe each component in more detail. Our basic approach is to recommend interesting news articles to the user based on a combination of his past interests and stories that are currently of broad interest. The user's past interests are captured in his user profile and the community as a whole's broad interest is captured from tweets collected from Twitter's public timeline.

The first step in the process is to collect a set of news stories to serve as the source of the recommendations. We collect RSS articles from online news sources and then index them using an open source tool called SOLR. We also collect tweets from Twitter and, using the tweets as queries against the index, to identify the stories most closely related to the tweets. By accumulating the number of tweets related to a given news story, we can identify the "hottest" or most popular news stories.

In this version of the system, the RSS news stories are collected from a news site that has the stories associated with high level categories, e.g., Sports, Business, Politics, etc. Thus, for every story in our collection, we know at least one topic with which is it associated. We automatically categorize the stories to identify further topics that are related to each story. The users build their profiles by explicitly identifying the topics in which they are most interested. By matching the profile topics to news story topics, we can identify the news stories that best match the user's profile.

Finally, our intuition is that user's most want to see news stories related to topics in their profile that are also creating a buzz on the blogosphere. Thus, our hybrid system recommends news stories that are a combination of the previous two systems. In other words, users are shown hot stories related to their favorite topics.

3.1 High Level Design

Figure 1 shows an architectural diagram of our Hybrid News Recommender system.

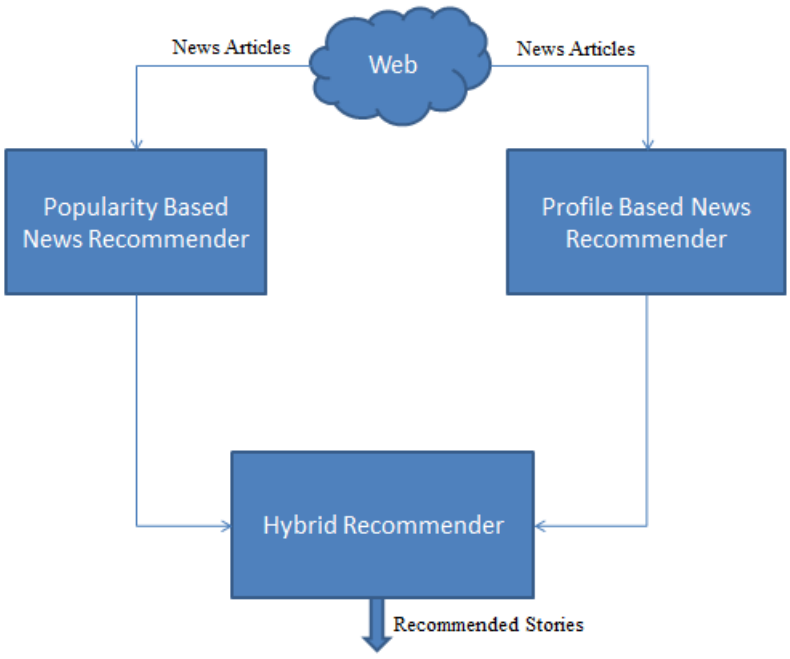


Figure 3: Overall Architecture of the Hybrid Recommender System

The system consists of three modules:

1. Popularity-Based News Recommender
2. Profile-Based News Recommender
3. Hybrid News Recommender

Each module implements a different approach to selecting the news articles to recommend to the user. The first module selects news articles based on the popularity of the article. The popularity of the article is identified with the help of a micro-blogging service such as Twitter. Tweets are collected from the Twitter's public timeline and analyzed to identify the stories about which users from around the world are tweeting. These tweets from the public timeline are compared to news articles based on the co-occurring terms in the tweets and the articles. Articles that are mentioned frequently in the tweets are considered "hot" or popular. This recommender module ranks the articles based on their overall popularity.

The second module ranks the news articles based their similarity to a user's profile. In this work, the user builds their profile by explicitly providing input about their level of interest in each of 7 different news categories. Incoming news articles are automatically classified into the same set of categories using a k nearest neighbors text classifier [31]. The news articles are then ranked based on the similarity between the categories in the user's profile and the categories to which the article belongs.

The third module fuses the results from the above two modules to recommend news articles to the user. The articles are ranked based on a combination of their popularity ranking and the similarity to the user's profile. In contrast to the first module that recommends "hot"

articles that are of interest to the world at large and the second module that recommends articles related to the user's profile regardless of their popularity, this module recommends “hot” articles to that are relevant to topics interesting to the user.

3.2 Popularity-Based News Recommendations

Figure 2 shows an architectural diagram of the Popularity Based News Recommender system.

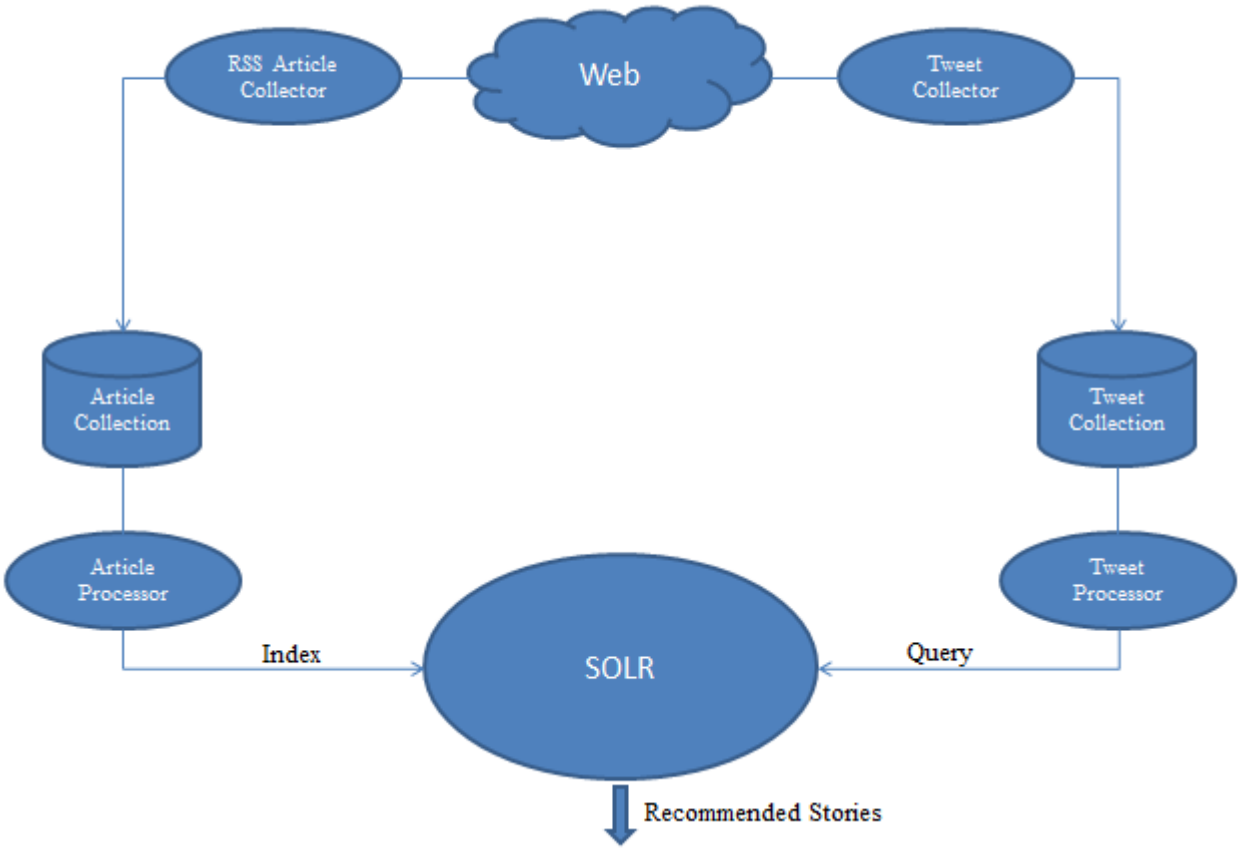


Figure 4: Architecture of the Popularity-Based Recommender System

First, the RSS articles are collected from a news source such as CNN, BBC, or Wiki etc. and stored in the file system. These websites organize their stories by category, e.g., Sports, Business, Politics, Entertainment etc. The file system stores all the articles organized into folders based on the category to which they belong. The article collection contains 280 different articles collected from 7 different categories (Sports, Crime, Business, Politics, Tech, Health and Entertainment). This data set is depicted by the Article Collection object in Figure 1. The RSS articles are first pre-processed before they are indexed. The Article Processor is responsible for processing the articles by removing unnecessary content (html tags, numbers, etc.) and preserving the textual content.

