Incorporating popularity in a personalized news recommender system

Nirmal Jonnalagedda, Susan Gauch, Kevin Labille and Sultan Alfarhood

Computer Science and Computer Engineering, University of Arkansas, Fayetteville, Arkansas, United States

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 overwhelmed by information of little interest to them. News recommender systems help users manage this flood by recommending articles based on user interests rather than presenting articles in order of their occurrence. We present our research on developing personalized news recommendation system with the help of a popular micro-blogging service, "Twitter." News articles are ranked based on the popularity of the article identified from Twitter's public timeline. In addition, users construct profiles based on their interests and news articles are also ranked based on their match to the user profile. By integrating these two approaches, we present a hybrid news recommendation model that recommends interesting news articles to the user based on their popularity as well as their relevance to the user profile.

Subjects Agents and Multi-Agent Systems, World Wide Web and Web Science
Keywords Twitter, Personalized news recommendation, News recommender systems, User profile

Submitted 10 March 2016
Accepted 9 May 2016
Published 6 June 2016

Corresponding author
Susan Gauch, sgauch@uark.edu

Academic editor
Jason Jung

Additional Information and Declarations can be found on page 20

DOI 10.7717/peerj-cs.63

INTRODUCTION

Owing largely to the ever-increasing volume and sophistication of information on the web, we are able to access an enormous amount of data from around the globe. The downside of this information explosion is that users are often overwhelmed by information of little interest to them. The key challenge for the users is to find relevant information based on their interests. This problem has led to the evolution of recommender systems that help users find the 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 explicit queries.

Many organizations use recommender systems to recommend various types of products to the user. For example, Netflix recommends movies to its users based on the user's movie ratings compared to other similar users' ratings. Amazon recommends various types of products such as gadgets, books, or movies and Pandora Radio recommends music based on a user's past history and preferences. In addition, news recommender systems that recommend news articles from around the globe have become popular. There are many online news services such as Google News and Yahoo News. However, with plenty of news available, the driving problem is to identify and recommend the most interesting articles to each user so that they are not swamped by irrelevant information. These articles should be related to each user interests but also include those news stories that are generating a lot of interest around the globe.

News recommender systems are broadly classified into two types, content-based filtering and collaborative filtering. Content-based filtering methods are based on the information and attributes of the product that is being recommended. This approach focuses on analyzing user's past interest to make future recommendations. Essentially, it recommends products whose contents are similar to the contents of previously viewed products that the user has rated highly. Content-based filtering has some limitations. It can be difficult for the system to learn and understand user preferences from the user's behavior. However, for some products, it is possible to extract features that have semantic meaning. A popular service that uses this approach is Pandora Radio that uses the properties of a song or artist in order to select a station that plays music with similar attributes. User feedback is used to refine the station's results based on whether the user likes or dislikes the song.

On the other hand, collaborative filtering is a method based on information about the actions of other users whose preferences are similar to the existing user. This method studies the pattern of other user's behavior rather than extracting features from the product. The key advantage of this approach is that prior analysis and understanding of the existing data is not required to make recommendations to the user. However, these systems need a large amount of data from various users in order to make recommendations. Amazon is a popular service that uses item to item (people who buy 'x' also buy 'y') collaborative filtering to recommend products based on other user purchases and reviews. Other popular examples that use this approach are the social networking sites that recommend new friends, groups and other social connections to the user.

In this paper, we develop a hybrid personalized news recommender system that recommends interesting news articles to the user using a micro-blogging service "Twitter." Our news recommender system ranks the articles in different ways: (1) We consider the user's profile to recommend articles to the user; and (2) we also consider the article's popularity with the help of tweets from Twitter's public timeline. This paper presents a novel approach to help users find interesting articles to read by merging the above two methods of selecting articles.

The next section reviews the relevant literature followed by a section explaining the design and implementation of the news recommender system. We continue with a section describing the evaluation of our system and a discussion and analysis of the results of the experiments conducted. We conclude with a summary of the research and briefly outline future work.

LITERATURE REVIEW

Recommender systems

Recommender systems are widely used to help readers filter through an ever-growing flood of information. These systems implement an information filtering method to select products from a stream of information. Also, recommender systems collect data from 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 (*Tuzhilin & Adomavicius, 2005*). Recommender systems are classified into three categories, based on how the recommendations are made (*Balabanovic & Shoham, 1997*).

1. Content-based recommender systems: These recommender systems recommend an item to the user similar to the ones the user preferred in the past.
2. 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. They have the advantage that they can recommend items for which little or no semantic information is available (music, movies, books).
3. 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 (*Su & Khoshgoftaar, 2009*). 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 scarcity, 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 (*Koren, 2010*). 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 scarcity 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 (*Das et al., 2007*).

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 features extracted from the content of the items themselves. The features associated with the items are then matched with the user's profile to make recommendations. This approach is most commonly used by applications whose items have sufficient associated text from which keywords can be extracted. Recommendations are done based on the similarity of keywords associated with the items and the keywords associated with the user profile. The user profile's keywords can be manually supplied (explicit) or identified by mining keywords from items viewed, liked, or purchased by the user (implicit).

