Twitter-based Products/Sales Events Recommendation System

Dongxue Liu, Qianyi Zhong, Chia-Jung Lin, Sung-Yen Liu
Electrical Engineering and Computer Science Department
Columbia University
e-mail: {dl2856, qz2198, cl3295, sl3763}@columbia.edu

Abstract—**User behaviors on social media reveal tremendous potential purchase demand as with the dramatic growth of social network. In this project, we built an application that can directly give recommendation to users if they express purchase demand on social network. Aside from that, to allow retailers make use of these opportunities, we also built a reciprocal platform for retailers and users of Twitter with web service. To obtain sentiment of tweets, we adopted map reduce pattern design for sentiment algorithm in the analysis of tweets; to accelerate recommendation process, we utilize map reduce pattern when training with dataset. These two applications adopted different recommendation algorithms but share the same online training set stored in RDS database provided by AWS.**

Keywords- Recommendation System; TF-IDF; MapReduce; Web Service

I. INTRODUCTION

Recommendation systems are widely used in retail industry to provide users with personalized purchase experience and thus generate profit for retailers in return. Meanwhile, social network provides a great opportunity for retailers to expand their commerce worldwide.

In this project, our work can be divided into two apps – a Twitter application that detects user's potential purchase demand and a web application that provides social network users and retailers (both online and local) a reciprocal platform. These two applications share the same online training set.

Different from traditional recommendation systems that mostly utilize graph data social media only, our Twitter application provides social media users with more consistent purchase experience. Once following the host account in Twitter, users would receive automatic respond from our host account if they post tweets with purchase demand. Users can also rate for our recommendation simply by replying us with a plain text version review (which would be automatically converted to rating using sentiment algorithm). Successful recommendation would be put into training table in database to it an online training set.

Aside from Twitter app, we also built an Internet application additionally to allow users and retailers interact with each other online. In this application, retailers can post detailed products/sales events information and get feedback from users. Meanwhile, users can also rate recommendation or search results within the platform.

II. RELATED WORKS

Recommendation systems are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item.[1][2] Recommendation systems are widely used in movies, music, news, books, restaurants and social networks. These system generate a list of recommendations in two ways – collaborative or content-based filtering.[3]

A. Collaborative Filtering

This approach is based on the collection and analysis of a large amount of information on users' behaviors, activities or preferences. The system would therefore generate prediction results based on user's similarity to other users. The assumption of collaborative filtering is that people who share the same taste would agree in the future as well, that is, they would like similar categories of products as they liked in the past. A key advantage of this approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items without analyzing of the items. To compute the similarity between users, there are many methods including k-nearest neighbor[4], Pearson Correlation and so on. Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity.[5] One of the most famous examples of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon.com's recommender system.[6]

B. Content-based Filtering

Different from collaborative filtering, Content-based filtering methods are based on a description of the item and a profile of the user's preference.[7] In the system, items are described by key words and users' preferences are indicated by their profile or their rate history, that is, system would generate recommendation list based on users' liked list. To extract features of items, a widely used algorithm is the TF-IDF representation. [8] Compared with collaborative filtering, content-based filtering does not require a large amount of information of users to make accurate recommendation in the beginning, that is, content-based

filtering needs less information to get started. However, the disadvantage of content-based filtering is that it highly depends on the original seeds (rated history) of users when making new recommendations. Thus, a key issue with content-based filtering is whether the system is able to learn user preferences from user's actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from recommendation system is significantly less than when other content types from other services can be recommended. [8]

C. *Hybrid Recommender System*

Observing the advantages of both approaches above, recent search demonstrated a hybrid recommendation algorithm that combines collaborative filtering and content-based filtering could be more effective in some cases. In general, hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combing them; by adding content-based capabilities to a collaborative-based approach; or by unifying the approaches into one model. [9] Hybrid algorithm not only provides more accurate prediction result, but also overcome some of the common problems in pure recommendation systems such as cold start.[8]

Based on the pros and cons of collaborative filtering and content-based filtering, we adopted content-based filtering in twitter application since the only content source of twitter app is Amazon; meanwhile, we adopted collaborative filtering in our Internet application given that there would be a variety of retailers providing diversified products/sales events.

