An Analysis of Recommender Algorithms for Online News

Doychin Doychev, Aonghus Lawlor, Rachael Rafter, and Barry Smyth

Insight Centre for Data Analytics,
University College Dublin Belfield, Dublin 4, Ireland.
{firstname.lastname}@insight-centre.org

Abstract. This paper presents the recommendation algorithms used by the Insight UCD team participating in the CLEF-NewsREEL 2014 online news recommendation challenge.

1 Introduction

Recommender systems have become an essential part of our day-to-day lives, when it comes to dealing with the overwhelming amount of information available, especially online. Recommender systems improve user experience and increase revenue in the context of online retail stores (Amazon, eBay), online news providers (Google News, BBC) and many more. In this work we present a wide range of online recommender algorithms and compare their performance in the scope of the CLEF-NewsREEL 2014 online challenge.

In CLEF-NewsREEL 2014, participating teams connect to the Plista Open Recommendation Platform [1, 2] and have to respond to real-time user requests for recommendations. Given that recommender systems have traditionally been evaluated offline, this poses an interesting challenge in terms of algorithm efficiency, tracking users, building quality models of user browsing behaviours and preferences, and in terms of dealing with a highly dynamic domain like news in which there is a constant cycle of new articles appearing and older articles becomming redundant. We consider the challenges of online news recommendation more fully in Section 3.1.

The rest of this article is organised as follows: in Section 2 we present some related research in news recommendation; in Section 3 we provide a more detailed description of the Plista ORP framework[3] and the CLEF-NewsREEL challenge, the challenges of online news recommendation, and our system architecture; in Section 4 we describe our recommender algorithms; in Section 5 we report and analyse the data collected and the performance of our recommender algorithms, and in Section 6 we conclude and discuss directions for future research.

2 Background

Within the field of recommender systems, the problem of recommending news articles to readers has a number of unique and interesting features. In many traditional application domains of recommender systems a user profile is built from a users transaction or preference history, for example movies that have been rated or products purchased. The profile is associated with the user as they interact with the site, and feeds into the creation of personalised recommendations. In the news domain it is generally not common to have detailed user profiles, as users are not often required to sign in or create profiles. It is also uncommon in the news domain for users to explicitly provide feedback on each news article they read and often the only feedback available is implicit in the logs of the users click patterns. This presents a particular challenge for collaborative filtering methods which rely on the opinions of similar users to generate recommendations [4–6].

Further complications for collaborative filtering arise from the dynamic nature of the users and the news articles themselves [7, 8]. In general, users will prefer fresher news articles, and building an accurate model of user preferences based on the items they have previously read can be difficult [7, 9, 10]. While users may have preferred categories of news articles, or topics they are particularly interested in, these preferences are difficult to learn [11, 12]. User preferences change over time too, and another challenge is to provide a diverse set of interesting recommendations, accounting for known users preferences and recency and popularity of the news articles themselves [13, 14].

Content based approaches can run into problems where some measures of similarity identify news articles which are in fact about different topics. Extracting the constantly changing distribution of topics in the news presents a challenge [9, 15] in addition to learning how users choices are influenced by these latent factors [6, 16, 17].

3 Methodology

Plista provides the ORP platform for making live recommendations to a number of their client sites. Plista communicates with the teams via an HTTP API, by sending (JSON) messages. These messages are triggered when a user reads an article or clicks on a recommendation (*event notification*), and whenever an article is created or updated by the publisher (*item update*). Requests for article recommendations (*recommendation request*) are sent as message to the teams, to which a response must be given within a certain timeframe (100ms). The resulting dataset is unique in many respects, providing detailed user profile information for some users where available and cross-domain data from 11 different news providers. The dataset is fully described in [1].