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 Term Frequency – Inverse Document Frequency (TF-IDF) method.

Examples of the earliest traditional content-based recommender systems include (*Lang, 1995*; *Krulwich & Burkey, 1996*; *Pazzani, Muramatsu & Billsus, 1996*). *Lang (1995)* implemented a Netnews filtering system, "Newsweeder," that addresses the user profile building problem by letting the user enter 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. *Lang (1995)* used two approaches TF-IDF and Minimum Description Length (MDL) for term weights. 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.

Krulwich & Burkey (1996) and *Pazzani, Muramatsu & Billsus (1996)* conducted similar work by developing intelligent information agents in their Information Finder system 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 (2000) developed a next-generation content-based recommender system that utilizes information extraction and a machine-learning algorithm for text categorization. Their prototype system, Learning Intelligent Book Recommending Agent (LIBRA), 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. Preprocessing 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 two different users rated the books. 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, *Kang, Doornenbal & Schijvenaars (2015)* introduced the Elsevier journal finder. The proposed service is a content-based recommender system that recommends Elsevier journals that have published papers related to the one an author is willing to submit. It covers all major scientific domains available in the Elsevier database and about 2,900 per-reviewed journals. The system typically uses natural language processing (NLP) techniques to generate noun phrases features from papers which will then be used to do the matching. They are extracted using pattern of part-of-speech tag sequences and are represented using the Backus-Naur-form. After being extracted, the noun phrases are mined and normalized by the Elsevier Fingerprint Engine (EFE) which consist of several NLP techniques used to generate relevant annotations (sentence boundaries, tokens, part-of-speech tag, phrase chunking . . .). The well-known Okapi BM25 algorithm is used to do the matching between the submitted query and the papers in the database, but instead of using bag-of-words as input they use the previously generated noun phrase annotations. As an output, they produce a list of ranked paper along with their respective Bm25 score. Finally, they compute an average BM25 score for each journal by averaging the score of all papers published in this journal.

Tkalcic et al. (2013) introduce a content-based recommender system for images that uses affective labeling of multimedia content. The system uses emotion detection techniques and a *k*-nearest neighbor machine learning algorithm to create affective labels.

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 (*McNee et al., 2002*). 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 a user is interested in an item, similar items are also recommended to the user (*Yu & Zhang, 2012*; *Sarwar et al., 2001*). 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 rating information is collected. Each user is provided with a collection of items for him to rate according to his interests. Each user is thus 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 most commonly used metrics to measure the similarity between pairs of vectors.

The next task is to identify the neighboring users who will serve as recommenders. There are two general techniques, threshold-based and top-n. With the 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.

Examples of the earliest traditional collaborative recommender systems include GroupLens (*Resnick et al., 1994*) and Bellcore's video recommendation system (*Hill et al., 1995*) that recommended news articles and videos to, respectively. Although successful first approaches, these traditional collaborative recommender systems often faced challenges such as scarcity, 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. The cold-start problem is one of the most difficult to deal with. When the first users interact with a new recommender system, there are no previous user preferences to exploit in order to generate any recommendations at all.

To overcome the above-mentioned problems, *Gong (2010)* 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 (2010)* is explained in detail 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 of item clustering, predictive strategies are applied to supply missing 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 the information gathered, the weighted average of the neighbors' ratings is calculated, weighted by their similarity to the target item to produce recommendations to the users.

Gong (2010) created a dataset containing 100,000 ratings from 1,000 users on 1,680 movies with every user providing at least 20 ratings. The ratings were 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 indicated that his algorithm outperformed traditional collaborative recommendation algorithms.

Recently, *Li et al. (2015)* present Rank-GeoFM, a Point Of Interest (POI) recommender system that uses both context-aware and collaborative filtering techniques. Their approach differs from the traditional ones because they obtain geographical factorization in a ranking based way. Moreover, by incorporating different types of context information and by using a stochastic gradient descent-based algorithm, they are able to address the data scarcity problem common to this type of recommender system.

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 the user's 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.

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 current news stories. 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. *Jackoway, Samet & Sankaranarayanan (2011)* discuss a method to determine which Twitter users are posting reliable information and which posts are interesting.

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 reviews of movies and used this technique with various movie recommendation strategies on live-user data (*Esparza, O'Mahony & Smyth, 2010*). *Phelan, McCarthy & Smyth (2009)* focus on using micro-blogging activity to recommend news stories. Their recommender system, *Buzzer,* is applied to RSS feeds to which the users have subscribed. *Buzzer* mines terms from RSS and Twitter feeds and uses them to rank articles. *Phelan et al. (2011a)* and *Phelan et al. (2011b)*, they extended their work by considering the public-rank and the friends-rank strategy rather than just considering the articles from the users' index.

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,* *Muramatsu & Billsus (1996)* created *News Dude*, a personal news-recommending agent that uses TF-IDF in combination with a Nearest Neighbor algorithm in order to recommend news stories to users (*Billsus & Pazzani, 1999*). 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, *Li et al. (2014)* described a content-based recommender system that recommends news articles to a user based upon the user's short-term and long-term reading preferences.