The pre-processed articles are then indexed by SOLR, an open source enterprise search platform from the Apache Lucene project [30]. The major features provided by SOLR include powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, and geospatial search. SOLR uses the Lucene Java search library at its core for indexing and search, and it has several APIs that make it flexible to use from any programming language. SOLR is comprised of three main components, an indexer, a Lucene index and a server. The indexer is responsible for collecting all the pre-processed articles from the Article Collection and creating a Lucene index over them. The Lucene index is a file-based inverted file that supports fast lookup by document content. The SOLR Apache server works on top of this Lucene index developed by the indexer. Any requests must be made to the server which outputs results based on the underlying index. All the queries are submitted to the

server which outputs the documents based on the index. Figure 3 diagrams the SOLR architecture.

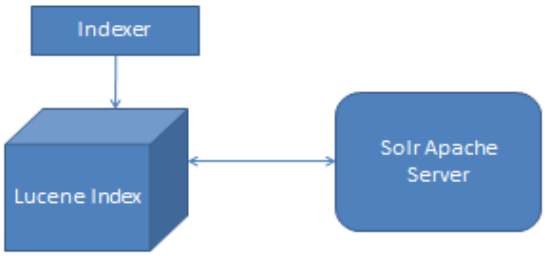


Figure 5: SOLR Architecture

In order to identify which news stories are most popular, we also collect data from the Twitter micro-blogging site. These tweets are collected from the Twitter's public timeline by the Tweet Collector that forms a request to Twitter's streaming API. The collected tweets are stored in the Tweet Collection in JSON format. The Tweet Processor is responsible for parsing the Tweet Collection by eliminating the unwanted noise and preserving the tweet content. These processed tweets from the collection are passed as queries to the Apache SOLR server. Each parsed tweet is queried against the server to retrieve the articles corresponding to the tweet content. The server retrieves the articles associated with the tweet content along with a weight that indicates the similarity between the query (tweet) and the article. These articles are sorted

based on their weights accumulated across all the tweets. Thus, the news articles are ranked based on their popularity.

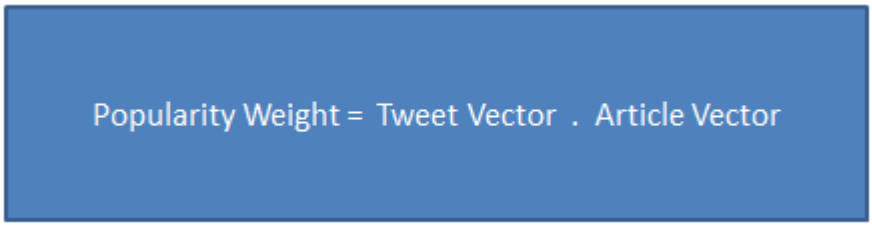


Figure 6: Popularity Weight

Consider the following example of 5 news articles and 3 tweets. The methodology on how the articles are ranked to present user with the popular articles is explained in detail below. For the purpose of understanding this example consider the all the tweets and the news articles pertaining to a single category, sports. The news articles are depicted by numbers ranging from 100 to 104 and the tweets are considered as t1, t2 and t3. The news articles are indexed with the help of Solr lucene indexer and tweets are queried against this index. The following table shows the articles and the tweets queried along with the weights of the retrieved articles.

Table 1: Article vs Tweets

Article/Tweet	T1	T2	T3	Total Wt
100	0.7		0.9	1.6
101		0.4		0.4
102	0.6	0.3	0.5	1.4
103	0.5	0.2		0.7
104			0.1	0.1

The blank fields in the above table indicate that the particular document is not retrived for the corresponding query. The total weight indicated in the fourth column is the aggregate weight for all the tweets and is known as the popularity weight. The following table shows the sorted order (left to right) of the documents that is displayed to the user.

Table 2: Recommended Stories by the Popularity-Based Recommender

Popular Articles	100	102	103	101	104

3.3 Profile-Based News Recommendations

In this section, we describe the profile-based news recommender system. This system is comprised of four components and each component is explained in detail in the following paragraphs. Figure 3 shows the architectural diagram of the Profile-Based News Recommender system.

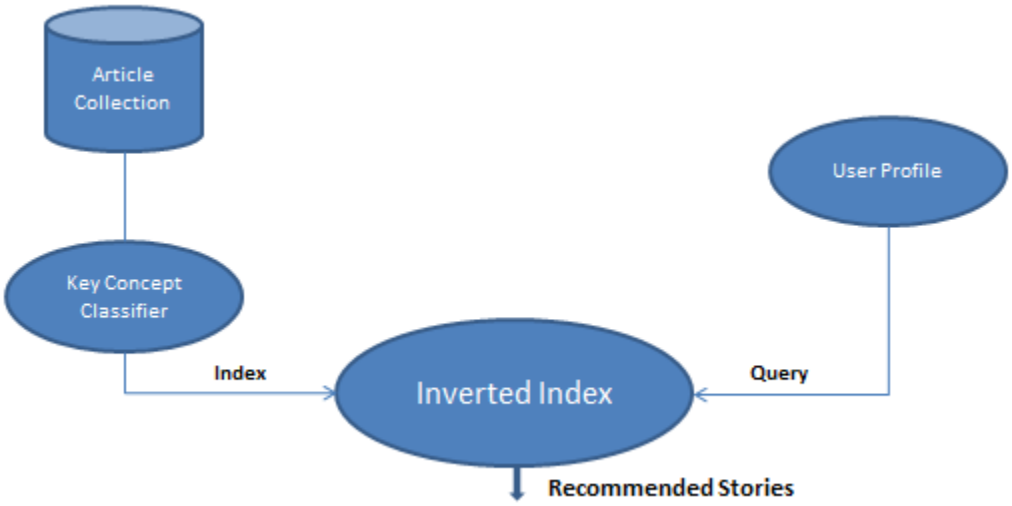


Figure 7: Architecture of the Profile-Based Recommender System

The profile-based recommender system uses the same article collection as the popularity-based recommender system. Although articles are placed in only one category by the website editor, they may actually partially belong to more than one category. To allow for this, each article is classified into all 7 potential categories using a k-nearest neighbor classifier, the classification module of the KeyConcept project. This tool classifies the articles to categories based on the article's similarity to training documents for each category. The k-nearest neighbors

classification approach identifies the k most similar training documents to the article then uses those similarity scores as votes for the category to which the training documents belong. The similarity of the article to each category is thus the sum of the scores of the article's similarity to the training documents in that category that fall in the top k most similar documents overall. The categories are then sorted by their accumulated scores. In this module, we store the top 3 most similar categories (and their similarity scores) for use in profile matching.

In order to do fast lookup by category, we build an inverted index that allows us to quickly located documents that belong to a particular category. Just as SOLR can be used to build a Lucene index that maps from keywords to document ids and weights, SOLR can again be used to build a second Lucene index that maps from category ids to document ids and weights. This inverted index stores the news articles based on the category to which they belong rather than the keywords in the article.

Next, for each user creates a profile based on his interests in various categories. Each user is provided with an interface to create his own profile by manually scoring the categories. The following figure shows the interface that prompts the user to develop his profile.

