Content Bookmarking and Recommendation

Ananth Vyasarayamut[1] *Satyabrata Behera*[1]
Manideep Attanti[1] *Ganesh Ramakrishnan*[1]
(1) IIT Bombay, India
ananthv@iitb.ac.in, satty@cse.iitb.ac.in, manideep@cse.iitb.ac.in,
ganesh@cse.iitb.ac.in

ABSTRACT
Personalized services are increasingly becoming popular in the Internet. This work proposes a way of generating personalized content and simultaneously recommending users, web pages that, he/she might be interested in, based on his/her personalized content. In this work, we portray a system that not only helps the user in bookmarking the URL and snippets from the web page but also recommends web pages relevant to his/her interest. It applies a content-based filtering approach. We describe the details of the approach and implementation as well as address the challenges associated with it.

KEYWORDS: Bookmarking, Recommendation System, PEBL, Clustering.

Proceedings of the Workshop on Speech and Language Processing Tools in Education, pages 69–78,
COLING 2012, Mumbai, December 2012.

1 Introduction

Bookmarking systems allow us to store URLs that we tend to visit often and URLs that had information which were hard to find. As far as we know, there aren't systems which not only store URLs but also store the part of the web page that the user found relevant. For this reason, we need systems which can store this along with the URLs. Still, how do we facilitate others from finding similar content without going through the same hassle? This is where a recommendation system comes into play. This is an attractive feature as it not only gives a greater focus on the content being the principal bookmarked data but also generates recommendations from other users with a similar interest. As an example consider someone buying a telescope. Let's say A is interested in a reasonable aperture telescope but doesn't want it to be too sluggish and has bookmarked a page talking about the Maksutov-Cassegrain telescope. But A might not be aware of the Schmidt-Cassegrain cadioptric scopes which are much lighter and offering a higher aperture too. This could be recommended from a user B who has already made this research and has come across this and has since bookmarked it.
A system was developed using the above approach. This system can be used by a community of people which not only facilitates the recommendation of new web pages to read but also stores the previously read web pages along with the selected text (important parts of web pages) and tags/labels related to the web page entered by the user. The complete approach will be described in the next few sections.

2 Related Work

Recommendation Systems[1] **:** The problem of recommending items has been studied extensively, and two main paradigms have emerged. Content-based recommendation systems try to recommend items, similar to those the user has liked in the past, whereas systems designed according to the collaborative recommendation paradigm identify users whose preferences are similar to those and recommend from those repositories.

For example, a content-based component of the Fab system (Balabanović and Shoham, 1997), which recommends Web pages to users, represents Web page content with the 100 most important words. Similarly, the Syskill & Webert system (Pazzani and Billsus, 1997) represents documents with the 128 most informative words. The "importance" (or "informativeness") of a word in a document is determined with some weighting measure that can be defined in several different ways.
As stated earlier, content-based systems recommend items similar to those that a user has liked in the past. (Lang et al., 1995), (Mooney and Roy, 2000), (Pazzani and Billsus, 1997). In particular, various candidate items are compared with items previously rated by the user and the best-matching item(s) are recommended. They create an user profile for each user depending on the items the user preferred. These profiles are obtained by analyzing the content of the items previously seen and rated by the user and are usually constructed using keyword analysis techniques from information retrieval. All these systems require feedback in the form of ratings from the user. Although our problem is similar to all these mentioned above, the scenario is different in that we do not have pages that the user does not prefer, as in negative instances. Here, only those web pages are bookmarked by the user which he likes. Also the user does not provide any ratings for the web pages. Since we only have web pages that user liked (positive training data), we have more content to learn user interest

[1] referred and extracted from (Adomavicius and Tuzhilin, 2005)

and the disadvantage here, is the unavailability of web pages that the user does not like (negative training data).