The work from *Abel et al. (2011)* presents a good correlation between user profile features and its relative efficiency for recommendations. They evaluate and compare three different strategies for building user profile upon Tweeter stream. The three types of user profile are: entity-based user profiles, hashtag-based user profiles, and topic-based user profiles. They concluded that entity-based user profiles are the most valuable profiles and that they are a better fit for recommendation purposes. They then used this type of profile in their personalized news recommender system.

Recently, *Oh et al. (2014)* proposed an innovative recommender system to recommend personalized news stories using content-based analysis of tweets in Twitter. In their work, they first build a user profile by extracting keywords from the content of tweets, re-tweets and hashtags. A keyword classifier based on deep neural network analysis is being used to classify interesting keywords. Then, they recommend news articles to the user using topic modeling techniques as well as TF-IDF.

Our news recommender system incorporates both the strategies explained above, popularity and conceptual user profiles, to present a novel hybrid approach to recommend news articles to the user that are both relevant to their interests and popular with a wide audience.

APPROACH

In this section, we present an overview of our hybrid news recommendation system. 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 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.

Finally, our intuition is that users most want to see news stories related to topics in their profile that are also creating a buzz on the blogosphere.

In other words, users are shown hot stories related to their favorite topics.

High-level design

Figure 1 shows an architectural diagram of our hybrid news recommender system.

The hybrid system consists of three modules:

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

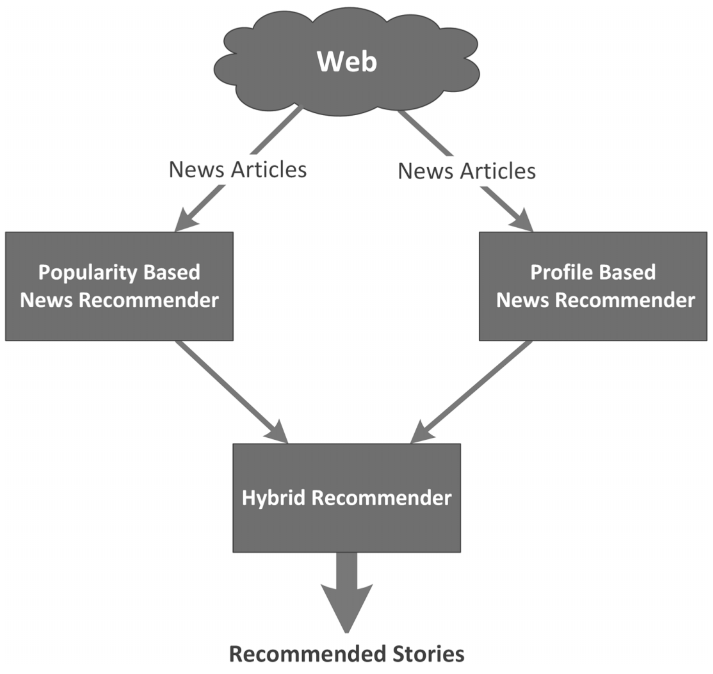

Figure 1 **Architecture of the hybrid news recommender system.**

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 tweets from the Twitter's public. 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 on their similarity to a user's profile. The news articles are ranked based on their 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 to modules to recommend interesting 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 that are relevant to topics interesting to the user.

Popularity-based news recommender system

Figure 2 shows an architectural diagram of the popularity-based news recommender system. First, the RSS articles are collected from a news source such as CNN or the BBC and stored in the file system. These websites organize their stories by category, e.g., Sports, Business, Politics, and Entertainment etc. The file system stores all the articles organized into folders based on the category to which they belong. The RSS articles are pre-processed to remove unnecessary content (html tags, numbers, etc.) while preserving the textual content.