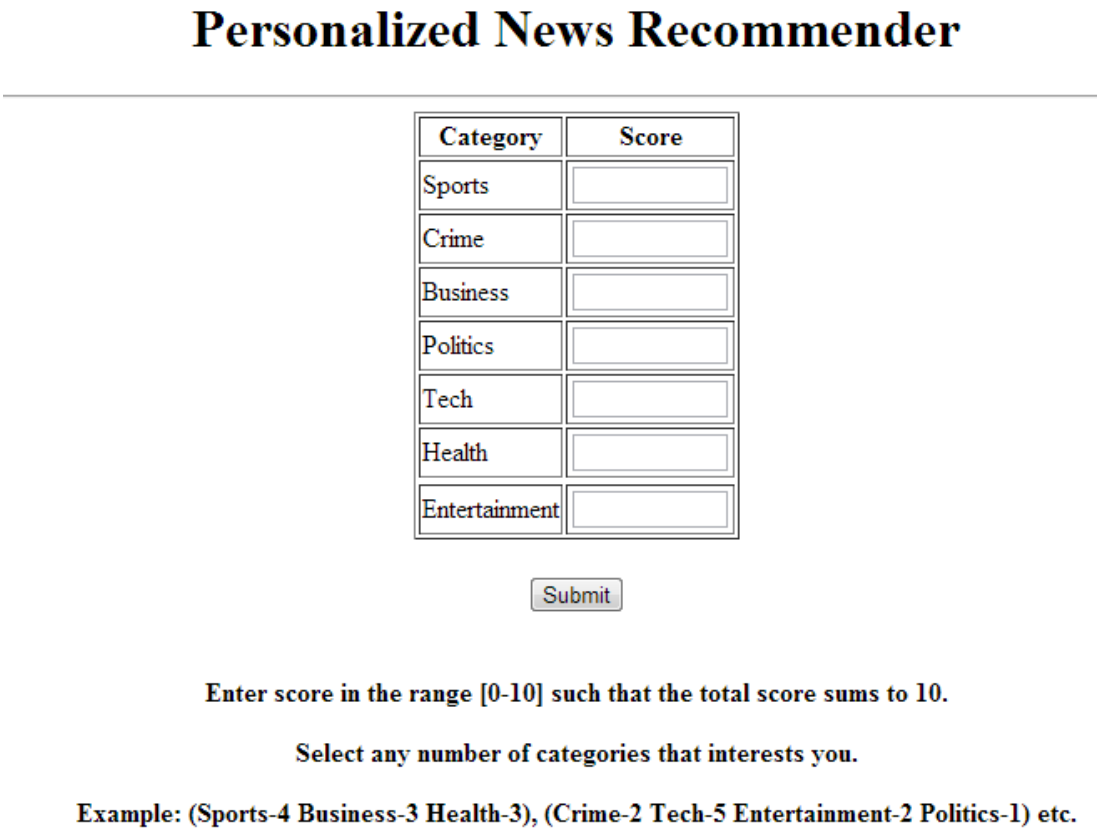


Figure 8: User Profile Interface

Finally, the user profile is used to identify documents that best match their profile. The profiles and the articles can be viewed as feature vectors where each category is a feature. The similarity of each article to the user's profile is calculated using the cosine similarity measure between the user's profile and each article. This simplifies down to doing a dot product between the two feature vectors.

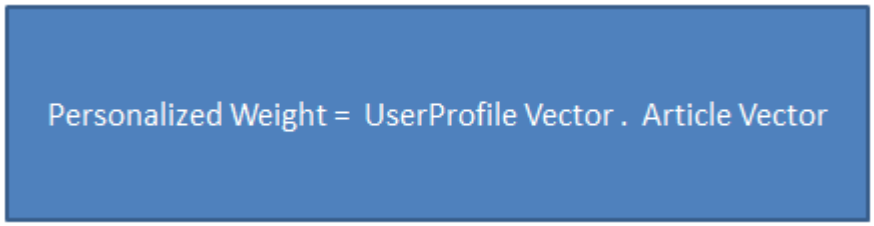


Figure 9: Personalized Weight

To implement this, we simply query the category-based Lucene index for each non-zero category in the user's profile with the category id and category weight. Lucene accumulates the scores for each article (multiplied by the category weight) and returns a list of the best-matching articles, sorted by weight.

Consider an example similar to the one considered in the previous section with five news articles. These news articles are classified with the help of KeyConcept and the following table depicts the articles (labeled 11-15) with their associated category and weight. All the articles are classified into three categories and each article is associated with a different category weight based on its relevance to its corresponding cluster. The following table depicts the news articles along with their score given by the KeyConcept classifier.

Table 3: Articles and their Category Weights

Cateory/Article	11	12	13	14	15
Sports	0.7		0.5		
Crime		0.4		0.4	
Business	0.4		0.2		0.7
Politics		0.5		0.3	0.3
Tech			0.6		
Health	0.5				0.1
Entertainment		0.2		0.1	

Consider a user who is solely interested in the sports category and hence he rates the sports category to ten. All the articles in the above table corresponding to sports category are multiplied with the users rating. As the user is only interested in sports all the articles associated with sports are considered and their category weights are multiplied by the user's score for that category. This weight is known as the personalized weight. All the articles in the above table are updated and sorted with the user's score on the category and displayed in the table below.

Table 4: Articles and their Personalized Weights

Articles	11	12	13	14	15
Personalized Wt	7		5		

The articles displayed in the following table are presented to the user. These articles are personalized to the user based on his interests.

Table 5: Recommended stories by the Profile-Based Recommender

Personalized Articles	11	13	12	14	15

Thus, this module recommends news articles based on how well they match the user's profile. This module does not take any popularity into account. It merely ranks articles based on how well the categories and weights associated with an article match the categories and weights provided by the user's profile.

3.4 Hybrid News Recommendations

This section explains hybrid model that recommends articles to the user based on a combination of by the article's popularity and its match to the user's profile. Essentially, this module combines the scores provided by each of the previous two modules to produce a recommendation that combines the article's match to the user's interests with the articles popularity to users everywhere.

This module calculates a Hotness Weight by multiplying the Popularity Weight by the Personalized Weight.

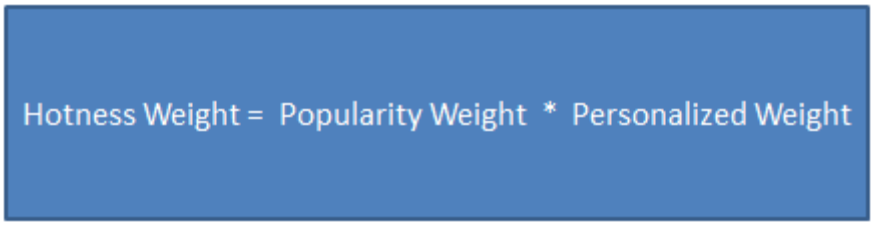


Figure 10: Hotness Weight

Consider the two examples described in the above two sections for calculating the popularity weight and personalized weight. The following table depicts all the 5 news articles with their corresponding normalized popularity weight and personalized weight.

Table 6: Popularity and Personalized weights of the Articles

Articles	Popularity Wt	Personalized wt
100	1	0.42
101	0.25	0.71
102	0.875	0.28
103	0.4375	1
104	0.0625	0.57

The Hotness Weight for an article as depicted in the above formula is calculated by multiplying the articles popularity weight and the personalized weight. The following table shows the articles with their corresponding hotness weight.

Table 7: Articles and their Hotness Weights

Articles	100	101	102	103	104
Hotness Wt	0.42	0.177	0.245	0.4375	0.035

The following table shows the order in which the hot articles are presented to the user.

Table 8: Stories Recommended by the Hybrid Recommender

Hot Articles	103	100	102	101	104

The following table shows the order in which the articles are presented to the user for all the three approaches discussed above.

Table 9: Stories Recommended by three Strategies

Popularity-Based Reco.	Profile-Based Reco.	Hybrid Reco.
100	103	103
102	101	100
103	104	102
101	100	101
104	102	104

Thus, this module recommends popular articles that are in the user's areas of interest.

4. EVALUATION

This section describes the experiments that we have conducted on the modules described in Chapter 3 to compare the accuracy of the various recommender systems. In particular, we evaluated the accuracy of the recommendations based on popularity alone, the recommendations based on matching the user profile alone, and the hybrid recommendations that take into account both factors. All experiments were conducted on the same collection of news articles using the same set of 27 volunteer test subjects.