III. System Overview

We have built two applications for the recommendation system: a twitter application and a web application. These two apps share the same backend logic and database. Both of the backend are implemented in Java. For database part, we use RDS provided by AWS.

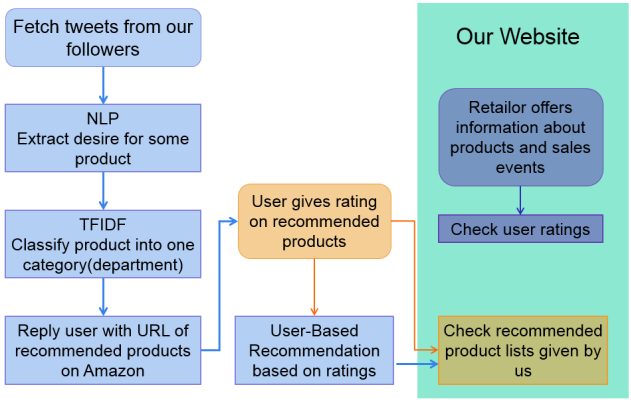

Figure 1 Flowchart of Recommendation System

A. *Twitter Application*

To respond to users who have purchase demand automatically, we maintain a host account in Twitter to send product information to users. When backend detects purchase demand from followers' tweets of the host account, system would match the keywords extracted from tweets with Amazon API by departments and rate history of this follower. After this step, system would respond to the follower with recommend item and URL. At the same time, we would store this product in RDS for reference of recommendation dependency later. An example of this step is shown in Figure 2.

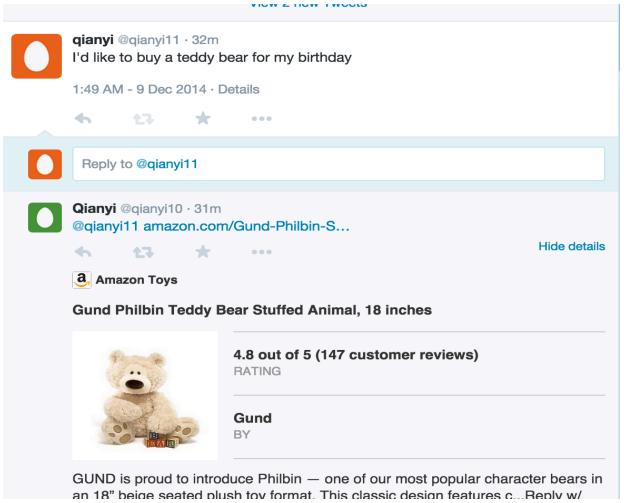

Figure 2 Automatically Respond

Users then can rate our recommendation by simply responding to our comment. Their response would be parsed into values (between 0 to 5) using sentiment algorithm to represent their preference. In addition, users' original tweets would be stored in our training set if the rate value were high enough, meaning that our recommendation is successful. An example of this step is shown in Figure 3.

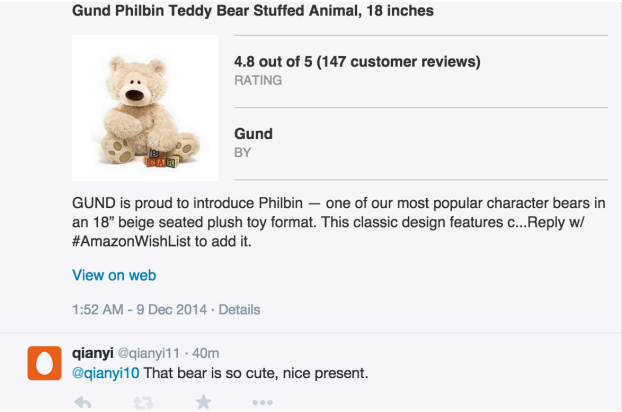

Figure 3 Collect Review of Recommendation

B. Web Application

There are three layers in our application including front-end, servlet and database. To enable retailers and users to interact with each other, we allow retailers to post products/sales events information. In the mean time, they can also manage their profile and post history on this platform. In the user main page, users can search products/sales events by keywords and give rate or comment; in addition, users would receive a recommendation list that is generated based on their rate history both in twitter app and web app.