With each message type (request, event, update) the Plista ORP framework provides additional metadata regarding the current user and article. Although the Plista ORP API documentation lists almost 60 fields of information, in practice we found many were unclear, or not useful or detailed enough to use. Fields like *Filter_allowosr* typify the ones we simply didn't understand (they had no description either), and popular demographic signals like age, gender and income, expressed as probabilities (male vs female, age and income brackets) turned out to be too vague to be of use in reality. In the end, we settled on 8 metadata fields for use in our recommenders:

geo_user The geolocation of the user
time_weekday The day of the week
category The subcategory under which an article is published within a given domain, e.g. sport, world, local etc.
time_hour The hour of the day
item_source The unique item (article) identifier
publisher The publisher (domain) of an article.
keyword Keywords with their corresponding counts occurring in the article
user_cookie The user id, which is not necessarily unique

Note that the *category*, *publisher* and *keyword* fields only provide a numerical id rather than a textual representation. For *category* and *publisher* it was possible to exploit URLs in order to determine this information, but for keywords there was no way to uncover what the words actually were, or how they were chosen. Nonetheless, we found that despite this the keywords still provided a useful signal. In Section 5 we provide a deeper analysis of the data provided and used.

3.1 Challenges

In this section we present the challenges of producing online news recommendations. In the section that follows, we detail the system architecture that we have implemented to cope with these challenges.

Traditionally, recommender systems are evaluated offline, with plenty of time to build complex models of user browsing behaviours and preferences. In real-time online scenarios like CLEF-NewsREEL however, such leisure is not afforded; not only do teams have to respond within a very tight timeframe (100ms), they also have to deal with factors like the lack of rich user browsing history as users are not obliged to register and therefore do not always have persistent profiles or identifiers. Moreover, a user can access the news sites from different devices, and many users can do so from the same shared device, further complicating the ability to reliably track their browsing, as pointed out in [1]. Tracking user preferences is also non-trivial, as users do not provide any explicit feedback about recommendation quality, and clickstream data is the only signal of relevance available. Finally, the nature of the news domain itself throws its own set of challenges into the mix due to the dynamic nature of the data, where many new articles appear every hour, and older articles quickly become redundant. In such unstructured and dynamic environments, it is necessary to apply techniques that satisfy requirements such as response time and scalability while improving the user experience using limited and sometimes noisy information.

3.2 System Architecture

Our system architecture, shown in Fig 1, is implemented in Python and is designed to accomplish our goals of scalability, speed and extensibility. The first

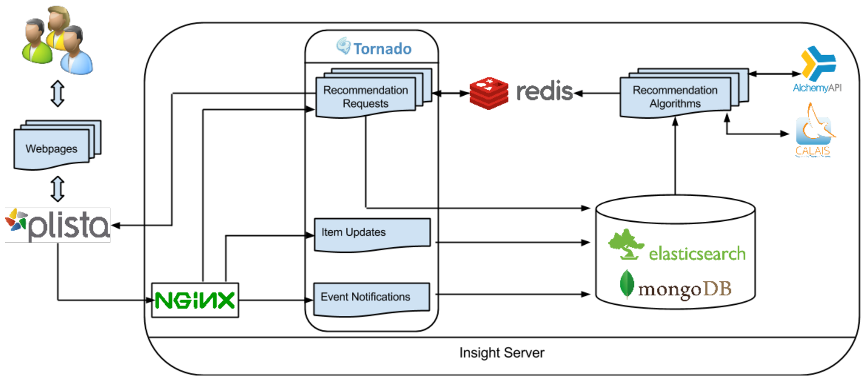


Fig. 1: System Architecture

point of interaction with Plista is NGINX, a load balancing HTTP server, which filters and redirects the different types of Plista messages to individual python processes (using the Tornado framework, one for each different types of message). All Tornado servers write to a permanent archive database, for which we use MongoDB. In order to make content-based recommendations we use an ElasticSearch instance, which keeps only data that is relevant to *recommendable* news articles, (articles often expire after a few days and get flagged as unavailable for recommendation).