4.1 Experiment One: Evaluating the Accuracy of Matching Tweets to News Articles

In our popularity-based recommender, the popularity of the news articles is determined with the help of the tweets from the Twitter's public timeline. A key component of that recommender is the ability to associate tweets from Twitter with news articles, thus being able to identify those articles being talked about the most on the blogosphere. The goal of this experiment is to determine whether or not we can accurately match a tweet with a related news article so that we can build our popularity-based recommender around that tweet/news article matching component.

The datasets for this experiment are a collection of over 100,000 tweets from the Twitter's public timeline and a set of 288 RSS news articles from the websites of CNN and the New York Times collected on the same day.

The tweets were obtained by requesting the Twitter's Streaming API that returns the most recent tweets from the public timeline. The Streaming API returns the tweets in JSON format that is processed to obtain the actual tweet content. Figure 4.1 shows the JSON format.


```
{
    "in_reply_to_status_id_str":null,
    "text":"RT @daVirgoJB: #thestruggle college life",
    "favorited":false,
    "in_reply_to_user_id_str":null,
    "geo":null,
    "in_reply_to_status_id":null,
    "source":"web",
    "in_reply_to_screen_name":null,
    "truncated":false,
    "entities":
    {   "user_mentions":
        [
            {
                "indices":[3,13],
                "name":"Ms. McLovin",
                "screen_name":"daVirgoJB",
                "id":113486861,
                "id_str":"113486861"
            }
        ],
        "hashtags":
        [
            {
                "text":"thestruggle",
                "indices":[15,27]
            }
        ],
        "urls":[]
    },
    "id":190148436308398080,
    "id_str":"190148436308398080",
    "contributors":null}
```


Figure 11: JSON format for a tweet

This experiment is conducted over 102,224 tweets from the public timeline and 280 RSS news articles from the CNN website (http://www.cnn.com/services/rss/) and New York Times

website (http://www.nytimes.com/services/xml/rss/index.html). These news sources categorize their news articles into 18 top level categories. We selected seven categories common on the two sites as the source of articles for this study. A collection 280 articles, 40 per category, were downloaded to form the news article dataset. Both the tweet and new article datasets were obtained on the same day, 18th March 2013.

First, a Lucene index was created to provide searching over the RSS news articles dataset with the help of an open source enterprise search platform from the Apache Lucene project. The following figure describes the statistics of the SOLR search platform indicating the number of documents that are indexed.

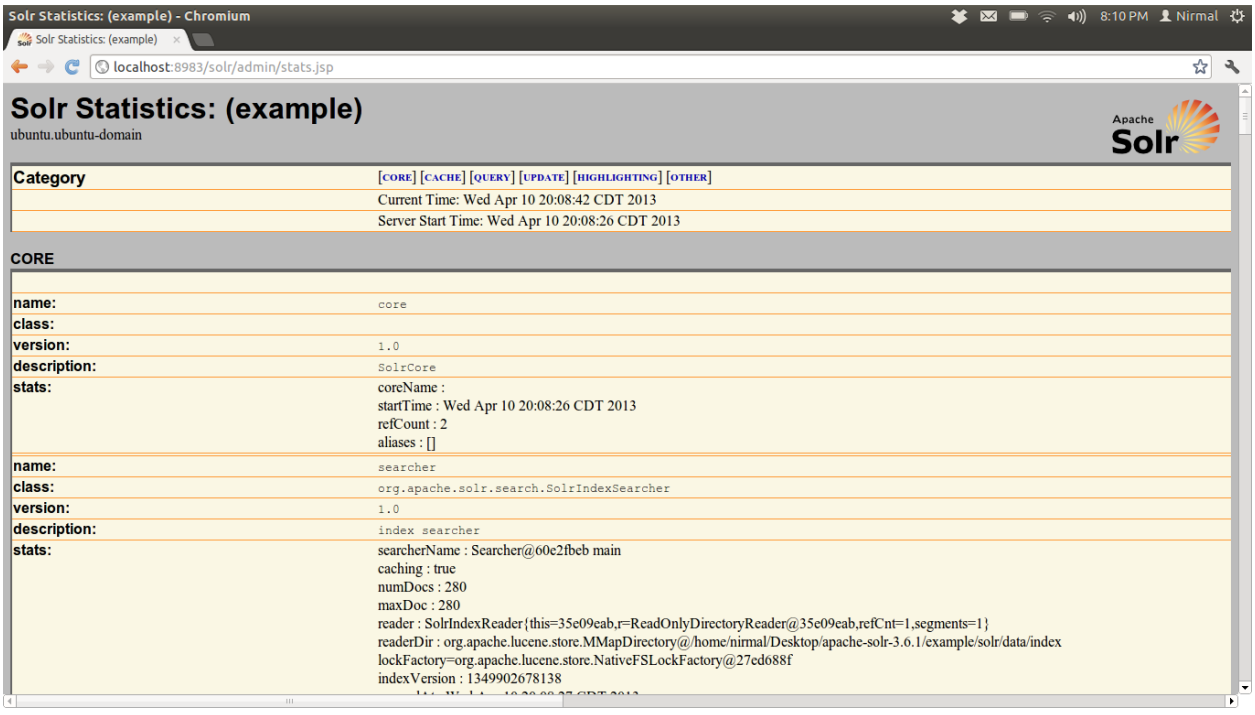


Figure 12: SOLR Statistics

Next, each tweet from the tweets dataset is sent as a query to the SOLR server to retrieve the documents and their weights associated with the corresponding tweet content. Our popularity-based recommender will accumulate the match weights between tweets and articles across a large collection of tweets to identify the most-discussed articles. Key to this is the accuracy of the tweet-article matching.

To evaluate the accuracy of the SOLR-provided tweet/article matches, we have selected 5 tweets from each of the seven news categories as test tweets (35 test tweets total). We randomly selected 5 tweets from the tweets database that were good matches for the categories in our news article database. Each test tweet was queried against the SOLR Apache server to retrieve the top matching news articles. For each tweet's result set, we examined the category associated with each of the top 5 articles. For each of those articles, we examined the category associated with the article and compared that article with the category associated with the tweet to judge the accuracy of our tweet/article matching module.

The results of our evaluation are shown in Table X and also Figure X. Overall, we have an 88% accuracy in the top 5 articles matched against a tweet. That is, 88% of the time, the articles retrieved in response to a given tweet match the category to which the tweet is related. Our best performance is with Sports related tweets (96%) and the worst is with Health (76%).

Table 10: Accuracy of tweet/category matching

Category	Accuracy
Sports	96%
Crime	92%
Business	88%
Politics	88%
Tech	84%
Health	76%
Entertainment	96%
Overall	88%