C. Dataset

In this project, we use Amazon review dataset processed using Amazon API collected by J. McAuley and J. Leskovec as our recommendation source. The training data set (tweets) are collected using Twitter API.

a) Amazon API

Amazon offers Product Advertising API for bloggers to advertise products on Amazon through their webpages. We took advantage of one of its function that let us to retrieve product name and its webpage URL by sending out an http request with valid signature encoded. To process, we firstly assign product keyword, product category, public key and combine them into a query string; then we compute the HMAC of the canonical form of the query string; at last, we apply RFC 3986 mechanism to encode the HMAC and send out the http request. Amazon US area offers a variety of product category options as follows. We selected 5 categories (Beauty, Clothing & Accessories, Computers, Shoes and Toys & Games) to make recommendation.

Table 1: Categories of Amazon API

Category of Product	Category Names recognized in API
Books	Books
Movies, music, games	Music/Videogames/DVD
Electronic, computer	Electronics/PC Hardware
Tools	Tools
Home	Home
Beauty	Beauty
Health	Health Personal Care
Toys, kids, baby	Toys
Clothing	Apparel
Shoe	Shoes
Jewelry	Jewelry
Sports	Sporting Goods

Amazon server will return a JSON package that contains the first 10 matches of the given keyword and category. We parse the JSON file and extract the name and URL of the top three products.

b) Amazon Review Dataset

To extract features for each category, we use the amazon review dataset provide by SNAP. [10] We used 5 categories (Beauty, Clothing & Accessories, Computers, Shoes and Toys & Games) in all categories in the dataset. Parsed information using Python, we use TF-IDF to generate a word bag for every category. These word bags are used as criteria for classify tweets with purchase demand and contribute to the recommendation afterwards. We also use this data set as the resource of Amazon as a retailer in our web application.

IV. ALGORITHM

A. Category Classification

a) Bag-of-words

Given that category information should be provided for querying products, we have trained a category classifier, which return the highest related categories to retrieve more accuracy results.

To build this classifier, we utilized Amazon product review dataset[10], which contains the product IDs, names, descriptions and customer reviews. This dataset contains more than 2,000,000 products and information of products is separated into several text files. To extract meaningful information while avoid computational issue, we randomly choose a subset with about 100,000 items to represent our idea and algorithm of whole dataset.

After matching the category, product name and description by product ID, we collect all descriptions under same category (but different products) into a single file. Therefore, we are able to calculate TF-IDF of 15 category text files, which tell the discrimination of each word, to rank the relation between user queries and each category. The query format could be one word, free text or phrase, and the command line application would return the most related category that could be used for Amazon API.

b) Training and Classifying

The features that we use in training and classifying are TF-IDF Weighting (Term Frequency-Inverse Document Frequency Weighting) and TF-IDF Similarity of each essay to each class.

TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or

corpus. $n_{i,j}$ stands for the frequency of term t_i in document D_j, and $|D|$ is the word count of document D, which is divided by the term frequency and taken logarithms.