GroupLens (Konstan et al., 1997), (Resnick et al., 1994), Video Recommender (Hill et al., 1995), and Ringo (Shardanand and Maes, 1995) were the first systems to use collaborative filtering algorithms to automate prediction. Other examples of collaborative recommendation systems include the book recommendation system from Amazon.com, the PHOAKS system that helps people find relevant information on the WWW (Terveen et al., 1997), and the Jester system that recommends jokes (Goldberg et al., 2001). Since unlike content-based recommendation methods, collaborative recommendation systems (or collaborative filtering systems) try to predict the utility of items for a particular user based on the items previously rated by other users. These scenarios are not related to ours since we only want to recommend web pages related to a user interest and not content from users with similar activity. Also these collaborative systems also require user preference as ratings or likes/dislikes.

Learning from positive examples : Our problem scenario demands learning a model in the absence of negative training data. For this we explored One-Class SVM (Manevitz and Yousef, 2002), PEBL (Positive Example Based Learning for Web Page Classification Using SVM) (Yu et al., 2002) and found that PEBL performs much better than any of the approaches discussed in One-Class SVM. We discuss the implementation and the approach of PEBL in relevant sections.

3 Problem Description

Before stating the problem statement, let's define some terms:
Bookmark : Bookmark consists of two parts : The web page URL and selected text portion of that web page, tags or labels entered by the user reading it. Tags or labels are words that the user wants to store along with the selected text which may capture the basic idea of the web page content or any information relating to the web page.

Content Bookmarking and Recommendation System : This system refers to the process of bookmarking web pages that the user found interesting and is then bookmarked. Each user generates his bookmarks depending on the web pages he browses. Then depending on a user's bookmarks, he is recommended new web pages to read.

Problem Statement : Given a set of users and their bookmarks, the goal is to suggest each user bookmarks, from the bookmarks of other users in that closed community, related/relevant to his existing bookmarks. This can also be interpreted as suggesting a user new web pages to read related to his area of interest (which is indicated by his bookmarks).

This raises the need to create a model for each user capturing the pattern of his bookmarks which necessitates the storage of full text content of each bookmarked web page along with the selected text, tags/labels and URL.

The challenges that were faced and some of which were addressed are listed below:

- It needs to scale well with the addition of users and bookmarks.
- It needs to be fast.
- It should adapt to the change in user's interest indicated by the addition of bookmarks relating new topics.
- It should be time sensitive so that user is suggested more bookmarks related to his current set of interests.

4 Approach

The need to build a model for each user for capturing user interests can be seen in a scenario where a given user has a set of bookmarked web pages which indicates his interest and a set of bookmarked web pages of other users each of which may or may not belong to that given user's interest. This given user's data (i.e. bookmarks) are positive training examples and other user's bookmarks are unlabeled data since it is not known whether they are related to user's interest or not. This reduces to the problem of learning a model where only positive examples are available.
Previous literature was explored for the above and found PEBL(Yu et al., 2002) performs much better than other approaches. So this paper's approach has been implemented to solve our problem. Models are built for each user which captures their interests. These models are used for classifying whether a given web page is relevant to the user or not. Once all the web pages relevant to the user has been found, they are ranked to get the top 'k' web pages. These top 'k' web pages are clustered in order to present the related web pages in groups.
The approach of how the entire system was built is presented below in details :

4.1 PEBL: Positive Example Based Learning for Web Page Classification Using SVM

There is a need to learn a classifier for each user based on their bookmarks. These classifiers are used to classify bookmarks which are relevant to that particular user from the rest of the corpus. On the data, mapping-convergence (M-C) algorithm is run in training phase to build an accurate SVM from positive and unlabeled data. This paper uses SVM since it has properties like maximization of margin, nonlinear transformation of the input space to the feature space using kernel methods and tolerates the problem of high dimensions and sparse instance spaces. These properties makes SVM perform better in many classification domains. The M-C algorithm is as follows:

4.1.1 Mapping Stage

- Identify strong positive features that occurs in the positive training data (POS : bookmarks of a given user) more often than in the unlabeled data (bookmarks of other users).
- By using this list of the positive features, filter out every possible positive data point from the unlabeled data set, which leaves only strongly negative data ($M_1(neg)$). For instance, say a strong negative is a data point not having any of the positive features in the list.