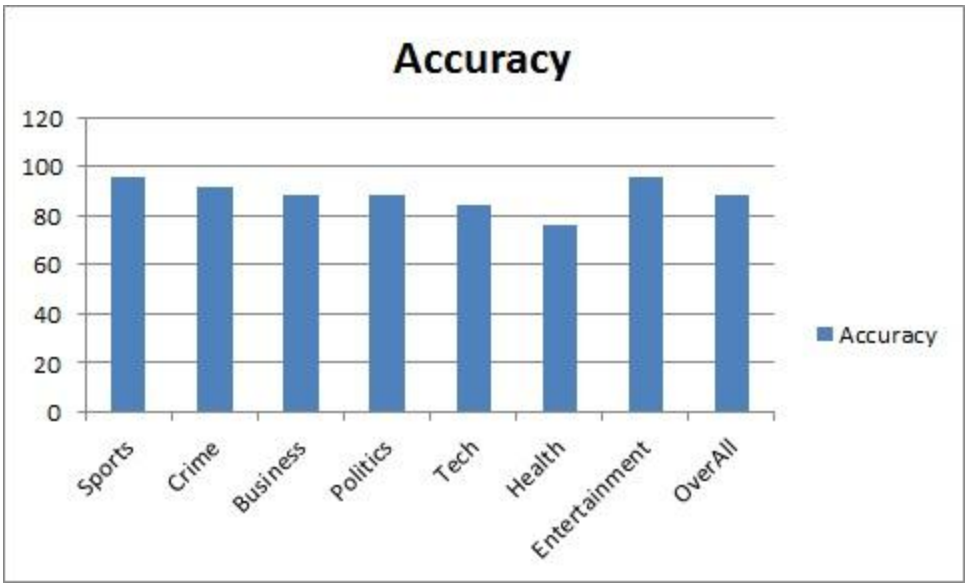


Figure 13: Accuracy of tweet/category matching

4.2 Experiment Two: Evaluating the Recommender Systems

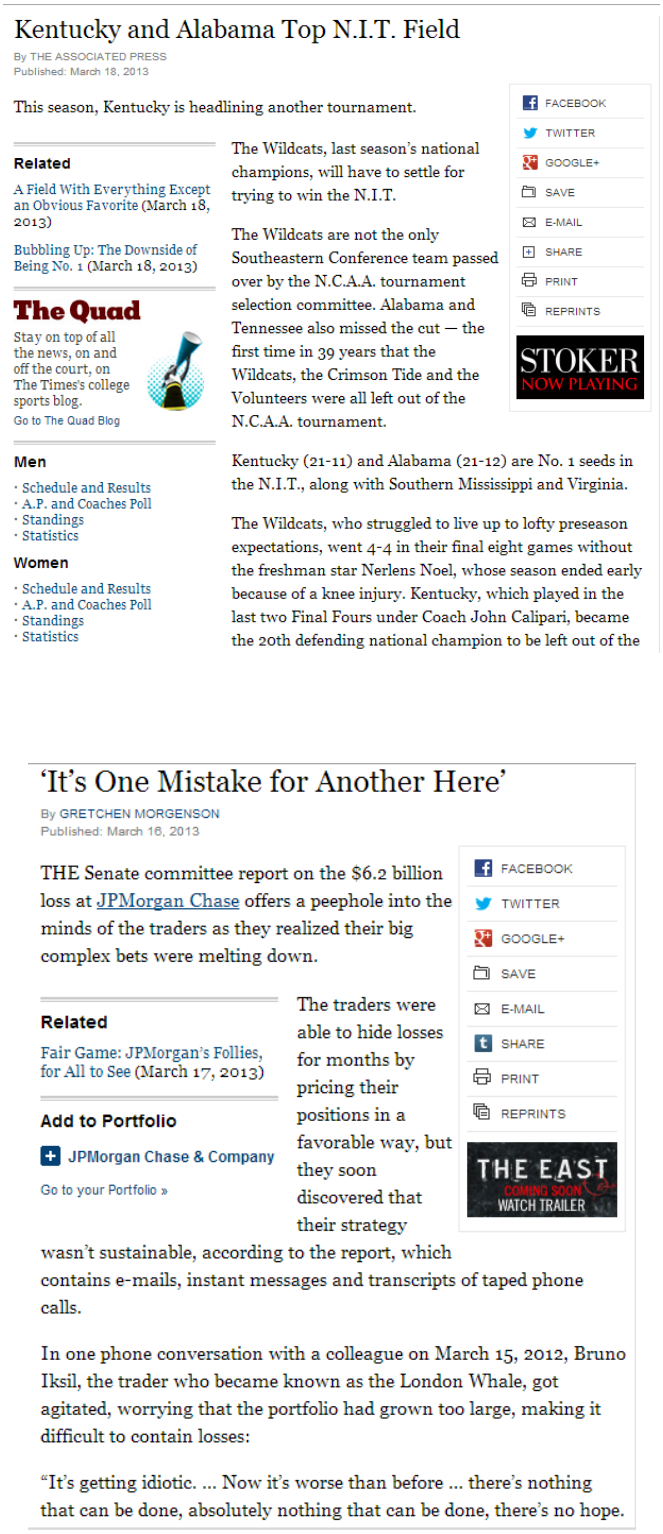
The datasets for this experiment are the same as in Experiment One, i.e., a collection of over 100,000 tweets from the Twitter's public timeline and a set of 288 RSS news articles from the websites of CNN and the New York Times collected on the same day. The news articles come from seven categories.

4.2.1 Popularity-Based Recommender System

As described in Section 3.2, we match all 100,000 tweets to our 288 news articles using SOLR. After all the tweets are processed, the documents are sorted by their total scores. The scores are normalized by dividing by the highest score of every article, yielding values in a range of 0.0 to 1.0. Finally, the top ten highest-scored documents are recommended to the user based on their popularity as measured by Twitter activity. Table 11 shows the top ranked documents and scores. Figure 13 shows the contents of the two most popular articles for 18th March 2013.

Table 11: The top 10 most popular articles

Document	Weight
131.html	1.000000
321.html	0.674333
120.html	0.609006
122.html	0.590006
609.html	0.579575
540.html	0.560575
521.html	0.518471
739.html	0.515005
439.html	0.504768
226.html	0.491905



Kentucky and Alabama Top N.I.T. Field

By THE ASSOCIATED PRESS
Published: March 18, 2013

This season, Kentucky is headlining another tournament.

FACEBOOK
TWITTER
GOOGLE+
SAVE
E-MAIL
SHARE
PRINT
REPRINTS

STOKER
NOW PLAYING

Related

A Field With Everything Except an Obvious Favorite (March 18, 2013)

Bubbling Up: The Downside of Being No. 1 (March 18, 2013)

The Quad

Stay on top of all the news, on and off the court, on The Times's college sports blog.

Go to The Quad Blog

Men

· Schedule and Results
· A.P. and Coaches Poll
· Standings
· Statistics

Women

· Schedule and Results
· A.P. and Coaches Poll
· Standings
· Statistics

The Wildcats, last season's national champions, will have to settle for trying to win the N.I.T.

The Wildcats are not the only Southeastern Conference team passed over by the N.C.A.A. tournament selection committee. Alabama and Tennessee also missed the cut — the first time in 39 years that the Wildcats, the Crimson Tide and the Volunteers were all left out of the N.C.A.A. tournament.

Kentucky (21-11) and Alabama (21-12) are No. 1 seeds in the N.I.T., along with Southern Mississippi and Virginia.

The Wildcats, who struggled to live up to lofty preseason expectations, went 4-4 in their final eight games without the freshman star Nerlens Noel, whose season ended early because of a knee injury. Kentucky, which played in the last two Final Fours under Coach John Calipari, became the 20th defending national champion to be left out of the

'It's One Mistake for Another Here'

By GRETCHEN MORGENSON
Published: March 16, 2013

THE Senate committee report on the $6.2 billion loss at JPMorgan Chase offers a peephole into the minds of the traders as they realized their big complex bets were melting down.

FACEBOOK
TWITTER
GOOGLE+
SAVE
E-MAIL
SHARE
PRINT
REPRINTS

THE EAST
COMING SOON
WATCH TRAILER

Related

Fair Game: JPMorgan's Follies, for All to See (March 17, 2013)

Add to Portfolio

JPMorgan Chase & Company

Go to your Portfolio »

The traders were able to hide losses for months by pricing their positions in a favorable way, but they soon discovered that their strategy wasn't sustainable, according to the report, which contains e-mails, instant messages and transcripts of taped phone calls.

In one phone conversation with a colleague on March 15, 2012, Bruno Iksil, the trader who became known as the London Whale, got agitated, worrying that the portfolio had grown too large, making it difficult to contain losses:

"It's getting idiotic. ... Now it's worse than before ... there's nothing that can be done, absolutely nothing that can be done, there's no hope.

Figure 14: The contents of the two most popular articles [32]

The top two articles are related to Sports (basketball) and Politics (Senate review of JP Morgan's losses), often the most active areas of discussion on Twitter.

4.2.2 Profile-Based Recommender System

In addition to the news article dataset used in the Experiment One, this recommender system also requires a set of user profiles. The tweets dataset is unused. We enlisted a set of 27 volunteers who manually created user profiles that summarized their interests in the seven categories related to our news article dataset (See Section 3.2). The volunteers were international graduate students from different universities in the US.