TF-IDF weighting of word t in tweet/document d can be calculated by equation (1).

$$w(t,d) = \frac{tf(t,d)\times log(N/n_t+0.01)}{\sum_t[f(t,d)\times log(N/n_t+0.01)]} \quad (1)$$

Where $tf\ (t,d)$ is the times of appearances of word t in essay d. N is the total number of tweets/documents, n_t is the number of tweets/documents that contain word t. Note that the stop words have been ignored for the calculation.
TF-IDF Similarity of two tweets/documents can be calculated by equation (2).

$$sim(d_i,d_j) = \frac{w(:,d_i)^T w(:,d_j)}{\sqrt{w(:,d_i)^T w(:,d_i)}\ \sqrt{w(:,d_j)^T w(:,d_j)}} \quad (2)$$

The TF-IDF Similarity of each tweets/documents to each class is defined as the mean similarity of a given essay to all tweets/documents in a given class.
As to classification algorithm, Naïve Bayes method is implemented. The expression for probability that i is from the j^{th} category, according to Bayes rule shown in the following equation.

$$p\left(c^{(i)}=j\middle|t^{(i)}\right) = \frac{p(c^{(i)}=j)p(,t_2^{(i)},\dots,t_{Nt}^{(i)}|c^{(i)}=j)}{\sum_l p(c^{(i)}=l)p(t_1^{(i)},t_2^{(i)},\dots,t_{Nt}^{(i)}|c^{(i)}=l)} \quad (3)$$

Where feature vector $t^{(i)}$ contains TF-IDF weighting of word $t_n^{(i)}$ in essay i. Nt is the dimension of vector $t^{(i)}$.
Now, assuming that each feature is conditionally independent, equation (3) can be rewritten as:

$$p\left(c^{(i)}=j\middle|t^{(i)}\right) = \frac{p(c^{(i)}=j)\prod_n p(t_n^{(i)}|c^{(i)}=j)}{\sum_l p(c^{(i)}=l)\prod_n p(t_n^{(i)}|c^{(i)}=l)} \quad (4)$$

If we are interested only in the most probable value of $c^{(i)}$, then we have the Naïve Bayes classification rule:

$$c^{(i)} \leftarrow ar\max_j p\left(c^{(i)}=j\middle|t^{(i)}\right) \quad (5)$$

Since the features in feature vector $t^{(i)}$, lie between 0 and 1, the values become continuous, hence $p(t_n^{(i)}|c^{(i)}=j)$ can be calculated by the equations of Naïve Bayes for continuous inputs[1] and I model them as Gaussian distribution.

In practice, $\prod_n p\ (t_n^{(i)}|c^{(i)}=j)$ can be calculated by two methods, the first one is to simply regard it as $sim\ (d_i, class_j)$. The second method is for each essay, to pick its $50 t_n^{(i)}$ which have the 50 largest TF-IDF weighting and regard $\prod_n p(t_n^{(i)}|c^{(i)}=j)$ as

$$\prod_n p(t_{n:t_n^{(i)}\in 50\ \text{largest TF-IDF}}|c\ \ =j) \quad (6)$$

Where

$$p\ (t_{n:t_n^{(i)}\in 50\ \text{largest TF-IDF}}|c\ \ =j) = \frac{1+tf(t_n|c\ \ =j)}{m+\sum_k tf(t_k|c\ \ =j)} \quad (7)$$

In equation (7), m is the dimension of t; $tf\ (t_n|c\ \ =j)$ is the total number of the appearances of word t_n in class j.

However, since tweets are very short and have a high possibility that each word in one tweet appears only once, we just use all the distinct words to construct the feature vector. The main reason for us to adopt Naïve Bayes algorithm is that the principle is concise thus has wide scope of application.

In this project, we employ Map Reduce Pattern Design to create the dictionary and calculate $p(t_n|c\ \ =j)$. The source code for the Map Reduce Algorithm can be seen on our Github Repositories.