Rather than trying to guarantee accurate recommendations in under 100ms, we precompute the recommendations and store them in a Redis instance. For each of our recommendation algorithms, we have a long running process which continually reads the latest events and articles from the database to build its recommendations. With this offline approach, there is a danger that we might send back a recommendation for an article which has expired during the time it took to compute the recommendations. To deal with this possibility, we refresh our recommendations in the background as frequently as possible so that our recommendations have a minimal lag. A typical refresh time for the most demanding recommenders (usually the content-based ones) is less than 5 minutes. We have constructed the system with a goal of compromising between accuracy or freshness and scalability. Our architecture is capable of sending recommendations back to Plista with a typical response time of about 3ms. We can easily scale it to handle many different algorithms for testing and evaluation purposes.

4 Recommendation Algorithms

In this section we describe our recommender algorithms. For each we describe the algorithm by how it selects the *candidate set* i.e. the larger set of items that will be considered for recommendation, and its *ranking* strategy, i.e. how it ranks the candidate items for recommendation, before returning the *top-N* to the user. We refer to the *target user* as the user for whom the recommendations are required, and the *current article* as the news article the user is currently reading. We have implemented sixteen recommenders in total: six popularity-based recommender, seven content-based recommenders, a feedback-based recommender, a recency-based recommender and an ensemble recommender.

Before returning the *top-N* recommendations to the user, we apply three basic filters to the candidate set:

Exclude items from other domains Recommendations must come from the same domain as the current article; we do not consider cross-domain recommendations although this is something we would like to investigate in the future, once we have an understanding of how single-domain recommenders work in this space.

Exclude already read articles We do not make recommendations of articles we know the target user has already read, however we do allow previously recommended articles to be re-recommended if the user did not click on the recommendation the first time. As always, it's hard to interpret non-clicks in recommender systems; a more mature model might limit the number of times the same recommendation is ultimately shown to a user.

Exclude non-recommendable items These are items that are flagged as non-recommendable (usually older articles) by Plista.

Each of the six popularity recommenders rank their candidate set by item popularity, i.e. the number of users who have read an article. However, they differ in the way they compile their candidate sets. We have developed a basic as well as more advanced recommenders that use additional features. They can be described as follows:

Popularity - Basic The candidate set is all items in the dataset.

Popularity - Geolocation The candidate set is all items that have been read by users in the same geographical location as the target user.

Popularity - Item Categories Every item is associated with zero or more news categories (business, sports, politics, etc.). The candidate set is all items whose categories intersect with the target article's categories.

Popularity - Weekday The candidate set is all items that have been seen at least once in the same week day as the target article.

Popularity - Hour The candidate set is all items that have been seen at least once in the same hour as the target article.

Popularity - Freshness The candidate set is all articles that have been published or updated in the last 24 hours.

Similar to the Lucene-based recommender described in [2], the group of content-based recommenders recommend articles that are similar to the current article. The intention here is not to present the user with an article essentially the same as the current one, something which would of course be undesirable, but rather to find articles that are strongly related to the current article. We believe that in the case of news stories that evolve over a period of days or even weeks such as the recent disappearance of the Malaysia Airlines flight MH370, such recommendations would be especially desirable, as new facets of the story unfold and change.

Each of the content-based recommenders we deploy use the conventional TF-IDF vector space model[18] to derive the candidate set. However, each algorithm uses a different selection of content within an article over which to apply the model. For each of the following content-based recommenders, the candidate set is always ranked in decreasing order of similarity to the current article.

Content - Title and Summary The candidate set is all articles, represented by the terms in their titles and summaries.

Content - Title and Summary + Freshness As above, but candidate set articles must also be fresh i.e. published or updated in the last 24 hours.

Content - Title and Summary + New As above but candidate set articles must be brand new i.e. published in the last hour, rather than fresh.

Content - Keywords The candidate set is all articles, represented by their keywords provided by Plista (recall that Plista provides a set of keywords and their frequencies within a document, although how the keywords are selected, or what they actually are is not disclosed).

Content - German Entities The candidate set is all articles, represented by their entities; using AlchemyAPI we extract entities, in German, from the full text of the articles.