As described in Section 3.3, we first categorize each of our 288 news articles and create a Lucene index that stores the top 3 categories (and their similarity values) for each article. We then use SOLR to match user profiles to the news articles by treating profiles as queries. Once again, the scores are normalized by the score of the top-weighted document, yielding values in the range 0.0 to 1.0. The top ten highest scored documents are then recommended to the user as those best matching their interests.

To demonstrate the accuracy of the above-mentioned approach, we show the results for two different user profiles. Document ids starting with 1 belong to the Sports category and document ids starting 3 and 4 belong to Business and Politics, respectively. Figure 14 shows the results for a user interested only in Sports. You can see that all recommended articles belong to the category Sports, i.e., the document ids all begin with a 1.

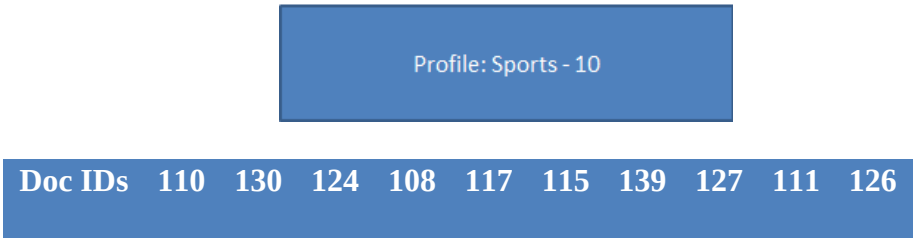


Figure 15: Recommendations for a user interested only in Sports.

Figure 15, on the other hand, shows the results for a user interested in Business and Politics, equally. You can see that the recommended articles are a mixture of Business and Politics since the document ids begin with either 3 or 4.

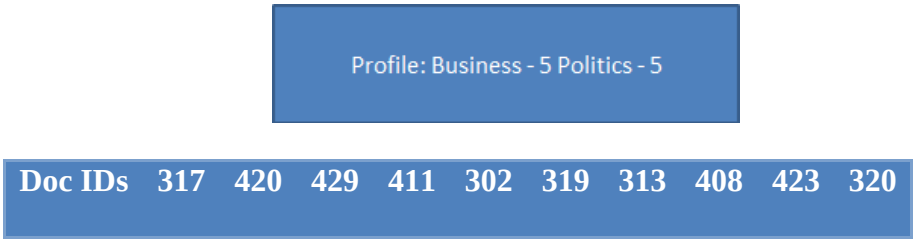


Figure 16: Recommendations for a user interested in Business and Politics

4.2.3 Hybrid-Based Recommender System

As described in Section 3.4, the hybrid recommender system combines the news article weights produced by the two previous systems. The open question is: How much should each sub-system's weights be counted in the final recommendations? For this we re-ranked the

articles presented by the profile-based recommender system with the help of the articles recommended by the popularity-based recommender.

4.2.4 Experimental Setup

Each of the 27 volunteer test subjects was presented with a web page on which enter weights for each of the seven categories. The users enter the weights such they sum to 10. These category/weight pairs form their user profile.

We then create 3 sets of results for each user, corresponding to all the three approaches. Each set contains the 15 highest weighted news articles for that approach. This could, theoretically, require the user to judge 45 different news articles. The lower bound for the number of unique articles, assuming that the popularity-based rankings and the profile-based rankings do not overlap, is 15. In actuality, the average number of unique articles judged by the users was around 35.

In order to avoid bias, we randomized the order of presentation of the news articles to the user. Thus, they did not know which system(s) recommended the news articles nor where the articles were ranked originally. Figure 16 shows a screenshot of the system used to collect the user feedback.

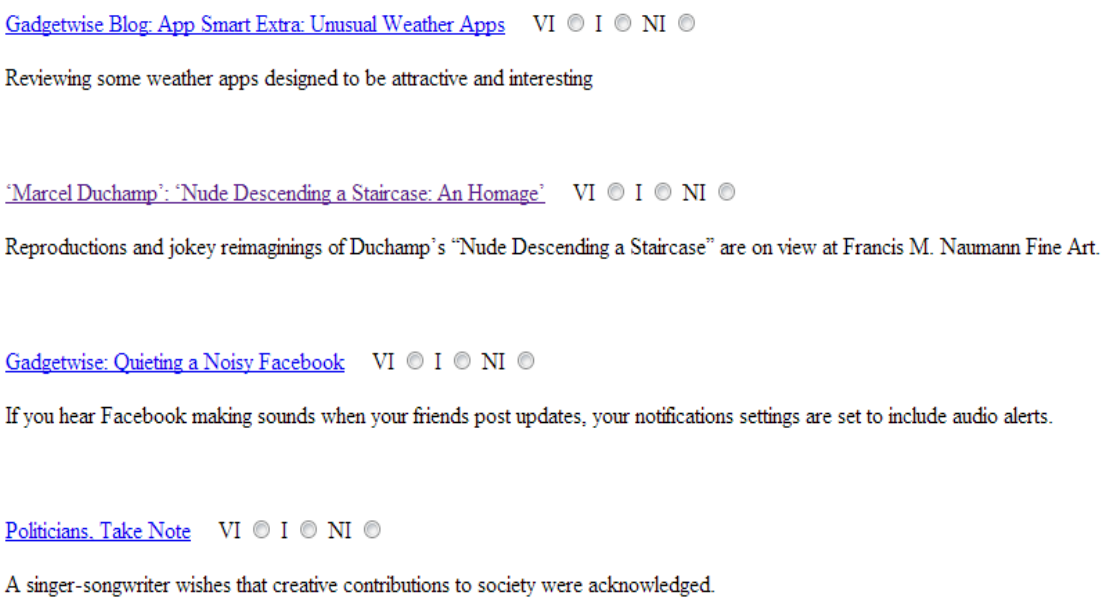


Figure 17: Screenshot of system used to collect user feedback

The user is asked to rate each and every article recommended as very interesting, interesting, or not interesting. Once the user finishes rating all the articles, information such as profile, the article's strategy, rank, weight and the user's rating are logged into a file.

	B	C	D	E	F	G	H	I
1	doc id	S1 rank	S1 wt	S2 rank	S2 wt	S3 rank	S3 wt	Rating
2	721.html	149	0.117198	10	0.924166	43	0.10831	1
3	609.html	5	0.579575	55	0.696775	3	0.403834	1
4	132.html	12	0.475401	272	0.000285	263	0.000135	2
5	730.html	92	0.166346	13	0.919266	27	0.152916	2
6	704.html	22	0.415172	5	0.927555	4	0.385095	0
7	540.html	6	0.560575	168	0.030806	153	0.017269	2
8	131.html	1	1	0	0	0	0	0
9	631.html	11	0.486714	80	0.686379	7	0.33407	1
10	718.html	28	0.374114	20	0.916342	5	0.342816	1
11	701.html	37	0.324158	29	0.915172	10	0.29666	
12	724.html	43	0.277879	6	0.927142	12	0.257633	
13	531.html	14	0.461306	221	0.005698	184	0.002629	
14	626.html	34	0.342414	69	0.689896	15	0.23623	
15	735.html	259	0.04976	4	0.935947	93	0.046573	
16	736.html	156	0.115383	9	0.924402	44	0.10666	
17	122.html	4	0.590006	210	0.007713	169	0.004551	
18	439.html	9	0.504768	170	0.029545	156	0.014914	

Figure 18: Snapshot of a user's response

Figure 17 shows a snapshot of one particular user's response. *S1 rank* shows where the article was ranked by the profile-based recommender and *S1 wt* shows the normalized weight for this result. Similarly, the same information is available for each article returned by any system for S2 (the profile-based recommender) and S3 (the hybrid recommender).

The Rating column indicates how the user rated the article, i.e., 0 for not interesting, 1 for interesting, and 2 for very interesting. These responses are collected from all the 27 users and the results are analyzed and discussed in the following paragraphs.