B. NLP Tag Analysis

We employ CMU ark-tweet-nlp[11][12][13][14] in our project to tag tweets and we assume that in the parse tree, only the children of right nodes are worth considering. Then we design patterns as follows to extract the noun words as the purchase demand that shows that users need to buy some products.

$$\left[want\ +\ \left(\frac{adj}{adv}\right) +\ (noun\ words)\right]$$

$$[want/hope/wish\ +\ \dots\ +\ buy\ +\ (adj/adv)\ +\ (noun\ words)]$$

C. Map Reduce Pattern Design

In our project, we hope to use sentiment algorithm to automatically calculate the rating of our recommendation on Twitter website from the users' response to our recommendation tweets. Besides, in our web platform, the sentiment algorithm is employed to show the real time overall altitude of the products in our websites. To realize it, we keep querying tweets using the product name as the keyword and implement the Map Reduce Pattern Design to calculate the overall sentiments of all the products.
The pseudo code of the map reduce pattern design is shown in Figure 4.

```
Mapper<text, DoubleWritable>   {
        for each word k in text
                if (negativeSet.contains(k))
                        negativeCount = negativeCount + 1;
                if (positiveSet.contains(k)) {
                        positiveCount = positiveCount + 1;

                rating = (positiveCount - negativeCount) /(positiveCount + negativeCount);
                word.set(productName);
                context.write(word, new DoubleWritable(rating));
}
Reducer{Text key, Iterable<DoubleWritable> values, Context context)   {
        int sum = 0;
        for (DoubleWritable val : values)
                sum += val.get();

        context.write(key, new DoubleWritable(sum));

}
```

Figure 4 Map Reduce Pseudo Code

The inputs of the Mapper is the twitter text, the outputs of the Mapper is the product name and the sentiment score. The Inputs of the reducer are the product name and the score, and the outputs are the product name and its score sum.

D. User-Based Recommendation with Mahout

We use Euclidean Likelihood with Nearest User Neighborhood in Mahout to do recommendation on our web application. In our early stage of collecting data, users have few correlations on a variety of products. Therefore, among the various similarity matrices that Mahout offers, we adopt Euclidean Likelihood.

Euclidean similarity computes the Euclidean distance d between two such user points and returns $1 / (1 + d)$ as its measure of similarity. According to Pearson similarity, if two users overlap on only one item, no correlation can be computed because of how the computation is defined. In Mahout, the cosine measure similarity and Pearson correlation have the same definition because Mahout centers the series of input values to 0. On the other hand, Log likelihood similarity is based on the number of items in common between two users, and it evaluates how unlikely it is for two users to have so much overlap, given the total number of items out there and the number of items each user has a preference for. To sum up, Euclidean matric is more likely to give recommendation when our data is still rare. In our case, it helps to collect data (need feedbacks from user on more recommended products) in our experiment.
For future algorithm improvement, we have two main directions: one is content-based recommendation with preference inferring and another is gender-based preference inferring.

Content-based recommendation takes into account the content or attributes of items, while collaborative filtering approaches are based on user associations with items only. This is a popular approach to enhance the recommendation techniques.

It's likely that users have implicit preferences for certain item attributes, which come out in their preferences for certain items and not others. This evaluation would need assigned product attributes, which we haven't collected yet. By discovering these associations, and discovering attributes of items, it's possible to construct recommender engines based on these more nuanced understandings of user-item associations.

These techniques come to resemble search and document-retrieval techniques: asking what items a user might like based on user-attribute associations and item attributes resembles retrieving search results based on query terms and the occurrence of terms in documents. Mahout's recommender doesn't support these techniques for now, so we are still working on it.

Gender-based recommendation would need gender information from part of the costumers. We separate the user data that gender info are known into those that are known to be men and those known to be women. Then we also determine and remember whether a user seems to rate more male or female profiles from rating history. These results are cached in two additional sets. We filter out these two sets based on an intuition: women who rate male profiles should get recommendation based on other women who also rate more male profiles rather than general male and female profile recommendation. The recommender should not estimate these women's preference for female profiles because such an estimate has no foundation. Taking one step further, the recommender should do recommendation based on ratings from female users rather than all users. On the other hand, male users should get recommendation from a similar algorithm.