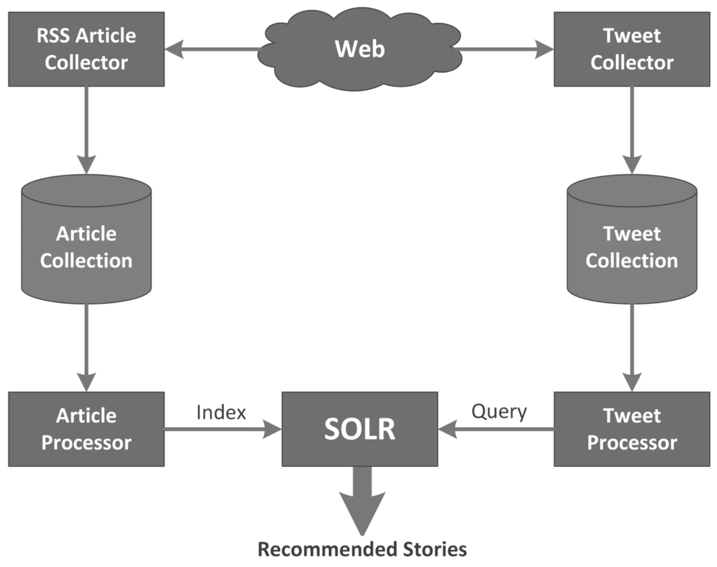

Figure 2 **Architecture of the popularity-based news recommender.**

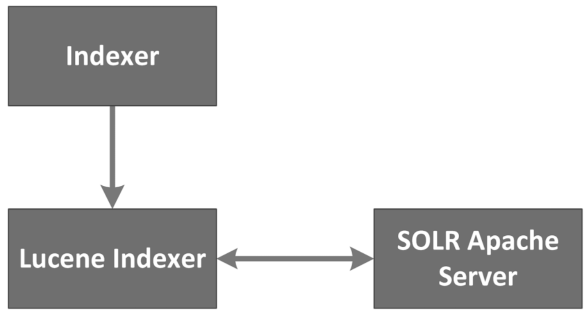

Figure 3 **SOLR architecture.**

The pre-processed articles are then indexed by SOLR, an open source enterprise search platform from the Apache Lucene project (Apache SOLR; http://lucene.apache.org/solr/). 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 that 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.

In order to identify which news stories are most popular, we also collect data from the Twitter micro-blogging site. The Tweet Collector collects the tweets from Twitter's public timeline via the 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

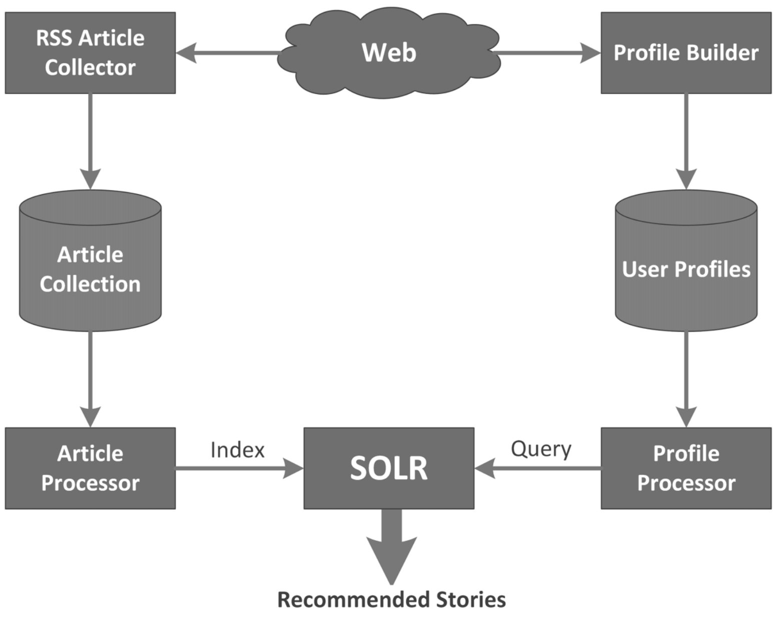

Figure 4 **Architecture of the profile-based recommender system.**

Collection by eliminating unwanted noise and preserving the tweet content. Each processed tweet is queried against the server to retrieve the articles that match the tweet contents. SOLR returns articles, and weights, based on how well each article matches the tweet according to the cosine similarity value. The weights for each article are accumulated across all tweets to produce a popularity weight for the article. Thus, the *Popularity_Wt* for article *i* is the sum of the cosine similarity of all matching current tweets:

$$Popularity\_Wt_i = \sum_{t \in T} cosineSimilarity(Article_i,\ Tweet_t)$$

where

T is the number of tweets collected

Article$_i$ is the news article whose popularity is being calculated

Tweet$_t$ is the tweet being matched against the article

Profile-based news recommender system

In this section, we describe the profile-based news recommender system which is used as a baseline later on for evaluating the effectiveness of our different approaches. This system is comprised of four components and each component is explained in detail in the following paragraphs. Figure 4 shows the architectural diagram of the Profile-Based News 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 (*Shiwen, Baoli & Qin, 2004*), the classification module of the KeyConcept project (*Gauch, Ravindran & Chandramouli, 2010*). 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. These categories are then sorted based on their accumulated scores. 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 again use SOLR to build a second Lucene index that maps from category ids to document ids and weights.

Next, each user creates his or her own profile by manually scoring the categories presented on a Web form. This 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 inner product between the user profile's category vector and the article's category vector. Thus, for a given user *j*, the *Personal_Wt* for article *i* is:

$$Personal\_Wt_{ij} = CosineSimilarity\left(ArticleProfile_i,\ UserProfile_j\right)$$

where

ArticleProfile$_i$ is a vector of the weights for each category in the ontology for Article$_i$
UserProfile$_j$ is a vector of the weights for each category in the ontology for User$_j$

To implement this, we reuse SOLR, querying with the Lucene index that stored the article category vectors with the user's profile.

