News Personalization using Support Vector Machines

Anatole Gershman
anatoleg@cs.cmu.edu
www.cs.cmu.edu/~anatoleg

Travis Wolfe
tgw@andrew.cmu.edu

Eugene Fink
eugenefink@cmu.edu
www.cs.cmu.edu/~eugene

Jaime Carbonell
jgc@cs.cmu.edu
www.cs.cmu.edu/~jgc

Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, United States

ABSTRACT

We describe a system for recommending news articles, called NewsPer, which learns news-reading preferences of its users and suggests recently published articles that may be of interest to specific readers based on their interest profiles. The underlying algorithm is based on representing articles by bags of words and named entities, and applying support vector machines to this representation. We present this algorithm and give initial empirical results. We also discuss broader issues in the news personalization and the challenges of performance evaluation based on historical data.

Keywords

News recommendation, personalization, support vector machines, learning, natural language.

1. INTRODUCTION

The modern news sources provide a flood of information, and finding the most interesting and relevant news items is often a difficult problem. Traditionally, people have selected several preferred sources, such as a national newspaper, a local newspaper, a few specialty magazines, a radio station, and a TV station, and have relied on the editorial choices of these sources. This model of news consumption is being increasingly challenged in the online world by the emerging news aggregators, such as Google News, which provide an almost unlimited selection of news items and a variety of customization tools. Smartphones and tablets are becoming the platform of choice for news access.

Intuitively, readers want to see only relevant news items, presented in the order of their relevance. But what is "relevant" for one person may not be relevant for another; what is relevant today may not be relevant tomorrow. Hence we need a personalized model of what is currently relevant for each reader. In some ways, this challenge is analogous to building recommenders for consumer goods, such as Amazon products and Netflix movies; however, news personalization differs from product recommendation in the followings ways, which make traditional recommender algorithms inapplicable to this problem.

- Most people are unwilling to invest time into explicitly rating news items. The system has to infer the users' tastes by watching various aspects of their behavior, such as how long they kept a specific article open on their screen, whether they saved a link, posted it to Facebook, or e-mailed it to friends.

- Different news items may be closely related. For example, they may be about the same event, and they may or may not provide different details or offer different viewpoints. As another example, a more recent article about the same event may be just a repetition of the earlier article or may describe new developments. The recommender system has to account for these dependencies

- The value of news is time-sensitive and the system has to account for the tradeoffs between relevance and recency of an article.

The development of news recommenders gives rise to the following challenges, which are similar to the problems involved in building product recommendation systems and search engines.

- While we can usually get positive examples of user preferences, by identifying news articles read by the user, the reliable selection of negative examples is a harder problem. The fact that a reader has skipped a specific news item does *not* always mean that she has no interest in the related news. She may ignore an article because she has already read about a related event; because she presently has no time to read it; or because the article title provides all the information she needs.

- The data about reading top news often obscures information about more specific individual interests. While we often have enough learning examples related to the top news, we get far fewer examples in less common areas. For instance, we may not have enough data to learn that a specific user is interested in news on cosmology but not planetary astronomy.

- When developing a recommender, we need to test it on past logs of news-reading behavior before deploying a live version, which gives rise to the problem of predicting its live performance based on experiments with static data.

The recent growth of online news services has caused an increased interest in news recommendation techniques. Researchers have experimented with various strategies for identifying relevant news, such as customizable profiles, which enable users to specify their preferences manually [2]; content-based filtering, which involves learning news contents that interest specific users, such as for example keywords, preferred sources and news categories, and relevant semantic information [5, 11, 14]; and collaborative filtering, which involves identifying people with similar interests and using this similarity in news recommendations [4, 10]. Since every strategy has its limitations, researchers have also studied synergetic approaches, such as combining content-based and collaborative filtering [12], and allowing manual tuning of automatically learned profiles [1]. The reader may find a summary of recent work on news personalization and related key challenges in a review article by Pazzani and Billsus [13].

Although researchers have studied a variety of recommender techniques, to date, there are no widely used deployed news personalization systems. We have recently begun work on developing a new approach to news recommendation, based on application of support vector machines to building personalized content-based profiles. The main novelty claim of our system called NewsPer is the use of multiple profiles for each user.

2. ARTICLE REPRESENTATION AND SIMILARITY

A quantitative measure of similarity between news articles is a key concept in NewsPer, which helps identify similar articles

covering the same event. We measure the similarity on the scale from 0.0 (completely dissimilar) to 1.0 (near-identical). The definition of similarity is based on the *cosine similarity* between *bags of words* and *bags of named entities*.

