Combining Collaborative Filtering and Search Engine into Hybrid News Recommendations

Toon De Pessemier
iMinds-Ghent University
G. Crommenlaan 8 / 201
B-9050 Ghent, Belgium
toon.depessemier@ugent.be

Sam Leroux
iMinds-Ghent University
G. Crommenlaan 8 / 201
B-9050 Ghent, Belgium
sam.leroux@ugent.be

Kris Vanhecke
iMinds-Ghent University
G. Crommenlaan 8 / 201
B-9050 Ghent, Belgium
kris.vanhecke@ugent.be

Luc Martens
iMinds-Ghent University
G. Crommenlaan 8 / 201
B-9050 Ghent, Belgium
luc1.martens@ugent.be

ABSTRACT

Recommender systems have proven their usefulness in many classical domains, such as movies, books, and music, in helping users to overcome the information overload problem. When properly configured, recommender systems can also act as a supporting tool for content selection and retrieval in more challenging fields, such as news content. The short life span of news items and the demand for up-to-date recommendations require a specially tailored approach. This paper proposes a hybrid recommender system using a search engine as a content-based approach and combining this with collaborative filtering for diversifying the user profiles. Based on similar users, user profile vectors are extended with related terms interesting to read about. The recommender system is fed real-time streams of news content originating from different sources. The resulting recommendations are clustered into topics and presented through a web application. This paper demonstrates that the advantages of both search engine and collaborative filtering can be successfully combined into a recommender system for domains with transient items, such as news.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Recommender system, Hybrid, Real-time, News, Content-based, Storm, Collaborative filtering

1. INTRODUCTION

Recommender systems are software tools and techniques providing suggestions for items that may be of interest to a user such as videos, songs, or the products of an online shop. Although most research on recommender systems has been performed in these traditional content domains, recommender systems have also been deployed for services that focus on more transient items, characterized by a short life span. Cultural events, such as concerts or theater performances, are only available during a certain time period, in which the recommender has to learn users' preferences for these events as well as generating recommendations for interested users [8]. Reciprocal recommender systems recommend people to people such as for dating, employment, and mentoring services. Recommendations are only successful if both people like each other. However, when people are successful in finding a date, a job, or a mentor, they may never return to the web site [19], thereby limiting the availability in time of candidate recommendations. For news content, items quickly lose their information value and should therefore be recommended as soon as they are available in order to minimize delay between production and consumption of the content. A preview of a sports game has lost any information value after the game for instance. Especially for online news, fast delivering and recommending of content is of utmost importance.

For content with a short life span, and for news content in particular, collaborative filtering (CF) systems have difficulties to generate recommendations because of the new item problem (cfr. cold start problem). CF requires a critical amount of consumptions (explicit or implicit feedback) before an item can be recommended. Once enough consumption data is available, the information value of the content might be degraded, making recommendations for the content useless. Therefore, content-based or hybrid approaches are considered as more suitable for news recommendation [9].

2. RELATED WORK

In the domain of digital news services, various initiatives to personalize the offered news content have been proposed. One of the first recommender systems for personalizing news

INRA 2015 in conjunction with RecSys '15, September 16-20, 2015, Vienna, Austria

content was GroupLens [14]. GroupLens used collaborative filtering to generate recommendations for Usenet news and was evaluated by a public trial with users from over a dozen newsgroups. This research identified some important challenges involved in creating a news recommender system.

SCENE [15] is such a news service. It stands for a SCalable two-stage pErsonalized News rEcommendation system. The system considers characteristics such as news content, access patterns, named entities, popularity, and recency of news items when performing recommendation. The proposed news selection mechanism demonstrates the importance of a good balance between user interests, the novelty, and diversity of the recommendations.

The News@hand system [5] is a news recommender which applies semantic-based technologies to describe and relate news contents and user preferences in order to produce enhanced recommendations. This news system ensures multimedia source applicability. The resultant recommendations can be adapted to the current context of interest, thereby emphasizing the importance of contextualization in the domain of news.

In the CLEF NEWSREEL track [3], news recommendation techniques could be evaluated in real-time by providing news recommendations to actual users that visit commercial news portals. A web-based platform is used to distribute recommendations to the users and return users' impressions of the recommendations to the researchers.