V. SOFTWARE PACKAGE DESCRIPTION

A. Package Description

The entry main function of recommendation on twitter side is GetFollowersIDs.java in BigDataProject/ javaResources/src/cmu.arktweetnlp directory.

This function is used for fetching tweets from our followers in nearly real time, detect desires, query products, respond with recommendations and analyze the users reviews. In order to run it, you would have to firstly include your own AWS credential and twitter credential.

The folder named Classification contains all the Map Reduce pattern designs. WordCount.java is used for constructing the dictionary. ClassWordCount.java is used for training the data set. Sentiment.java is used for the implementing the sentiment algorithm. In order to run them, we suggest you first build the path to make your Eclipse be able to run Hadoop and then open the run configuration to set the input path and the out put path. More details will be included on the readme file on our github.

In the folder named WebApp, there are two subfolders including Recommend2U and sign_in_with_twitter. Recommend2U folder contains the front-end code, servlet and database API; sign_in_with_twitter folder contains OAuth2.0 codes to allow Twitter users be redirected to Twitter and get authentication afterwards.

B. How to Use the Web

Users can simply get access to our Twitter application by following the host account. The following part gives instructions on how to interact with the web application.

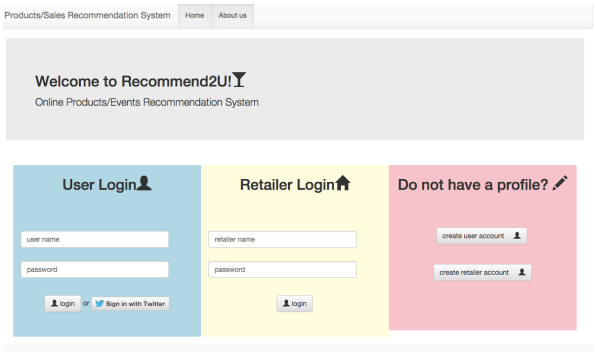

Figure 5 Index Page

Index/Login page allow users/retailers to login the system. Users can use their social media account (Twitter account) or choose to create a separate account on our platform.

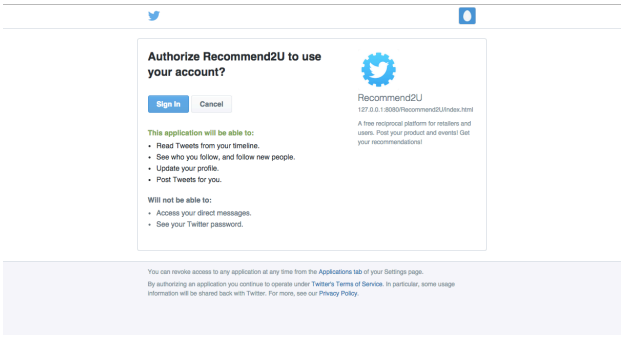

Figure 6 Sign in with Twitter

Use OAuth2.0 to allow authentication for Twitter users to login.