For each article, we construct its *bag of words*, which is a list of words in the article. We apply a stemming procedure that merges words with common stems. Furthermore, we prune common words, such as prepositions and conjunctions, which are called *stop words*; as well as extremely rare words. We then compute the TF-IDF value of each word in each bag. We view the resulting bags of words as vectors in multidimensional space, where each word is a coordinate axis and the related TF-IDF value is a specific coordinate on that axis. We define the similarity between two articles u and v as the scalar product of the two respective vectors, $words(u)$ and $words(v)$, normalized to their length, which is called their *cosine similarity*.

Similarity threshold: The initial experiments show that the similarity of two articles about the same event, published within a day from each other, is usually between 0.5 and 0.8. We have currently set this threshold to 0.75, which means that the system views articles with higher-than-0.75 similarity as similar descriptions of the same event.

Named entities: We define an article's *bag of named entities* in the same way as the bag of words. We prune rare named entities, which are mentioned in less than a certain pre-set number of documents. Note that we do *not* prune common named entities.. We compute the TF-IDF values for the entities and prune the ones with the TF-IDF values below a certain threshold, specified by a pre-set parameter.

3. SVM LEARNING

The system learns the interests of a user based on her past reading history and constructs a classifier that predicts her interest in previously unseen articles. The intuitive idea is to recommend articles that are (1) similar to the user's past readings and (2) dissimilar to the articles that the user skipped in the past. We explain the selection of training examples, give a list of features used in training, and present the learning algorithm.

3.1 Training examples

When learning to recognize articles of interest for a specific user, the system selects positive and negative examples based on the user's past reading history. The procedure for selecting examples distinguishes among five types of articles in the user's reading log, which are defined based on two global time thresholds.

Time thresholds:

- The *read-time* parameter is the length of time sufficient for reading at least part of an article, such as the first paragraph; we have set it to 7 seconds.

- The *skim-time* parameter is the length of time that may be sufficient for skimming through the article without reading it; we have set it to 5 seconds.

Article types:

- *Read article:* The user opened this article and kept it open for at least *read-time*.

- *Skimmed article:* The user opened this article and kept it open for strictly less than *read-time* but at least *skim-time*.

- *Rejected articles:* The user opened this article and then closed it in strictly less than *skim-time*.

- *Scrolled article:* The user "scrolled over" this article; that is, saw its title in a news list but did not open it.

- *Unseen article:* The user has not seen this article; that is, its title has never been displayed in the user interface.

If an article falls into multiple categories, the earlier category in the above list takes precedence. For example, if the user opened an article for a short period (*rejected article*), later opened it for a long period (*read article*), and also scrolled over it on other occasions (*scrolled article*), then we view it as a *read article*.

Example selection: We currently use the following rules for identifying positive and negative learning examples.

- *Positive:* The *read articles* are positive examples.

- *Negative:* The *rejected and scrolled articles* are negative examples if they are not similar to any of the positive examples. The reason for using similarity is to avoid confusing the learning algorithms in situations when the user read an article and scrolled over other similar articles. The current system does not distinguish between the rejected and the scrolled articles. In the future, we may experiment with assigning different weights to these two categories.

- *Other:* The *skimmed and unseen articles* are not used in training, since we cannot determine whether the user liked or disliked them.

The training examples are selected from a specific training period. It can be as short as one session, that is, the time between the user's login and logout. Alternatively, it can span several sessions. We skip the short sessions during which the user read fewer than 5 articles because we believe that the user was under time pressure and the negative examples from the short sessions do not correctly reflect the user's dislikes.

3.2 Features

We use four types of features of a news article in training the system; that is, we represent an article as a vector of these features, which serves as input to the learning algorithm.

- The *popularity* is a measure of interest in an article among *all* users, defined as $log(1+n)$, where n is the number of people who read it before the user saw the headline for the first time.

- The *age* is a measure that becomes larger as an article gets older. Specifically, if t is the time since the article publication, measured in hours, then its age is $log(1+t)$.

- The *bag of words* is a vector of TF-IDF values for the words in an article.

- The *bag of named entities* is a vector of TF-IDF values for the named entities in an article.

We have experimented with various subsets of these feature types and have used the *L2* normal form of the selected features to represent an article.

3.3 Application of support vector machines

After selecting positive and negative examples for a user, the system constructs a classifier that represents the user's interests during the training period. Specifically, it applies a support vector machine (SVM) learner, which generates a hyperplane that separates positive and negative examples. We have experimented with two different SVM algorithms.

- SVM-Perf
(www.cs.cornell.edu/People/tj/svm_light/svm_perf.html) [8, 9]:

An open-source package for standard SVM learning, which constructs a hyperplane that divides positive and negative examples.

- **SVM-Rank**
(www.cs.cornell.edu/people/tj/svm_light/svm_rank.html) [7, 9]: An open-source package for a different version of SVM learning. It constructs a classifier that ranks previously unseen examples rather than just dividing them into positive and negative.