There are several ways to analyze the results. We will present three different metrics: Average rank; Average rating; and Cumulative rating are discussed here for our case study. The

average rank of all the user responses spanning over the three different strategies is displayed in the following table followed by its corresponding graph.

Table 12: Average rank

	Not Relevant	Relevant	Very Relevant
Popularity	57.24	69.07	58.39
Profile	74.66	62.50	60.45
Hybrid	55.44	53.03	43.22

From Table 12 shows the average rank of the 10 documents presented to the user by each strategy. For example, for documents judged by the users to be very relevant, the popularity system ranked those documents, on average, 58th; the profile-based system ranked very relevant documents 56th on average, and the hybrid system ranked them 39th. For very relevant documents, it is clear that the hybrid system does much better than the other two; those documents are ranked 17-19 spots higher than with the other two systems.

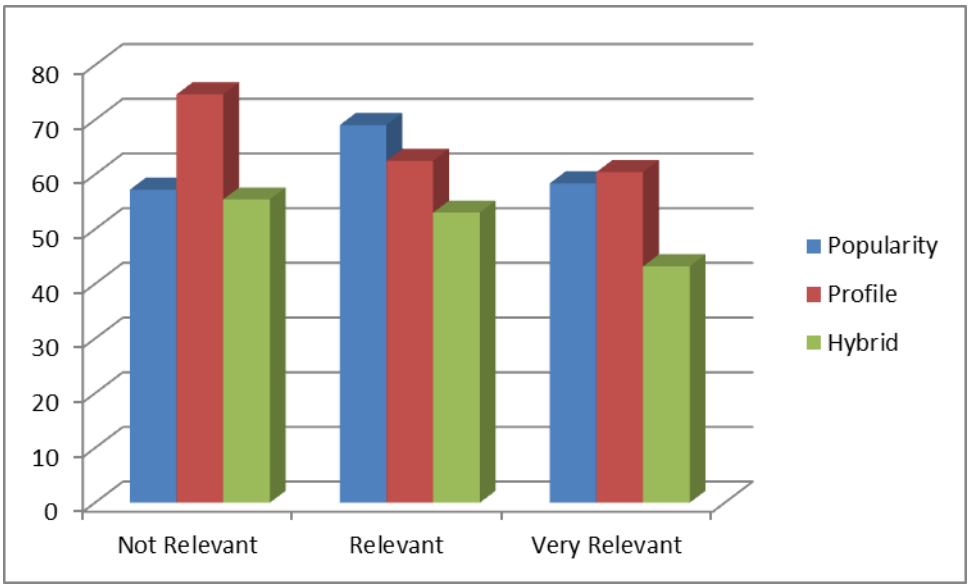


Figure 19: Average Rank

Figure 18 shows the same data graphically. As you can see, hybrid system, no average, ranks the documents a bit higher across all categories, not just documents in the desireable categories. This metric is thus not particularly useful. Changes in rankings of documents by the methods in the lower end (moving a non-relevant document from 80 to 100, for example) have no real effect on the user's experience. Since users only really look at the top 10 documents, we explore other metrics that better capture the important differences between the methods.

A better metric would be to examine the average user rating of the 10 documents presented by each method. The average rating for the three methods are:

Table 13: Average rating of the top 10 articles

Rank Order	Popularity	Profile	Hybrid
1	0.88	1.208	0.958
2	0.95	1.166	1.25
3	0.91	1.347	1.04
4	0.75	0.875	1.24
5	1.24	1.208	1.4
6	1.48	0.875	1.125
7	1.54	1.086	1.12
8	0.66	1.083	1.25
9	0.66	1.166	1
10	1.13	1.25	1.083

As shown in Table X, the average rating fluctuates for all the three approaches. However, the average rating metric does not consider the rank order of the responses. Consider the following synthetic example that displays ratings of 5 articles by a user for two different systems.

System 1

Article Rank	User Rating
1	1
2	1
3	0
4	2
5	2

System 2

Article Rank	User Rating
1	2
2	2
3	1
4	1
5	0

Figure 20: Synthetic user ratings

In the above two tables, the average rating for both the responses is the same. However, because the most relevant articles (those rated a 2 by the user) are ranked higher by System 2, System 2 performs better than System 1. To take rank order into account, we employ a Cumulative Rating metric. The Cumulative Rating preserves the rank order of the article ratings. For the above example considered, a graph is plotted for the cumulative rating of the two strategies and displayed below.

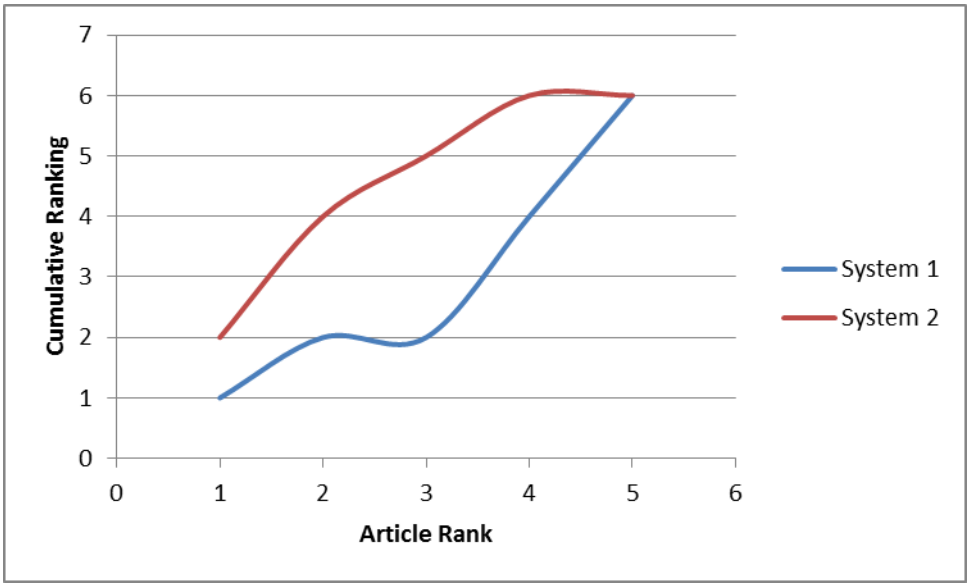


Figure 21: Cumulative rating vs Rank

Table X shows the cumulative rating for all the three strategies. This same date is presented graphically in Figure Y

Table 14: Cumulative rating for all three strategies

Rank Order	Popularity	Profile	Hybrid
1	22	29	23
2	43	56	52
3	64	86	77
4	81	106	107
5	111	134	141
6	147	154	167
7	183	178	194
8	198	203	223
9	213	230	245
10	238	259	270

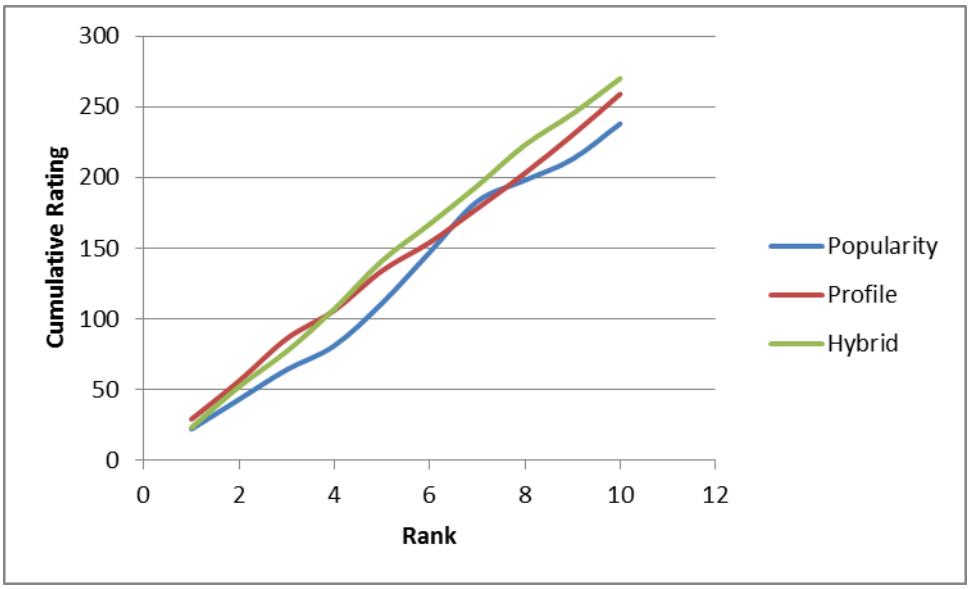


Figure 22: Cumulative rating for all three strategies

From the above results, we see that the profile-based strategy and the hybrid strategies consistently outperform the popularity-based strategy. Users are only interested in, well, things

they are interested in. Very popular articles outside their normal areas of interest are just not very relevant to them. We can also observe that the relevance of the highest ranked articles are essentially tied for the profile-based and hybrid algorithms. However, the articles ranked 5 - 10 by the hybrid system are more relevant than those ranked by the profile-based system alone.