Content - English Entities The candidate set is all articles, represented by their entities. This time, we use Google Translate to first translate the articles into English and then use OpenCalais to extract entities from the full text of the articles.

Content - English Entities in Context Expanding on the previous recommender, we represent articles using both their entities and the context surrounding them in the article. In OpenCalais context represents the relevant text before and after an entity.

Positive Implicit Feedback The candidate set is all articles that have been successfully recommended in the past (by any team's algorithm) to some user, i.e. clicked by the user. The more popular an article is as a recommendation, i.e. the more clicks it has, the higher the algorithm ranks it.

Most Recent The candidate set is all articles in the dataset. Candidate articles are then ranked in reverse chronological order of when they were published or updated.

Ensemble Recommendations are produced based on a combination of all the popularity- and content-based recommenders above. The candidate set is the union of candidate sets from each recommender. Candidate articles are then ranked using the sum of the rankings for the top n articles from each recommender. If, for any recommender, an article does not occur in the top n recommendations, it receives the maximum ranking of n, plus a penalty of 1, as in equation 1. We set $n = 100$ in these experiments.

$$rank(a) = \sum_{r \in recommenders} \begin{cases} rank_r(a) & \text{if a} \in \text{recommendations(r)} \\ n+1 & \text{otherwise} \end{cases} \tag{1}$$

5 Analysis and Evaluation

In this section we give a brief overview of the data we receive from Plista, in order to better understand the performance of our recommender algorithms. We will finish with a discussion on possible evaluation methods.

5.1 Dataset Overview

To illustrate the nature and dynamic quality of the data and how it informs our choice of recommender algorithms, we look closely at a one week sample, between Wed, 25 May '14 and Tue, 30 Jun '14. During this time period we observe 11 websites. They range from domain-specific news e.g. sport, (*sport1.de*) to more general news (*tagesspiegel.de*) to online home and gardening stores (*wohnen-und-garten.de*). There were approximately 8 million visits from 1.75 million unique users to 400 thousand unique articles published. The mean number of observed articles per user is approximately 4, and the data sparsity is 99.99%.

Over 90% of the traffic generated cumulatively from the websites is *event notifications* (impressions and clicks). This traffic is unevenly distributed across the domains, with *sport1.de* contributing almost 40% and together with *ksta.de* and *channelpartner.de* over 80% of all traffic. We find that most of the users that visit *tagesspiegel.de* also visit most of the rest of the websites, indicating the broad appeal of this domain. Surprisingly enough, there is a noticeable overlap between users of *sport1.de* (sports website) and *wohnen-und-garten.de* (online home and gardening store). On the other hand, users who frequent *motor-talk.de* rarely visit any of other websites. The uneven patterns of cross-domain traffic are certainly interesting, but for simplicity we have restricted our algorithms to only make recommendations in the same top-level domain for this work.

5.2 Evaluation

We show the average and maximum CTR results as reported by Plista in Fig. 2, and we compare the CTR with our algorithms in Table 1. We also show the precision results from an offline evaluation we additionally performed, as we discuss later in this section. With the objective of testing many algorithms the final averaged CTR $< 1\%$ is not particularly competitive. We can also see that there is a noticable difference between the average and maximum CTR. This could be due to a number of reasons including network congestion, server overload and change in algorithm performance based on features such as the domain, time of the day, user geolocation, etc. In future work we plan to get a better understanding of the specifics of the problem and bring the average CTR closer to the maximum. In contrast, the scalability of our architecture is apparent in the high number of requests that we were able to respond to.

We show the distribution of article popularity in Fig. 3(a). Although, most articles have very low popularity among users, there are plenty with higher popularities suggesting that popularity-based recommenders should perform comparatively well and that is what we find - three of the top five of our recommenders are popularity-based.