Hybrid news recommender system

The hybrid recommender module combines the weights provided by each of the previous two modules to produce a recommendation based on both the articles popularity to users everywhere and the article's likely interest to the particular user. We first experimented with multiplying the two factors together. This module calculates a hybrid weight by combining the two scores. The hybrid weight for article *i* and user *j* is given by:

$$Hybrid\_Wt1_{ij} = Popularity\_Wt_j * Personal\_Wt_{ij}$$

Next, we incorporate a tunable parameter, α, that controls how much each of the two components contributes.

$$Hybrid\_Wt2_{ij} = \ \alpha * Popularity\_Wt_j + (1-\alpha) * Personal\_Wt_{ij}$$

When α is 0.0, only the *Personalized_Wt* contributes to the overall weight. As α increases from 0.0–1.0, the *Popularity_Wt*'s contribution increases and the *Personalized_Wt*'s

Table 1 **Data set for the sample scenario.**

	Article B1	Article B2	Article B3	Article E4	Article S5	Article S6
Tweet 1	0.6					0.7
Tweet 2	0.3	0.1				
Tweet 3	0.5			0.9		
Tweet 4			0.5	0.4		
Tweet 5		0.2			0.2	
Tweet 6		0.1	0.1			
Tweet 7		0.1				0.3

Table 2 **Popularity-based ranking of the articles.**

Article	*Popularity_Wt*
B1	1.4
E4	1.3
S6	1.0
B3	0.6
B2	0.5
S5	0.2

contribution decreases. Eventually, when α is 1.0, only the *Popularity_Wt* influences the ranking.

Sample scenario

We now include a sample scenario to illustrate how the various recommender systems work. Consider the following example consisting of six news articles collected from online news sites and seven recent tweets collected from Twitter. Assume that articles B1, B2, and B3 were related to business, article E4 is about entertainment, and that articles S5 and S6 were about sports. Furthermore, assume that these three categories are the only categories in the ontology and that the system's goal is to recommend three articles to the user.

Popularity-based recommender

We will begin describing the popularity based recommendation system that recommends news articles based only on the number of tweets related to the articles. The news articles are indexed with the SOLR Lucene indexer and the tweets queried against that collection to identify the most relevant two articles for each tweet and the associated cosine similarity scores. The cosine similarity scores for each article are then accumulated to produce the *Popularity_Wt* for each article. Table 1 shows the top two cosine similarity scores for each tweet with respect to each article.

The popularity based recommender would then accumulate the cosine similarity scores for each article and the resulting aggregated value is known as the *Popularity_Wt* for that article. The articles would be sorted by in decreasing order by *Popularity_Wt* as shown in Table 2.

Table 3 **User profile.**

Category	Weight
Business	6
Entertainment	1
Sports	3

Table 4 **Articles and their category-match weights.**

Articles	Business Wt	Entertainment Wt	Sports Wt
B1	0.3	0.2	0.0
B2	0.7	0.0	0.6
B3	0.4	0.7	0.0
E4	0.0	8.0	0.2
S5	0.6	0.1	0.0
S6	0.4	0.1	0.0

Table 5 **Profile-based ranking of the articles.**

Article	*Personal_Wt*
B2	6.0
S5	3.7
B3	3.1
S6	2.5
B1	2.0
E4	1.4

The popularity-based news recommender system would thus produce the following recommendations, in order: Article B1, Article E4, and Article S6. Note the three recommendations are about three different topics, i.e., business, entertainment, and sports respectively.

Profile-based recommender

We will now discuss the profile-based news recommender system in more detail. The user first needs to manually create his/her profile based on his/her interests in various categories. For this simple example, assume that the user creates the profile shown in Table 3.

The news articles in the collection are classified with using a kNN classifier. For each article, the classifier returns a weight for each category that represents the similarity between that category and the article. Table 4 summarizes the results of this classification, showing the weights for each article for the top two matching categories.

The *Personal_Wt* for each article is then produced by calculating the dot product of the category vectors for each article with the category vector for the user profile. For example, the weight for Article B2 would be (6×0.7) in Business + (0×1) for Entertainment + (3×0.6) in Sports for a total weight of 6.0. The articles are then sorted in decreasing order by *Personal_Wt*. Table 5 shows the results of these calculations for this example.

Table 6 **Hybrid ranking of the articles.**

Article	*Popularity_Wt*	Normalized *Popularity_Wt*	*Personal_Wt*	Normalized *Personal_Wt*
B1	1.4	1.00	2	0.33
B2	0.5	0.36	6	1.00
B3	0.6	0.43	3.1	0.52
E4	1.3	0.93	1.4	0.23
S5	0.2	0.14	3.7	0.62
S6	1.0	0.71	2.5	0.42

Table 7 **Hybrid ranking of the articles.**

Article	*Hybrid_Wt*
B2	0.36
B1	0.33
S6	0.30
B3	0.22
E4	0.22
S5	0.09

The profile-based news recommender system would thus produce the following recommendations, in order: Article B2, Article S5, and Article B3. Note that two of the three recommended articles are business-related and the middle recommendation is about sports. This reflects the user's interests as captured by their profile. Furthermore, the top recommendation is for article B2 versus B1 previously. B2 is a better match to the business category whereas B1 was much more popular on Twitter.

Hybrid recommender

We now continue our example using the set of articles in the previous examples to describe the hybrid recommendation system. The *Popularity_Wt* and *Personal_Wt* scores for each article are normalized and the *Hybrid_Wt* is the product of these normalized weights. Table 6 illustrates this process.

The articles are then sorted by decreasing order by *Hybrid_Wt* to produce the final recommendations, as shown in Table 7.