The News Recommender Systems Challenge [22] focused on providing live recommendations for readers of German news media articles. This challenge highlighted why news recommendations have not been analyzed as thoroughly as some of the other domains such as movies, books, or music. Reasons for this include the lack of data sets as well as the lack of open systems to deploy algorithms in. In the challenge, the deployed recommenders for generating news recommendations are: Recent Recommender (based only on the recency of the articles), Lucene Recommender (a text retrieval system built on top of Apache Lucene), Category-based Recommender (using the article's category), User Filter (filters out the articles previously observed by the current user), and Combined Recommender (a stack or cascade of two or more of the above recommenders).

The usefulness of retrieval algorithms for content-based recommendations has been demonstrated with experiments using a large data set of news content [2]. Binary and graded evaluation were compared and graded evaluation showed to be intrinsically better for news recommendations. This study emphasizes the potential of combining content-based approaches with collaborative filtering into a hybrid recommender system for news.

Although the various initiatives emphasize the importance of a personalized news offer, most of them focus on the recommendation algorithms. However, the way in which content is gathered, delivered, and presented to end-users is of crucial importance for a successful service. Users want an up-to-date, personalized news offer, providing a complete overview of all news events, which is clearly structured and classified by topic. In this study, the focus is not on improving state of the art recommendation algorithms or search engines, since many studies covered this already [22, 3, 6, 2]. The focus of this paper is rather on investigating the real-time aspect of delivering personalized recommendations (up-to-date content offer), the aggregation of multiple content sources of a different nature, such as premium content, blogs, Twitter, etc. (complete overview), and the clustering of content items by topic (clearly structured).

The remainder of this paper is structured as follows. Section 3 compares the recommendation and content retrieval problem and indicates resemblances between the two approaches. Section 4 discusses the architecture of our system and zooms in on the data fetching, search engine, recommender, and clustering component of the proposed system. Section 5 provides details on the implementation, the user interaction with the system, and the user interface. Finally, Section 6 draws conclusions.

3. RECOMMENDATION AS A CONTENT RETRIEVAL PROBLEM

Content-based algorithms typically compare a representation of the user profile with (the metadata of) the content, and deliver the best matching items as recommendations [16]. These algorithms often use relatively simple retrieval models, such as keyword matching or the Vector Space Model (VSM) with basic Term Frequency - Inverse Document Frequency (TF-IDF) weighting [17]. As such, the matching process of content and profile in a content-based algorithm shows many resemblances with the content retrieval process of a search engine.

Before employing the VSM and TF-IDF weighting in a content-based algorithm, preprocessing of the content is often required. If the content consists of complete sentences, the text stream must be broken up into tokens: phrases, words, symbols or other meaningful elements. Tokens that belong together, e.g. United States of America or New York, deserve special attention, and can be handled by reasoning based on uppercase letters and n-gram models [4]. Before further processing of the content, the next operation is filtering out stop words, the most common words in a language that typically have a limited intrinsic value. Another important operation is stemming, the process for reducing inflected (or sometimes derived) words to their word stem, or root form. In our implementation, Snowball [20] is used, a powerful stemmer for the English language. Again, a resemblance with content retrieval processes can be noticed, since these preprocessing operations are also performed during the indexing of web pages in search engines.

Based on these similarities between the content recommendation and content retrieval problem, we opted to utilize a search engine as the core component of our recommender service. The user profile is used as search query and provides the input for the search engine. Consequently, the search results are the content items best matching the user profile and can therefore be considered as personalized recommendations for the user.

Utilizing a search engine to generate personalized recommendations for news content brings some additional advantages.

- *Short response time.* Search engines are strongly optimized to quickly identify and retrieve relevant content items. An inverted index [6] is used as a very efficient structuring of the content, enabling to handle massive amounts of documents.
- *Fast processing of new content.* New content items can be processed quickly by making additions to the index structure, thereby making these new content items

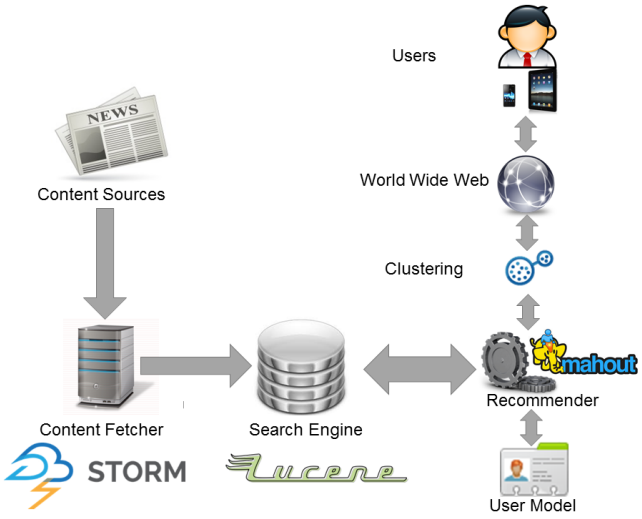

Figure 1: The architecture and content flow of the news recommender system.

available for recommendation almost immediately. In contrast, traditional recommender systems often require intensive calculations of similarities before a new item can be recommended.