Overall, the results show that the hybrid strategy recommends interesting articles to users and outperforms both the popularity-based and profile-based strategies when used on their own.

5. CONCLUSIONS

5.1 Summary

In this paper, we presented the design and implementation of a news recommender system that incorporates a novel approach to recommend interesting news articles to the user.

We implemented three different strategies to recommend news articles to the user that are interesting to read. The first strategy was to recommend the news articles based on their popularity as identified with the help of tweets from Twitter's public timeline. The second strategy was to recommend the news articles based on their match to categories in the user's profile. The third strategy is a hybrid approach that fuses the first two strategies to present a novel approach that recommends news articles to the user based on both the article's popularity and similarity to the user's profile.

In order to evaluate our approach, we ran several experiments to test the accuracy of the three strategies. We have compared each of the three strategies using several metrics based on the user's feedback on the news articles recommended. To summarize our experimental results, the hybrid approach was 53% accurate in the top Y articles versus 44% for the popularity-based recommender and 51% for the profile-based recommender.

We draw the following conclusions from the results obtained from the experiments:

1. Our hybrid approach outperforms the popularity and profile-based recommendations. In other words, the news articles recommended by the hybrid approach are more relevant to the user as compared to the other two strategies.
2. The profile-based strategy provides slightly better recommendations as compared to the popularity-based strategy.

Thus, we demonstrated that our hybrid approach, identifying popular articles in areas of interest to the user, is more effective than either of the two approaches alone.

5.2 Future Work

The hybrid approach we implemented in this thesis can be improved in several ways to increase the precision of our news recommender system. We considered two different dimensions (popularity and profile) to implement the hybrid approach. The accuracy of our news recommender system can be improved by considering another dimension "location", which could be implemented to recommend news articles to the user based on his location. Also, our users provided explicit feedback about the categories in which they were interested. The recommender system could be improved by implicitly inferring the users interests based on their reading habits. Finally, the experiments were conducted with only 27 users reading articles from seven categories. We should explore the effectiveness of our system on a wider range of news articles in order to improve or adapt our algorithms for greater scalability and applicability.

REFERENCES

[1] S. G. Esparza, M. P. O'Mahony, and B. Smyth, "On the Real-time Web as a Source of Recommendation Knowledge," in *RecSys 2010*, Barcelona, Spain, September 26-30 2010.

[2] A. Jackoway, H. Samet and J. Sankaranarayanan, "Identification of Live News Events using Twitter," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, New York, New York, USA, 2011.

[3] M. Pazzani and D. Billsus, "Content-based Recommendation Systems," in *The Adaptive Web, pages 325–341*, 2007.

[4] D. Billsus and M. J. Pazzani, "A Personal News Agent that Talks, Learns and Explains," in *The Third Annual Conference on Autonomous Agents, pages 268–275*, May 1999, ACM.

[5] W. IJntema, F. Goossen, F. Frasincar, Hogenboom, "Ontology-Based News Recommendation," in *International Workshop on Business intelligence and the Web*, New York, USA, 2010.

[6] I. Cantador, A. Bellog´ın, and P. Castells, "News@hand: A Semantic Web Approach to Recommending News," in *AH 2008, pages 279–283*, Berlin, Heidelberg, 2008.

[7] O. Phelan, K. McCarthy and B. Smyth, "Using Twitter to Recommend real-time Topical News," in *Proceedings of the Third ACM Conference on Recommender Systems*, October 23-25, New York, New York, USA, 2009.

[8] O. Phelan, K. McCarthy, M. Bennett and B. Smyth, "Terms of a Feather: Content-based news recommendation and discovery using Twitter," in *Proceedings of the Thirty-Third European Conference on Advances in Information Retrieval*, Berlin, 2011.

[9] 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*, March 28-April 1, Hyderabad, India, 2011.

[10] J. Zhang and P. Pu, "A Recursive Prediction Algorithm for Collaborative Filtering Recommender Systems," in *Proceedings of the 2007 ACM Conference on Recommender Systems, pages 57-64*, New York, New York, USA, 2007.

[11] A. Tuzhilin and G. Adomavicius, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," in *IEEE Transactions on Knowledge and Data Engineering, pages 734-749*, New Jersey, USA, 2005.

[12] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," in *Communications of the ACM, pages 66-72*, New York, New York, USA, March 1997.

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

[14] Y. Koren, "Factor in the Neighbors: Scalable and Accurate Collaborative Filtering," in *ACM Transactions in Knowledge Discovery from Data (TKDD)*, January 2010.

[15] A. S. Das, M. Datar, A. Garg and S. Rajaram, "Google News Personalization: Scalable Online Collaborative Filtering," in *Proceedings of the 16th International Conference on World Wide Web, pages 271-280*, New York, New York, USA, 2007.

[16] R. J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning For Text Categorization," in *Proceedings of the fifth ACM conference on Digital Libraries, pages 195-204*, New York, New York, USA, 2000.

[17] K. Lang, "NewsWeeder: Learning to Filter Netnews," in *Proceedings of the Twelfth International Conference on Machine Learning, pages 331-339*, San Francisco, CA, 1995.

[18] B. Krulwich and C. Burkey, "Learning User Information Interests through Extraction of Semantically Significant Phrases," in *Proceedings of the AAAI spring symposium on machine learning in information access, pages 100-112*, 1996.

[19] M. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & Webert: Identifying Interesting Web Sites," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 54-61*, Portland, OR, August 1996.

[20] Y. Lu, S. Yu, T. Chang, J. Y. Hsu, "A Content-Based Appraoch to Enhance Tag Recommendation," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pages 2064-2069*, 2009.

[21] J. Pagonis and A. Clark, "Engene: A Genetic Algorithm Classifier for Content-Based Recommender Systems that Does Not Require Continuous User Feedback," in *UKCI, pages 1-6*, 2010.

[22] H. Yu and F. Zhang, "Collaborative Filtering Recommender System in Adversarial Environment," in *International Conference on Machine Learning and Cybernetics (ICMLC)*, July, 2012.

[23] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in Proceedings of the Tenth International Conference on WWW, pages 285-295, 2001.

[24] M. McNee, I. Albert, D. Cosley, P. GopalKrishnan, K. S. Lam, M. A. Rashid, A. J. Konstan, J. Riedl, "On the Recommending of Citations for Research Papers," in *Proceedings of the 2002 ACM conference on Computer supported cooperative work, pages 116-125*, 2002.

[25] 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.

[26] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and Evaluating Choices in a Virtual Community of use," in *Conference on Human Factors in Computing Systems - CHI '95*, Denver, May, 1995.

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

[28] N. N. Liu and Q. Yang, "Eigenrank: A Ranking-Oriented Approach to Collaborative Fltering," in *SIGIR, pages 83–90*, 2008.

[29] F. Frasincar, J. Borsje, and L. Levering, "A Semantic Web-Based Approach for Building Personalized News," in *International Journal of E-Business Research, pages 35–53*, 2009.

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

[31] L. Baoli, L. Qin and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," in *Journal of ACM Transactions on Asian Language Information Processing (TALIP), pages 215–226*, 2004.

[32] New York Times website, http://nytimes.com/