The hybrid news recommender system would thus produce the following recommendations, in order: Article B2, Article B1, and Article S6. Article B2 appears at the top because it is such a strong match for the user's most highly weighted profile category. Articles B1 and S6 appear because they are moderate matches for the user's profile and also quite popular on Twitter, illustrating the effects of the hybrid recommender.

EVALUATION

This section describes the experiments that we conducted to evaluate the accuracy of the recommendations based on popularity, user profile, and the hybrid recommendations. All experiments were conducted on the same collection of news articles 280 news articles collected from CNN and BBC and 202,224 tweets collected from Twitter on the same day.

We collected 40 news articles for each of 7 topics (Sports, Crime, Business, Politics, Tech, Health and Entertainment). The results reported here were produced using 27 volunteer test subjects. To evaluate the relative importance of global popularity versus individual interest in a story, we varied α from 0.0–1.0 in increments of 0.1, a total of 11 values. The profiles-based recommender system alone constitutes our baseline approach, that is, we compared both the popularity-based recommender system and the hybrid recommender system to that baseline.

Evaluating matching tweets to 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.

To evaluate the accuracy of the SOLR-provided tweet/article matches, we have selected five tweets from each of the seven news categories as test tweets (35 test tweets total). We randomly selected five 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 five 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 Fig. 5. Overall, we have 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%).

Evaluating the hybrid recommender

Each volunteer test subject was presented with a web page on which they entered weights from 0–10 indicating their personal interest each of the seven categories. The users entered the weights such they totaled to 10. These category/weight pairs form their user profile. We essentially have two baselines to which we compare our hybrid recommender system, a purely conceptual, personalized recommender system and a purely popularity-based recommender system. These are incorporated into our evaluation of *Hybrid_Wt2*, i.e., when α is 0, only the *Personal_Wt* contributes to the score, so we get purely profile-based recommendations. Similarly, when α is 1, only *Popularity_Wt* contributes to the score, so the user receives purely popularity-based recommendations.

For the articles retrieved by *Hybrid_Wt1* and each of the 11 values of α for *Hybrid_Wt2*, evaluated, we calculated the weight of each article in our collection and sorted the articles by weight. We stored the ArticleID, the *Hybrid_Wt*, and rank for the top 10 articles. Thus, each subject had 12 sets of 10 documents to judge. Different values of α bring new

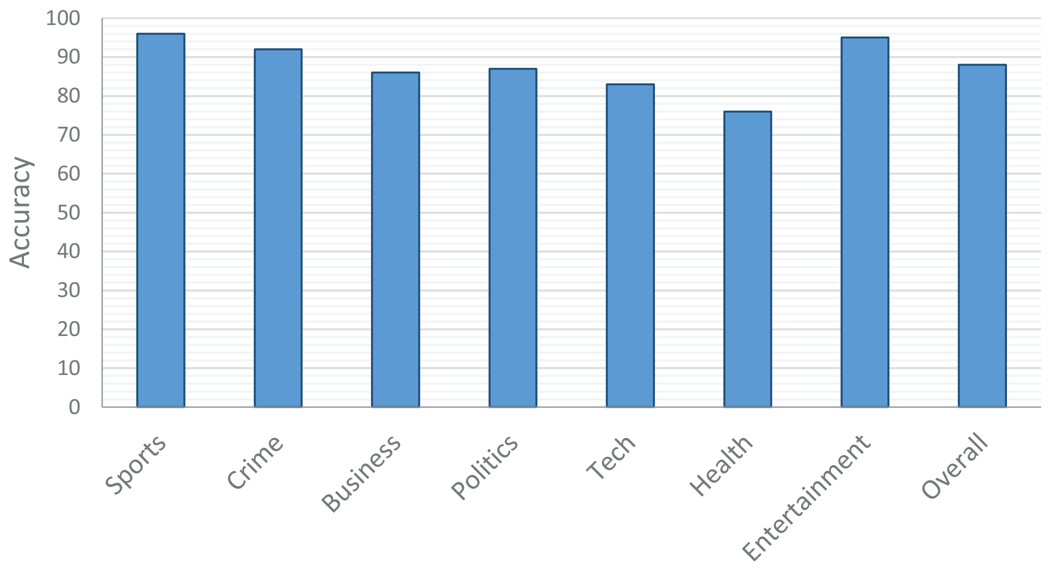

Figure 5 **Accuracy of tweet/category matching.**

articles to the top but they also rearrange the ordering of articles that appear in other result sets. Although subjects could potentially be asked to judge 120 articles, the average number of unique articles judged by the subjects was 45. To avoid user fatigue and gather consistent feedback, each article was presented to be judged only once, and that judgment used to evaluate all the result sets in which the article appeared.

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 or where the articles were ranked originally. The users were asked to rate each 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 were logged into a file for analysis.

Results

We evaluated our recommender systems using the normalized Discounted Cumulative Gain at top k (nDCG@k), a widely used metric for rank ordered lists. We analyzed the results of 14 users who completed the evaluation of the top 10 documents for each method. With that metric, *Hybrid_Wt1* produces a nDCG@10 of 0.616.