- *Limited storage requirements.* The index structure of search engines is a very efficient storage way to retrieve documents.

4. ARCHITECTURE

Figure 1 shows the architecture and content flow of the news recommender system. The different components will be discussed in more detail in this section.

4.1 Data Fetching

The first phase of recommendation process is to *fetch the news content* periodically from different sources. When new items are available, their content is fetched and processed. Many online news services provide their content through RSS-feeds. To parse these feeds, the Rome project [28] is used since this is a robust parser. Besides RSS-feeds, other sources, such as blogs, can also be incorporated into the system by using a specific content parser.

In order to keep track of the most recent news content, news sources are checked regularly for new content. Different news sources have a different publishing frequency, ranging from one news item per day, to multiple news items per minute. Therefore, we used a simple mechanism to adapt the frequency of checking for new content to the publishing frequency of the content source. For each content source, a dynamic timer is used to determine when to check for new content. After a timeout, the content is fetched. If new content is available, the content item is added to the search engine and the timeout is reduced by half. If no new content is available, the timeout is doubled. This simple mechanism showed to be sufficient as a convergence method for the timeout parameter.

In order to process the stream of incoming news articles of different sources continuously, Apache Storm [1] was used. Storm enables the processing of large streams of data in real time. As opposed to batch processing, Storm handles the news articles as soon as these are available. To use Storm, a topology composed of 'Spouts' and 'Bolts' has to be built, which describes how messages flow into the system and how they have to be processed. A Spout is a source of data streams. A Bolt consumes any number of data streams, does some processing, and can emit new data streams. Storm can make duplicates of these components, and even distribute these duplicates over multiple machines, in order to process large amounts of data. As a result, Storm makes the system scalable and distributed.

In our implementation, the Spouts input data into the system as URLs of RSS-feeds, blogs, or social network accounts. Storm will distribute the work load over different Bolts of the first type, which fetch the data from the feeds. In case new articles are available in the feed, the URL of these articles is passed to the Bolts of the second type. These Bolts fetch the article content and remove non-topical information, such as advertisements, by identifying specific HTML tags in the source code of the web page. Subsequently, the Bolts pass the article content to Bolts of the third type. The task of Bolts of the third type is to analyze the content and obtain information such as the title, date, category, etc. Next, the article content is passed to the fourth type of Bolts, which will input the news articles into the search engine. After inputting the content into the search engine, statistical information about the article content is stored by the fifth and last type of Bolts. E.g., the frequency of occurrence of a term at a specific moment in time is used to determine if a news topic is trending and important (Section 4.3).

4.2 Search Engine

In the second phase, the content is processed by a *search engine*. We opted to use Apache Lucene [24], a Java library that is typically used for services handling large amounts of data and offering search functionalities. Since Lucene's performance, simplicity, and ease-of-use have been investigated in related work [12], this research does not focus on the characteristics of Lucene, but rather on the combination of search engine and recommender system.

As alternative search engines, we considered Solr [26] and ElasticSearch [10]. Solr is a ready-to-use, open source search engine based on Lucene. In comparison with Lucene, Solr provides more specific features such as a REST webinterface to index and search for documents. However, the disadvantage of Solr is that some of the specialized functionality is hidden and not directly usable. Besides, the overhead of the webinterface of Solr introduced some delay in comparison with Lucene in our experiments. Similar to Solr, ElasticSearch hides some of Lucene's functionality by using a simple web interface. Specific information about the content items, such as the term frequencies or statistics about the complete index, are not directly accessible using ElasticSearch. Therefore, Lucene was chosen to provide the functionality of the search engine. In case the processing load for the Lucene index becomes an issue, distribution over different machines is possible by solutions such as Katta [13], thereby making it scalable.