3.4 Recommendation list

For each news-reading session, the system applies the learned classifier to the candidate set of articles and assigns each article a relevance value. Using these values, the system constructs an ordered list of article clusters. The article with the highest score starts the first cluster, and all articles similar to it are added to the same cluster. The article with the highest score among the remaining articles starts the next cluster and so on. This ordered list of article clusters is the system's recommendation for the user.

4. EVALUATION METRICS

While the best way to evaluate a recommender is to conduct experiments with live users, the designers of prototype systems usually do not have access to a live news system. Even if they do, the managers usually require evaluation on historical data before integrating a recommender into a live system.

We developed two types of tests and two metrics that can be applied to historical data. In all cases, the unit of testing is a user session. We have the ground truth in the form of a log of the session and classify the articles as positive, negative, or other as described in Section 3.1. A recommendation cluster is considered relevant if any of its articles is similar to any of the positive articles (*rel* = 1); otherwise, it is irrelevant (*rel* = 0). Articles published more than a day apart are always considered dissimilar. Note that during the experiments with historical data, we have no control over what was actually presented to the user. For the purpose of evaluation, we assume that the user would have opened any similar article if it were presented to him or her. Hence, we make no distinction between similar articles.

The two tests we used are called "seen" and "all". In the "seen" test, the candidate list for recommendations includes only the articles from the user session. In the "all" test, all articles published within 24 hours prior to the beginning of the user session are included. The two evaluation metrics are normalized Discounted Cumulative Gain (nDCG) [3, 6] and Mean Average Precision (MAP).

5. EXPERIMENTS

Available data: We used logs of a news-reading system with 37 thousand users and 1 thousand news sources (newspapers) over a period from June 14 to July 9 of 2010. The total number of news articles for that period available through the system is about 0.5 million. We have *not* had access to the live system and thus have run the experiments on past logs without live user feedback.

We randomly selected 100 frequent users who, on average, read between 10 and 50 articles per day. Most of them had 1 or 2 "valid" sessions per day, that is, sessions during which they read at least 5 articles. Typically, the number of positive, negative, and other articles in a session is between 30 and 100 each. The total number of articles published within 24 hours prior to a session is between 17,000 and 20,000.

We used the data from June 14 to July 4 for training and the rest for testing. We made and scored recommendations for each user and each session during the test period and then averaged the results for each user. For comparison, we created two baselines, called RAND and POP. For the RAND baseline, each candidate article received a random score. For POP, each article received the score equal to its popularity at the beginning of the user session, that is, $log(1+n)$, where n is the number of people who read the article before the session. As expected, POP performed much better than RAND. We then trained SVM-Perf on 50 positive and 50 negative examples from the sessions immediately preceding the testing period and applied it to the test session. The results were disappointing. While they were much better than RAND, they were worse than POP. We understood that people tend to read top stories, which explains why POP is a fairly good predictor, but we thought that the addition of the content-sensitive features should improve its performance. We tried various feature subsets, various parameter settings, larger training samples, and the use of SVM-Rank instead of SVM-Perf, but nothing could outperform POP.

At this point, we decided to revise the recommendation procedure. First, since popularity is such a strong predictor by itself, its influence is probably getting diluted by mixing it with less predictive features. We decided to take it out of the feature set and score articles separately for content relevance and for popularity. Second, we noticed that users were showing different interests at different sessions. For example, one day they were reading about the oil spill in the Gulf of Mexico and skipping other news; the next day they were focusing on Iraq. The stories on the same subject from two different days were sufficiently different to get into the training set as both positive and negative samples, thus reducing the effectiveness of learning. To address this issue, we used a separate user model for each session, based only on the content features. This model is a hyperplane created by training an SVM on that session.

When scoring a new article, we need to integrate all user models from the previous sessions and the popularity of the article. To accomplish it, we introduce two parameters, denoted α and γ, the values of which are between 0.0 and 1.0. The first parameter determines the relative weight of popularity and content scoring, and the second controls the integration of the prior user models. The content relevance *CR* of an article *A* is

$$CR = r_1 + \gamma \cdot (r_2 + \gamma \cdot (r_3 + \ldots)) = r_1 + \gamma \cdot r_2 + \gamma^2 \cdot r_3 + \ldots,$$

where r_1 is the content relevance of the article computed using the user model from the session immediately preceding the current session, r_2 is the content relevance computed from the session before that, and so on. Intuitively, γ is the discount factor for old sessions. The total relevance *R* of the article is the weighted sum of its content relevance *CR* and its popularity *P*:

$$R = \alpha \cdot CR + (1 - \alpha) \cdot P.$$

We used hill climbing to find the optimal values for α and γ for each training session. These values vary from session to session, ranging from 0.3 to 0.7 for α and from 0.7 to 1.0 for γ. When we use the model to score the articles for a test session, we take the averages of these parameters over the preceding two sessions.