Recommender	Avg CTR	Max CTR	Precision @10
Content-based (title and summary + freshness)	0.98	1.91	0.07
Content-based (German entities)	0.95	2.44	0.03
Popularity-based (weekday)	0.87	1.21	0.15
Popularity-based (user geolocation)	0.84	1.19	0.14
Popularity-based (item categories)	0.83	1.18	0.07
Content-based (English entities)	0.82	1.09	0.11
Popularity-based (basic)	0.79	1.04	0.20
Positive implicit feedback	0.78	1.17	0.84
Popularity-based (hour)	0.77	1.07	0.12
Content-based (title and summary)	0.76	1.01	0.13
Ensemble	0.73	2.00	0.18
Content-based (title and summary + new)	0.72	0.96	0.01
Content-based (English entities in context)	0.69	0.98	0.14
Popularity-based (freshness)	0.69	1.00	0.17
Most recent	0.57	0.76	0.08
Content-based (keywords)	0.523	0.92	0.15

Table 1: Average, max and precision of algorithms

The complex dynamics of the news cycle are apparent in the relative performances of our popularity recommendations based on different timeframes. We find better performance recommending articles that are popular over the course of a day compared to the course of an hour, indicating particular daily cycles in users' news consumption. It will be interesting to examine further the optimal timeframe over which recommendations should be made, and how this varies across each domain.

Our popularity recommendations based on geographic location of the user (see Fig. 3(b)), also performed well, and this highlights the importance of accounting for local and spatial contexts in news recommendations. Similar performance is found with category-based recommendations. Our intuitive understanding is that users tend to read news from the same category, and to further test this idea we looked at a transition matrix for user clicks from category to category in a single domain Fig. 4 (we chose the news domain *ksta.de*). We clearly see just a few dominant categories and clicks between these categories account for the vast majority of items that are read. However, it is clear that there are important patterns of activity between certain less popular categories and future recommender algorithms will attempt to exploit these.

The strong correlations in the click-transitions of a domain's categories suggests that content based recommenders are likely to have some success. We find high CTR for our content-based algorithms *Content - Title and Summary*, although the best performing variant is one which returns *fresh* recommendations. While users seem to express some degree of preference for articles that are similar, the behaviour is very different across domains and also changes over time.

Content-based recommenders have shown to be quite successful in the past. Our *Content - Title and Summary* recommenders perform among the best of our

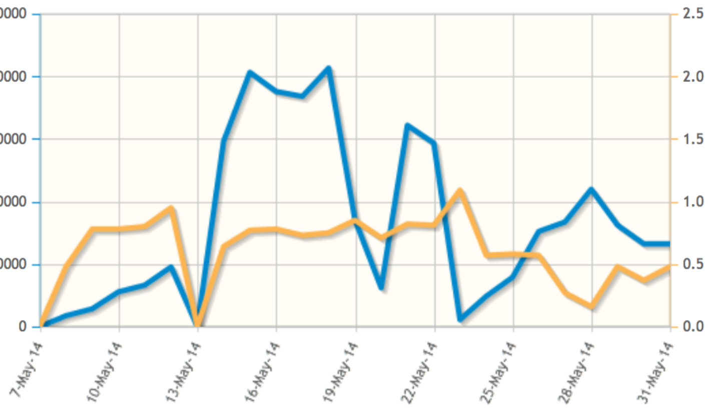


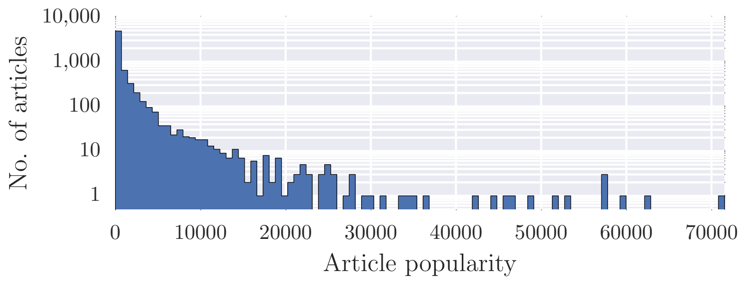
Fig. 2: Average CTR (orange/lighter colour) and number of requests (blue/darker colour). Due to problems with our server the number of requests are not is their expected flat line state.

algorithms. This shows that users enjoy articles that are related to the target article. The best performing variant is one which returns *fresh* recommendations.