Our next task is to tune *Hybrid_Wt2* to identify the best performing value for α so that we can compare our two hybrid approaches. The nDCG@10 over all users as alpha is varied from 0 (pure personalized) to 1 (pure popularity) in steps of 0.1 is depicted graphically in Fig. 6. The worst performance, 0.601, arises when the popularity measure dominates the ratings is considered ($\alpha = 0.9$). When only the *Popularity_Wt* is used, the nDCG is quite low, 0.0607. In contrast, when the only user's profile is considered ($\alpha = 0.0$), we achieve an nDCG of 0.643.

Our results show that as measured by the nDCG metric, both $\alpha = 0.5$ and $\alpha = 0.6$ outperform all other α values tied with an nDCG of 0.668. These results indicate that the

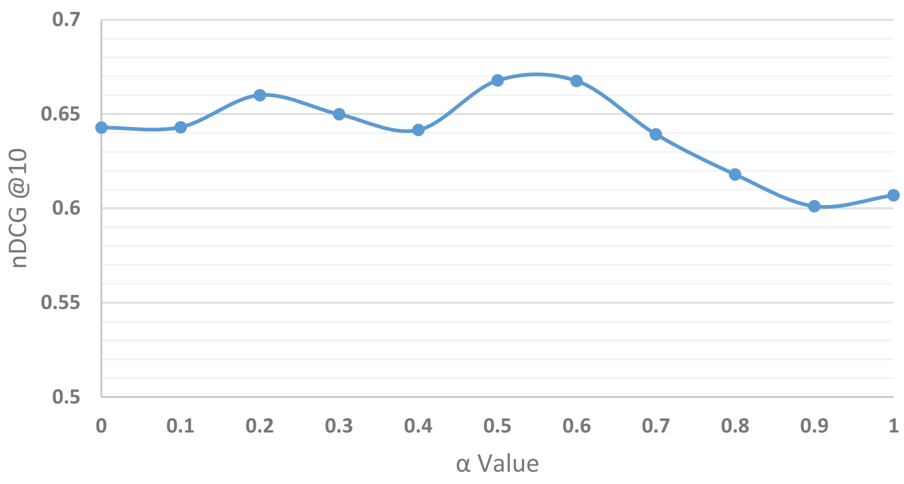

Figure 6 **Evaluating *Hybrid_Wt2*: nDCG@10 versus α.**

Table 8 **nDCG@10 per approach.**

	Personal (α = 0)	Popularity (α = 1)	*Hybrid_Wt1*	*Hybrid_Wt2* (α = 0.5)
nDCG@10	0.643	0.607	0.616	0.668

most relevant articles are those that are selected using a relatively equal combination mixture of their interests and general public's interests.

Table 8 contains a comparison of both hybrid approaches to the baselines of purely personal and purely popular recommendations. Overall, *Hybrid_Wt2* with α = 0.5 outperforms *Hybrid_Wt1* by 7.8%, *Personal_Wt* by 2.4%, and *Popularity_Wt* by 6.0%. We performed a paired two-tailed student's t-test analysis and confirmed that the *Hybrid_Wt2* improvement versus the *Personal_Wt* was statistically significant ($p = 0.002857$). Similarly, the *Hybrid_Wt2* improvement versus the *Popularity_Wt* was statistically significant ($p = 0.000049$) as was the improvement of *Hybrid_Wt2* over *Hybrid_Wt1* ($p = 0.04512$).

CONCLUSION

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 four different strategies to recommend news articles to the user that are interesting to read.

We have evaluated each of the strategies by comparing them to our baseline approach which is the personal recommender system. We found that:

1. Both hybrid approaches outperform the popularity and personal recommendations.
2. The personal recommender provides better recommendations than the popularity-based recommender.
3. The tunable hybrid algorithm with α = 0.4 provided the best overall performance.

We can extend this work in several ways. In particular, the accuracy of our news recommender system can be improved by considering other features such as location or temporal activity. 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.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding
The authors received no funding for this work.

Competing Interests
Susan Gauch is an Academic Editor for PeerJ Computer Science.

Author Contributions
- Nirmal Jonnalagedda conceived and designed the experiments, performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, wrote the paper, prepared figures and/or tables, performed the computation work, reviewed drafts of the paper.
- Susan Gauch conceived and designed the experiments, performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, wrote the paper, reviewed drafts of the paper, submitted manuscript for publication.
- Kevin Labille wrote the paper, prepared figures and/or tables, reviewed drafts of the paper, updated literature review.
- Sultan Alfarhood analyzed the data, contributed reagents/materials/analysis tools, prepared figures and/or tables.

Data Deposition
The following information was supplied regarding data availability:

The raw data includes human subjects information that cannot be shared.

Supplemental Information
Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.63#supplemental-information.

REFERENCES

Abel F, Gao Q, Houben GJ, Tao K. 2011. Analyzing user modeling on twitter for personalized news recommendations. *User Modeling, Adaption and Personalization*. Berlin, Heidelberg: Springer, 1–12.

Balabanovic M, Shoham Y. 1997. Fab: content-based, collaborative recommendation. *Communications of the ACM*. New York: ACM, 66–72.