When we use this approach, the recommender outperforms the POP benchmark. We summarize the results in Tables 1 and 2. We used the paired difference *z*-test to determine the *p* value. The results for SVM-Rank are not significantly different from the SVM-Perf results.

Table 1. The results of the "all" test for 100 users.

	MAP		nDCG	
	avg	sigma	avg	sigma
RAND	0.099	0.043	0.101	0.027
POP	0.189	0.066	0.245	0.084
PERF	0.266	0.107	0.270	0.097
PERF vs. POP	p value is 10^{-17}		p value is 0.003	

Table 2. The results of the "seen" test for 100 users.

	MAP		nDCG	
	avg	sigma	avg	sigma
RAND	0.485	0.089	0.707	0.070
POP	0.660	0.105	0.834	0.078
PERF	0.663	0.084	0.835	0.062
PERF vs. POP	p value is 0.3		p value is 0.5	

Discussion: The initial results look encouraging. The developed algorithms have performed much better than the random sorting. The popularity of articles plays an important role, which is understandable, since most people tend to read the same front-page news about the recent events. It is surprisingly hard to beat POP. The improvement over POP is significant only in the "all" test, in which we scored all recent articles, not just the ones with headlines seen by the user. Arguably, for our dataset, it is a better test then the "seen" test for the following reason. The available data is from users browsing the online versions of the actual newspapers, which preserve the original paper layout. It is likely that the users select the papers and their sections based on their content preferences, which increases the proportion of articles with relevant content in the "seen" set. This hypothesis would explain the statistical equivalence of the results in the "seen" test.

The available dataset is based on a short time period, which is insufficient for detecting more subtle but stable long-term user interests. A reader may be interested in a dozen of topics, such as astronomy or reptiles, but not enough to read about them every week. However, collectively these topics may account for a significant portion of the user's reading. It is hard to detect such preferences over a few weeks. Short-term interests, on the other hand, are much more volatile, changing from session to session. We believe that is the reason why the original approach of training an SVM over a longer period of time did not work as well as the session-based approach.

6. CONCLUSION

We have developed a system for recommending news articles based on previously observed reading habits. The initial results suggest that the system is effective in recommending relevant items. While its recommendations are imperfect, it makes appropriate suggestions most of the time. We believe that using it is better than manually filtering a flood of news.

The key limitation of the initial experiments is that they are based on past data. We plan to run live tests with real-time user feedback as part of the future work. The other direction of the future research is to develop collaborative filtering techniques, which will identify groups of users with similar tastes and account for this similarity in recommending articles. We aim to integrate the collaborative filtering with the current approach.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems: Help or Harm? In Proceedings of the Sixteenth International Conference on World Wide Web, pages 11–20, 2007.

[2] Krishna Bharat, Tomonari Kamba, and Michael Albers. Personalized interactive news on the web. Multimedia Systems, 6(5), pages 349–358, 1998.

[3] Bruce Croft, Donald Metzler, and Trevor Strohman. Search Engines: Information Retrieval in Practice. Addison Wesley, 2009.

[4] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In Proceedings of the Sixteenth International Conference on World Wide Web, pages 271–280, 2007.

[5] Norman Haas, Ruud Bolle, Nevenka Dimitrova, Angel Janevski, and John Zimmerman. Personalized news through content augmentation and profiling. In Proceedings of the IEEE International Conference on Image Processing, pages 9–12, 2002.

[6] Kalervo Jarvelin and Janna Kekalainen. Cumulative gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20(4), pages 422–466, 2002.

[7] Thorsten Joachims. Optimizing search engines using clickthrough data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, 2002.

[8] Thorsten Joachims. A support vector method for multivariate performance measures. In Proceedings of the International Conference on Machine Learning, 2005.

[9] Thorsten Joachims. Training linear SVMs in linear time. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, 2006.

[10] Joseph A. Konstan, Bradley N. Miller, David A. Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Reidl. GroupLens: Applying collaborative filtering to Usenet news. Communications of the ACM, 40(3), pages 77–87, 1997.

[11] Bernardo Magnini and Carlo Strapparava. User modeling for news web sites with word sense based techniques. User Modeling and User Adapted Interaction, 14, pages 239–257, 2004.

[12] Geogrios Paliouras, Mouzakidis Alexandros, Christos Ntoutsis, Angelos Alexopoulos, and Christos Skourlas. PNS: Personalized multi-source news delivery. In Proceedings of the Tenth International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II, pages 1152–1160, 2006.

[13] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred,Kobsa, and Wolfgang Neidl, editors, The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin, Germany, 2007.

[14] Hidekazu Sakagami and Tomonari Kamba. Learning personal preferences on online newpaper articles from user behavior. In Proceedings of the Sixth International World Wide Web Conference, pages 291–300, 1997.