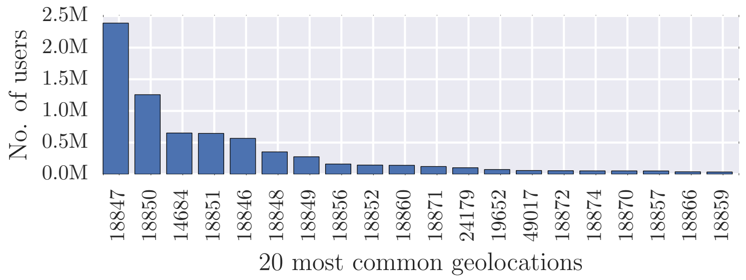
Alchemy and OpenCalais are entity extraction services which tag text and use linked data sources to identify entities in text such as people, companies, etc., and relationships between entities. While each service returns largely similar results, there are differences which can be significant (noticeable gaps in knowledge about non-English entities). The resulting recommendations achieve comparatively good scores, although it should be noted that for technical reasons we did not run all of our entity recommenders for the entire duration of the competition. Based on the maximum CTR they achieved (2.44%) we expect to look more closely at tuning these algorithms for further use.

The fact that the *Content - German Entities* outperforms *Content - English Entities* suggests that something is lost during the translation of articles into another language, even with entities which one might expect would be more reliably translated than full text. We also find that including contextual text that OpenCalais provides reduces recommendation accuracy. This is likely because this text firstly is not cleaned and contains stopwords, and secondly is likely to be more useful for identifying sentiment associated with entities, rather than more accurately classifying what the article is about.

The Plista competition provides a single evaluation metric - CTR. There are many possible approaches to evaluation [19], and in order to evaluate our own algorithms we constructed an 'offline' testing setup which allows us to 'rewind' the dataset to any point in time. We can then perform our recommendations against the data as if they were live and evaluate the results against the actual clicks that users are observed to perform. We use a simple average precision metric for evaluation, assuming that the order of the recommendations is not relevant. Our offline precision calculation allows us to shed more light on the crudely averaged



(a) Distribution of the popularity of articles. The most popular articles (the ones to the right on the x-axis) come from *sport1.de* and are about football.



(b) Histogram of number of users seen for the top 20 most common geolocations. We can see that there is a considerable amount of users concentrated at the big German cities, for example almost 2.5 million user could be located at Berlin and over 1.25 million at Munich.

Fig. 3

CTR provided by Plista, and provides us with many interesting suggestions for further work to improve our algorithmic techniques and evaluations.

6 Conclusions and Future Work

We have presented our results for the CLEF-NewsREEL 2014 Challenge. We have described our scalable architecture for delivering recommendations and detailed various algorithmic approaches that we have taken. We have highlighted some of the issues with the dataset which impact on the type of algorithms we can implement. For future work we intend to examine more closely the user features which are most relevant for collaborative filtering approaches. The dynamic click behaviour associated with browsing patterns is a rich area to exploit, and we also intend to improve on our content-based algorithms with better entity detection and similarity measures.

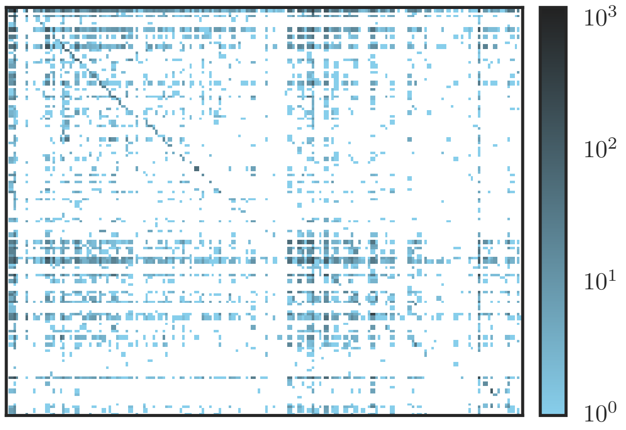


Fig. 4: Click transition matrix for categories on the *ksta.de* domain. We have excluded the category ids for which there are no click transitions to any other category. The colour scale represents the number of click transitions.