- $S_1(pos)$ denotes unlabeled data points excluding $M_1(neg)$.

4.1.2 Convergence Stage

- This step trains the SVM repeatedly to aggregate mapped negatives ($M_i(neg)$) as close as possible to the unbiased negatives (the bookmarks not relevant to the given user) as per the given user.
- Now add $M_1(neg)$ to the NEG (NEG was empty before this). That is now $NEG = M_1(neg)$.
- Then construct a SVM from the positives (POS) and only the strong negatives $M_1(neg)$,i.e., NEG. The decision hyperplane between POS and NEG would be far from accurate due to the insufficient negative training data. This is shown in figure 1.
- Accumulate $M_2(neg)$ (data points in $S_1(pos)$ which are classified negative by the trained SVM) into NEG, that is, now $NEG = M_1(neg) \bigcup M_2(neg)$.
- Then retrain using POS and NEG.
- Iterate these processes until the $M_i(neg)$ becomes empty set, i.e., when no data in $S_{i-1}(pos)$ is classified as negative. See figure 2.
- The SVM constructed at the end of the process will be close to the SVM constructed from positive and unbiased negative data because NEG will converge into the unbiased negative data in the unlabeled data.

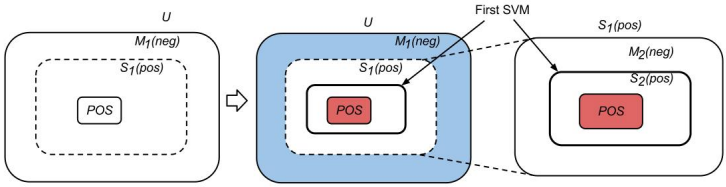

Figure 1: Training first SVM (from $M_1(neg)$ and POS that divides $S_1(pos)$ into $M_2(neg)$ and $S_2(pos)$). Taken from (Yu et al., 2002).

This process is applied to each user to create a SVM classifier for each of them. The SVM classifier of a given user is applied on the bookmarks of other users to find which are relevant to that particular user. The relevant bookmarks/documents are ranked to pick the top '*k*' which are highly ranked and fed to the clustering algorithm.

4.2 Ranking Algorithm

Word weights are calculated for each word in the vocabulary considering their occurrence in a given user's bookmarked pages. Then relevant documents for a user are given scores by aggregating weights of the words occurring in them normalized by the length of the

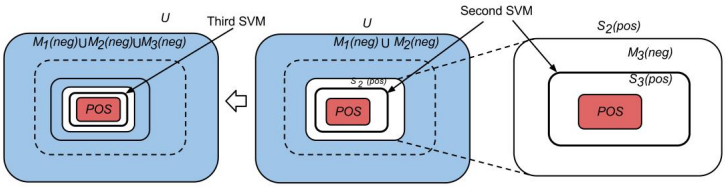

Figure 2: Training second SVM (from $M_1(neg) \bigcup M_2(neg)$and POS that divides $S_2(pos)$ into $M_3(neg)$ and $S_3(pos)$. Taken from (Yu et al., 2002).

document. Then relevant pages are ranked based on their scores and top 'k' documents are clustered and recommended to the user. All the relevant bookmarks and their scores are stored in the database.

4.3 Clustering Algorithm

Once the relevant bookmarks are found for a user, top k of them are clustered so that related bookmarks are presented as a group. The algorithm used is DBSCAN(Ester et al., 1998). This is chosen because it clusters all documents which are densely distributed in a region and ignores outliers. It doesn't have any constraints of how many clusters have to be formed and also inclusion of every object into some cluster. The choice of minimum similarity between the objects for them to be in a single cluster is obtained by a trial and error method. Also the selection of minimum number of objects around the selected object for it to be the cluster center is done by a trial and error method.

4.4 Feature Design

The bookmarks are preprocessed before determining features and creating feature vector. Pre-processing involves :

- Stop words removal
- Stemming

The stemmed words other than stop words are taken as features. Then each bookmarks is represented as a feature vector whose each column value is the number of times that the word occurred in the document plus a constant multiple of the times it occurs in selected text, tags/labels (constant multiplied to give higher weight to the selected text, tags/labels where this constant multiple is found by trial and error). All the feature vectors are stored in the database. Porter stemmer[2] is used for stemming the words.