Billsus D, Pazzani MJ. 1999. A personal news agent that talks, learns and explains. In: *Proceedings of the Third Annual Conference of Autonomous Agents*, New York: ACM Press, 268–275.

Das AS, Datar M, Garg A, Rajaram S. 2007. Google news personalization: scalable online collaborative filtering. In: *Proceedings of the Sixteenth International Conference on World Wide Web, New York, NY, USA*, 271–280.

Esparza SG, O'Mahony MP, Smyth B. 2010. On the real-time web as a source of recommendation knowledge. In: *Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, 26–30 September*, New York: ACM, 305–308.

Gauch S, Ravindran D, Chandramouli A. 2010. KeyConcept: conceptual search and pruning exploiting concept relationships. *Journal of Intelligent Systems* **19(3):**265–288 DOI 10.1515/JISYS.2010.19.3.265.

Gong S. 2010. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *Journal of Software* **5(7):**745–752 DOI 10.4304/jsw.5.7.745-752.

Hill W, Stead L, Rosenstein M, Furnas G. 1995. Recommending and evaluating choices in a virtual community of use. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York: ACM Press/Addison-Wesley Publishing Co., 194–201.

Jackoway A, Samet H, Sankaranarayanan J. 2011. Identification of live news events using Twitter. In: *Proceedings of the Third ACM SIGSPATIAL International Workshop on Location-Based Social Networks, New York, NY, USA*. New York: ACM, 25–32.

Kang N, Doornenbal M, Schijvenaars B. 2015. Elsevier journal finder: recommending journals for your paper. In: *Proceedings of the Ninth ACM Conference on Recommender Systems*, New York: ACM, 261–264.

Koren Y. 2010. Factor in the neighbors: scalable and accurate collaborative filtering. *ACM Transactions in Knowledge Discovery from Data (TKDD)* **4(1):**Article 1 DOI 10.1145/1644873.

Krulwich B, Burkey C. 1996. Learning user information interests through extraction of semantically significant phrases. In: *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. Palo Alto: AAAI Press, 100–112.

Lang K. 1995. NewsWeeder: learning to filter netnews. In: *Proceedings of the Twelfth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 331–339.

Li L, Zheng L, Yang F, Li T. 2014. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications* **41(7):**3168–3177 DOI 10.1016/j.eswa.2013.11.020.

Li X, Cong G, Li XL, Pham TAN, Krishnaswamy S. 2015. Rank-GeoFM: a ranking based geographical factorization method for point of interest recommendation. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM, 433–442.

McNee M, Albert I, Cosley D, GopalKrishnan P, Lam KS, Rashid MA, Konstan AJ, Riedl J. 2002. On the recommending of citations for research papers. In: *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, New York, NY, USA*. New York: ACM, 116–125.

Mooney RJ, Roy L. 2000. Content-based book recommending using learning for text categorization. In: *Proceedings of the Fifth ACM Conference on Digital Libraries, New York, NY, USA*. New York: ACM, 195–204.

Oh KJ, Lee WJ, Lim CG, Choi HJ. 2014. Personalized news recommendation using classified keywords to capture user preference. In: *Proceedings of the Sixteenth International Conference on Advanced Communication Technology, 16–19 Feb*. Piscataway: IEEE, 1283–1287.

Pazzani M, Muramatsu J, Billsus D. 1996. Syskill & Webert: identifying interesting web sites. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Palo Alto: AAAI Press, 54–61.

Phelan O, McCarthy K, Bennett M, Smyth B. 2011a. On using the real-time web for news recommendation & discovery. In: *Proceedings of the Twentieth International Conference Companion on World Wide Web, Hyderabad, India, 28 March–1 April.*

Phelan O, McCarthy K, Bennett M, Smyth B. 2011b. 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, Heidelberg: Springer.

Phelan O, McCarthy K, Smyth B. 2009. Using Twitter to recommend real-time topical news. In: *Proceedings of the Third ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October.* New York: ACM, 385–388.

Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In: *Proceedings of the ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, NC.* New York: ACM, 175–186.

Sarwar B, Karypis G, Konstan J, Riedl J. 2001. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the Tenth International Conference on World Wide Web*, 285–295.

Shiwen Y, Baoli L, Qin L. 2004. An adaptive *k*-nearest neighbor text categorization strategy. *Journal of ACM Transactions on Asian Language Information Processing (TALIP)* **3(4):**215–226 DOI 10.1145/1039621.1039623.

Su X, Khoshgoftaar TM. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* **2009**:19 DOI 10.1155/2009/421425.

Tkalcic M, Odic A, Kosir A, Tasic J. 2013. Affective labeling in a content-based recommender system for images. *IEEE Transactions on Multimedia* **15(2):**391–400 DOI 10.1109/TMM.2012.2229970.

Tuzhilin A, Adomavicius G. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering.* Piscataway: IEEE, 734–749.

Yu H, Zhang F. 2012. Collaborative filtering recommender system in adversarial environment. In: *2012 International Conference on Machine Learning and Cybernetics (ICMLC) 15–17 July.* Piscataway: IEEE.