Acknowledgements

The Insight Centre for Data Analytics is supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289

References

1. Kille, B., Hopfgartner, F., Brodt, T., Heintz, T.: The plista dataset. In: Proceedings of the 2013 International News Recommender Systems Workshop and Challenge. NRS '13, New York, NY, USA, ACM (2013) 16–23
2. Said, A., Bellogín, A., de Vries, A.: 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 '13, New York, NY, USA, ACM (2013)
3. : The plista open recommendation framework. (http://orp.plista.com) Accessed: 2014-06-30.
4. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: Scalable online collaborative filtering. In: Proceedings of the 16th International Conference on World Wide Web. WWW '07, New York, NY, USA, ACM (2007) 271–280
5. Ilievski, I., Roy, S.: Personalized news recommendation based on implicit feedback. In: Proceedings of the 2013 International News Recommender Systems Workshop and Challenge. NRS '13, New York, NY, USA, ACM (2013) 10–15
6. Garcin, F., Zhou, K., Faltings, B., Schickel, V.: Personalized news recommendation based on collaborative filtering. In: Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01. WI-IAT '12, Washington, DC, USA, IEEE Computer Society (2012) 437–441

7. Fortuna, B., Fortuna, C., Mladenić, D.: Real-Time News Recommender System. Machine Learning and Knowledge Discovery in Databases **6323** (2010) 583–586
8. Agarwal, D., Chen, B.C., Elango, P., Wang, X.: Personalized click shaping through lagrangian duality for online recommendation. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '12, New York, NY, USA, ACM (2012) 485–494
9. Capelle, M., Frasincar, F., Moerland, M., Hogenboom, F.: Semantics-based news recommendation. In: Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics. WIMS '12, New York, NY, USA, ACM (2012)
10. Li, L., Wang, D.D., Zhu, S.Z., Li, T.: Personalized news recommendation: A review and an experimental investigation. Journal of Computer Science and Technology **26** (2011) 754–766
11. Li, L., Zheng, L., Yang, F., Li, T.: Modeling and broadening temporal user interest in personalized news recommendation. Expert Systems with Applications **41** (2014) 3168–3177
12. Li, L., Wang, D., Li, T., Knox, D., Padmanabhan, B.: 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, New York, NY, USA, ACM (2011) 125–134
13. Iaquinta, L., De Gemmis, M., Lops, P., Semeraro, G., Filannino, M., Molino, P.: Introducing serendipity in a content-based recommender system. In: Eighth International Conference on Hybrid Intelligent Systems, IEEE (2008) 168–173
14. Medo, M., Zhang, Y.C., Zhou, T.: Adaptive model for recommendation of news. EPL (Europhysics Letters) **88** (2009)
15. Lv, Y., Moon, T., Kolari, P., Zheng, Z., Wang, X., Chang, Y.: Learning to model relatedness for news recommendation. In: Proceedings of the 20th International Conference on World Wide Web. WWW '11, New York, NY, USA, ACM (2011) 57–66
16. Li, L., Li, T.: News recommendation via hypergraph learning: Encapsulation of user behavior and news content. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM '13, New York, NY, USA, ACM (2013) 305–314
17. Jancsary, J., Neubarth, F., Trost, H.: Towards context-aware personalization and a broad perspective on the semantics of news articles. In: Proceedings of the Fourth ACM Conference on Recommender Systems. RecSys '10, New York, NY, USA, ACM (2010) 289–292
18. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18** (1975) 613–620
19. Bogers, T., van den Bosch, A.: Comparing and evaluating information retrieval algorithms for news recommendation. In: Proceedings of the 2007 ACM Conference on Recommender Systems. RecSys '07, New York, NY, USA, ACM (2007) 141–144