4.3 Recommender

In the third phase, personalized recommendations are generated. The user profile is used as a search query and sent to the search engine. The resulting search results are consid-

ered as personalized recommendations. As is common practice in the VSM [16], the user profile is modeled as a vector of terms (tags) together with a value specifying the user's interest in the term. These terms are words (or N-grams) in the article that are identified as relevant for the content. The current implementation is based on the traditional TF-IDF, but alternative solutions can easily be integrated. When the user reads a news article, the profile vector is updated with the TF-IDF values of the terms of the article. However, this update process is only executed if the user has spent more time on the article than a predefined threshold. In our implementation, we have chosen 10 seconds as a minimum time period for users to read the title and get an impression of the article content. More advanced approaches are possible using the reading time and article length, but these are not always reliable in a mobile environment.

Since our system uses implicit feedback based on users' selections (see Section 5), the profile update process is a simple summation of the item vectors of different articles. Articles from the past are considered as less representative for the user's preferences than recent articles. Therefore, the value of a term decreases exponentially as the age (in hours) of the article increases, meaning that older items will contribute less to the profile. Although these terms with their corresponding interest values may form a rather long profile vector, and as a result a long search query, Lucene is designed to handle such search requests in a very short time. Therefore, recommendations are requested when needed and hence always up-to-date.

News events with a high impact (e.g., a natural disaster in a remote part of the world) have to be detected and considered as a recommendation, even if the topic does not completely match the user's interests. These *trending topics* can be identified based on their frequency of occurrence. If the current frequency of occurrence is significantly higher than the frequency of occurrence in the past, the topic is considered as trending. Besides, trending topics are discovered by checking trends on Google's search queries [11]. Every hour, Google publishes a short list with trending searches. A special Spout was implemented to fetch these trending topics hourly. Trending topics are used to create a query for the search engine, and the resulting news items are added to the user's recommendation list. A final source of trending topics is Twitter. Research has shown that Twitter messages are a good reflection of topical news [18]. Therefore, another Spout was assigned specifically to query tweets regarding news topics using the Twitter API. Twitter accounts of specialized news services and newspapers were followed. The tweets originating from these accounts are focusing on recent news and characterized by a high quality. Retweets and Favorites give an indication of the popularity and impact of a tweet. Subsequently, Tweets are processed in the same manner as other news items by Bolts.

As stated in the introduction, straightforward collaborative filtering is not usable for news recommendations because of the new item problem. Unfortunately, content-based recommendations are typically characterized by a low serendipity; recommendations are too obvious. To introduce serendipity, a hybrid approach was taken by adding a collaborative filtering aspect to the content based recommender. A traditional nearest neighbor approach was used to calculate similarities between user-user pairs. Instead of recommending the items that the neighbors have consumed, our implementation will recommend profile terms that are prominent in neighboring profiles. These profile terms of the neighbors are used to extend the profile of the user, thereby making it more diverse. Subsequently, this extended profile is used to generate content-based recommendations using the search engine. By extending the profile of a user with terms that are significant in the profiles of the user's neighbors, profiles are broadened and diversified with related terms. These extended profiles will produce more diverse recommendations covering a broad range of topics. Since the additional profile terms are originating from neighbors' profiles, the added terms will probably be in the area of interest of the user. The collaborative filtering component is based on the implementation of Apache Mahout [25]. Mahout ensures the scalability of this component of the system. Moreover, the profile extension is not a time-critical component, and is therefore implemented as a batch process running periodically. Content-based recommendations are based on the current version of the user profile, and as soon as the profile extension is finished, the profile is updated. This ensures that real-time recommendations can be generated at all time.

Finally, also the publishing date of the article is taken into account in the recommendation process. In the current implementation, only news articles of the last two days are candidate recommendations. However, a more intelligent degradation over time, with a degradation rate depending on the category or content of the article, can be future work.

4.4 Clustering

In the fourth phase, the recommended news items are clustered into topics. Since the news items in our system originate from different content sources, multiple items may cover the same news story. To provide users a clear overview of the news without removing content items, items about the same topic are clustered together. To cluster the content, three clustering approaches are considered during the design.

1. A *periodic clustering* of the complete content library before generating recommendations. Traditional clustering algorithms, which assume that all items are known before the clustering starts, can be used to periodically cluster all news items [23]. This approach does not allow the recommendation process to begin before the complete clustering of the content library is finished. Since this disadvantage introduces too much delay when adding new content to the library, it was not an approach for our system.