4.5 Implementation Specifics

To make the system suitable for a online application, every activity like generating feature vector, training the classifier is done offline. To be clear and precise, when a user bookmarks

[2]http://tartarus.org/ martin/PorterStemmer/def.txt

a web page with selected text, tags/labels, then bookmarked content (web page with selected text, tags/labels) are stored in the database. After storing the bookmark, a module is called to create a feature vector corresponding to those bookmarks using the old features (words, except stop words, that occurred in previous bookmarks that were present when SVM was trained last time). Suppose there are words in the newly added bookmarks which is not there in features, they are kept track of separately. When the size of those new words (new features) crosses a threshold, then feature vectors for all bookmarks are created from scratch w.r.t the updated feature set. Also when a given user adds a bookmarks, that bookmark is tested against his classifier to keep track of change in user's interest. If that is classified as negative, then a counter is incremented to keep track of how many bookmarks added by user is classified wrongly by his classifier. If that number crosses a threshold or if the features set has changed, then retraining of the classifier is done for that user.
Also when a user adds a bookmark, that bookmark is tested against every other user's classifiers to know whether that is relevant to them. If it is, then that bookmark's score is calculated for each of those for whom it is relevant and added to the database table. Then top '*k*' are retrieved and clustered. The clustering information is kept and recommended to the user when logged in.

5 Case Studies

To test the system, user 'A' bookmarked content on Movies and Machine Learning while user 'B' bookmarked content on Cricket and Machine Learning. With a corpus of just over a hundred, the system was able to generate appropriate recommendations. In this case, both A and B got recommendation from web pages pertaining to machine learning. Since recommendations are subjective, a true measure of accuracy cannot be obtained in this system. However, we can 'judge' how well the system worked by getting a measure of the relevance of the documents suggested to the user's corpus. A snapshot of the recommended web pages to user B is shown in figure 3.
The system is able to provide good recommendations. Since all the processing is done

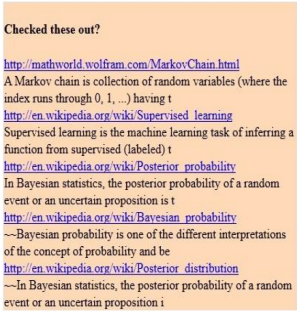

Figure 3: Top 5 relevant results which are recommended to user 'B'

offline (independently - as bookmarks are added to the system, recommendations are calculated and stored in the database table), when user logs in, the recommendations are fetched from the database table and displayed to the user. As described in the previous section, it also takes into account change in user interest.

6 Conclusion

The system designed here, for content bookmarking and recommendation can be used where the user's intention is to not only bookmark the URL but specific part of the web page that is relevant. Each bookmark can also be tagged, if necessary by the user. This system also generates recommendations based on the user's interest and is implemented in a generic manner so as to not adhere to any specific domain.
The usage of PEBL enables us to get a reasonably good accuracy and although our data does not entirely validate this, we still believe it is capable of further fine grained classification. In the future work, we expect to add the property of time sensitivity so that the current interests of the user are given more priority while providing recommendations.

References

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749.

Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

Ester, M., Kriegel, H., Sander, J., Wimmer, M., and Xu, X. (1998). Incremental clustering for mining in a data warehousing environment. In *Proceedings of the International Conference on Very Large Data Bases*, pages 323–333. Institute Of Electrical & Electronics Engineers.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.

Hill, W., Stead, L., Rosenstein, M., and 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*, pages 194–201. ACM Press/Addison-Wesley Publishing Co.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Lang, K. et al. (1995). News weeder: Learning to filter netnews. In *Proc. International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann Publishers, Inc.

Manevitz, L. and Yousef, M. (2002). One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154.

Mooney, R. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co.

Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. (1997). Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62.

Yu, H., Han, J., and Chang, K. (2002). Pebl: positive example based learning for web page classification using svm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248. ACM.