2. An *incremental clustering* of the content library before generating recommendations. In this approach, new content items are assigned to the best matching cluster, or a new cluster is made in case there is no match. Although this clustering approach is used in different existing systems [15, 7], we did not opt for this approach because it is not personalized. For a large content library, a large number of clusters can be identified. Since the clustering process is performed before the recommendation process, the clusters are identical for all users. However, personal interests may require a personalized clustering of the news content.

3. A *clustering of the recommended content items.* This is the approach that is used in our system, using a hierarchical clustering algorithm. Content items are not

clustered until the recommendation process is finished. The advantage of this approach is that only a small set of content items (250 candidate recommendations in our system) have to be clustered. Another advantage of clustering the recommendation results is the personalized nature of this set. For each user, the clustering process will result in a different clustering. Even a different level of clustering (number of clusters) can be chosen for every user. Users who are very interested in sports may find different clusters for soccer, baseball, cycling, etc., whereas users who are moderately interested may receive only one sports cluster containing all sporting disciplines. On the downside, users may not be familiar with a personalized clustering. As user preferences change or as collaborative filtering is applied to extend profile vectors, clusters are not stable over time. This behavior may surprise users who first got used to the existing clusters and then cannot find their 'old favorite' clusters anymore.

5. USER INTERACTION

Mobile has become, especially amongst younger media consumers, the first gateway to most news events published online. In a recent survey [21], conducted in 10 countries with high Internet penetration, one-fifth of the users now claim that their mobile phone is the primary access point for news. The small screen and typical interaction methods of mobile devices (touch screen) induce extra challenges and possibilities for news services.

Because of this, we made our news service available as a web application that is usable on desktop but also on tablets and smartphones. Figure 2 shows a screenshot of the user interface of the (mobile) web application, based on HTML5 and Javascript. On the left hand side, an overview of the recommended content items is shown. For each article, the number indicates how many articles covering this topic are clustered together. Selecting one of the items in the left column will show the article content on the right hand side using an HTML iframe. HTML iframes are used in order to provide all functionality of the source website, such as hyperlinks, while providing users the ability to browse their recommendations using the left column. Parsing the content of the source and reproducing it inside our own application is a technically feasible alternative, but violates the terms of use of many websites. Redirecting the users to the source website (using hyperlinks) would imply that users leave our web application and continue their news consumption on the source website, thereby making it impossible to track their behavior. The user interface is adapted to mobile devices by providing a clearly readable overview of the content, and interaction through tapping and swiping the touch screen. For smaller screens, such as smartphones, the column on the left hand side can be hidden to show the news articles in full screen. Further optimizations for mobile devices and touch screens are provided by using JQuery Mobile [27].

Explicit feedback for news services is difficult to interpret and therefore less common. E.g., a 1-star on a 5 point rating scale can be interpreted as a disinterest for the content, or as sympathizing with a story about some tragic event. Therefore, our system is using implicit feedback based on the user's viewing behavior. If an article is selected and shown on the screen for at least 10 seconds, we assume that the user has some interest in the topic of the story .

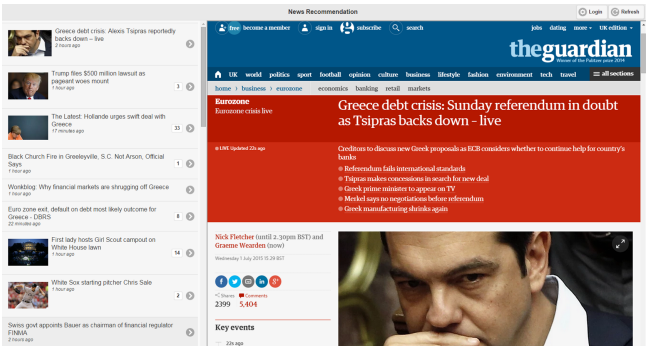

Figure 2: A screenshot of the user interface of the (mobile) web application.

Evaluating the system performance in terms of response time gave the following results. A mean response time of 800 ms was measured to generate 250 recommendations. This request includes retrieving the user profile and trending terms, executing the query on the search engine, and clustering the resulting items. These results were obtained on our test system, an Intel Xeon E5645 CPU at 2.40 GHz with 8GB of RAM running CentOS 6.6.

6. CONCLUSIONS

In this paper, we proposed a hybrid, real-time recommender system for news, combining technologies such as Storm, Lucene, and Mahout to ensure scalability and quick response times. Storm enables the processing of large streams of news content. Lucene provides the functionality of a search engine and is used as a content-based recommender. The collaborative filter of Mahout is used to exchange profile terms among neighboring users. User profile vectors are extended with related terms interesting to read about. The resulting hybrid recommendations are clustered according to their topic and presented to the user through a web application that is optimized for mobile devices. This research discussed the possibility of combining collaborative filtering and a search engine to compose a hybrid news recommender system, thereby combining the advantages of both. Search engines ensure a real-time response behavior while collaborative filtering adds community knowledge to the system. As future work, we consider to make a distinction between short-term interests and long-term interests of users. We also plan to focus more on entities mentioned in articles.

7. ACKNOWLEDGMENTS

We would like to thank Sam Leroux for the work he performed in the context of this research during his master thesis.

8. REFERENCES

[1] Apache Software Foundation. Apache storm, 2015. Available at `http://storm.apache.org/`.

[2] T. Bogers and A. van den Bosch. Comparing and evaluating information retrieval algorithms for news recommendation. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, pages 141–144, New York, NY, USA, 2007. ACM.

[3] T. Brodt and F. Hopfgartner. Shedding light on a living lab: The clef newsreel open recommendation platform. In *Proceedings of the 5th Information Interaction in Context Symposium*, IIiX '14, pages 223–226, New York, NY, USA, 2014. ACM.

[4] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, Dec. 1992.

[5] I. Cantador, A. Bellogín, and P. Castells. News@hand: A semantic web approach to recommending news. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 5149 of *Lecture Notes in Computer Science*, pages 279–283. Springer Berlin Heidelberg, 2008.

[6] D. Cutting and J. Pedersen. Optimization for dynamic inverted index maintenance. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '90, pages 405–411, New York, NY, USA, 1990. ACM.

[7] 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*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[8] T. De Pessemier, S. Coppens, K. Geebelen, C. Vleugels, S. Bannier, E. Mannens, K. Vanhecke, and L. Martens. Collaborative recommendations with content-based filters for cultural activities via a scalable event distribution platform. *Multimedia Tools and Applications*, 58(1):167–213, 2012.

[9] T. De Pessemier, C. Courtois, K. Vanhecke, K. Van Damme, L. Martens, and L. De Marez. A user-centric evaluation of context-aware recommendations for a mobile news service. *Multimedia Tools and Applications*, pages 1–29, 2015.

[10] Elastic. Elasticsearch, 2015. Available at `https://www.elastic.co/`.

[11] Google. Google Hourly Trends, 2015. Available at `http://www.google.com/trends/hottrends/atom/hourly`.

[12] E. Hatcher and O. Gospodnetic. Lucene in action (in action series). 2004.

[13] Katta. Lucune & more in the cloud, 2015. Available at `http://katta.sourceforge.net/`.

[14] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, Mar. 1997.

[15] L. Li, D. Wang, T. Li, D. Knox, and B. Padmanabhan. Scene: A scalable two-stage personalized news recommendation system. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 125–134, New York, NY, USA, 2011. ACM.

[16] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[17] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[18] 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*, RecSys '09, pages 385–388, New York, NY, USA, 2009. ACM.

[19] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. Recon: A reciprocal recommender for online dating. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 207–214, New York, NY, USA, 2010. ACM.

[20] M. F. Porter. Snowball: A language for stemming algorithms, 2001. Available at `http://snowball.tartarus.org/`.

[21] Reuters Institute for the Study of Journalism. Digital News Report, 2015. Available at `http://www.digitalnewsreport.org/`.

[22] A. Said, A. Bellogín, and A. de Vries. News recommendation in the wild: Cwi's recommendation algorithms in the NRS challenge. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge. NRS*, volume 13, 2013.

[23] K. G. Saranya and G. S. Sadhasivam. A personalized online news recommendation system. *International Journal of Computer Applications*, 57(18):6–14, November 2012.

[24] The Apache Software Foundation. Apache Lucene, 2015. Available at `https://lucene.apache.org/`.

[25] The Apache Software Foundation. Apache Mahout, 2015. Available at `http://mahout.apache.org/users/recommender/recommender-documentation.html`.

[26] The Apache Software Foundation. Apache Solr, 2015. Available at `http://lucene.apache.org/solr/`.

[27] The jQuery Foundation. jQuery mobile, a touch-optimized web framework, 2015. Available at `http://jquerymobile.com`.

[28] M. Woodman. Rome, 2015. Available at `https://rometools.jira.com/wiki/display/ROME/Home`.