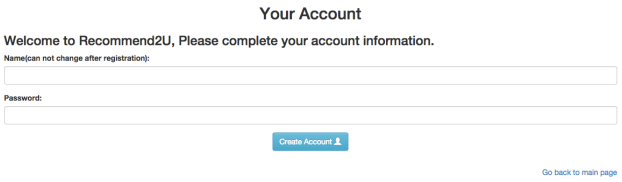

Figure 7 Create a User

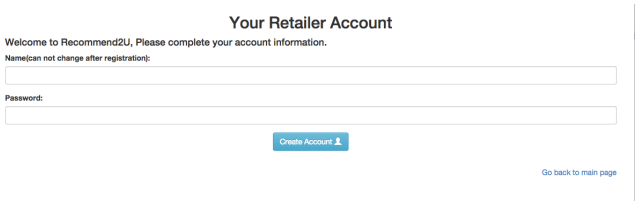

Figure 8 Create a Retailer

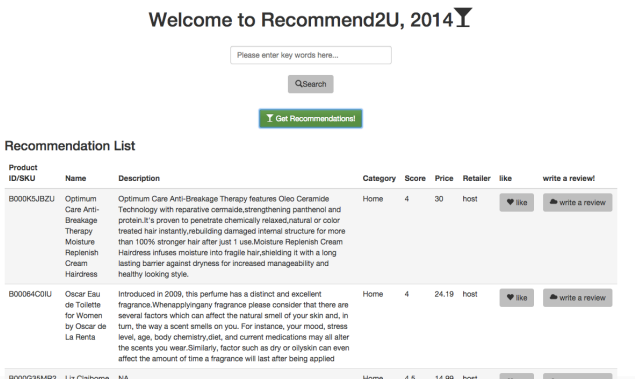

Figure 9 User Main Page

In user main page, users can search products/sales events by keywords and get recommendations list. They can also like or write reviews for our recommendation

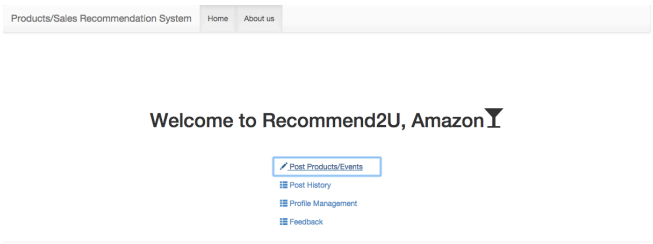

Figure 10 Retailer Main Page

In retailer main page, users can post products/sales events, retrieve post history, modify profile and get feedbacks. In the screen shot, we use Amazon as a retailer example.

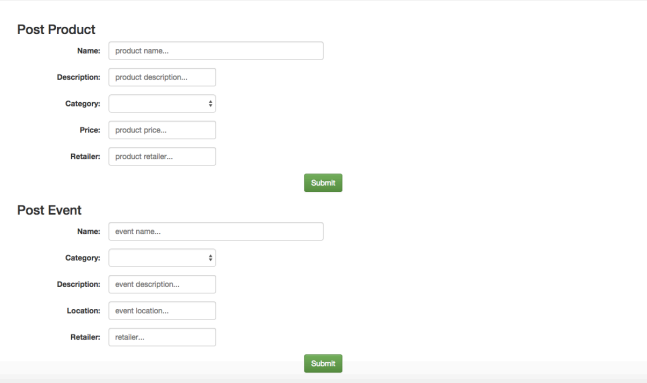

Figure 11 Post Products/Sales Events

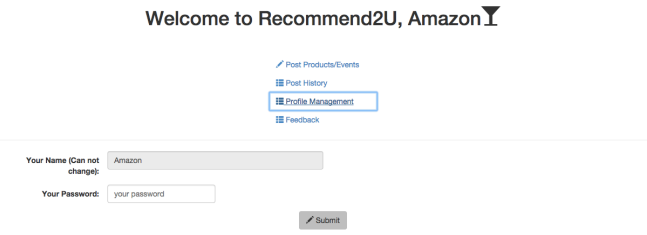

Figure 12 Profile Management

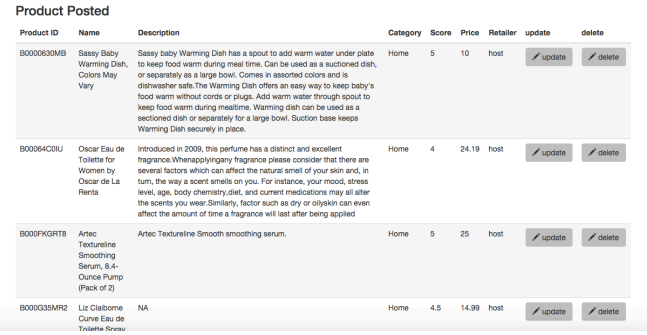

Figure 13 Retrieve/Update/Delete Histories (1)

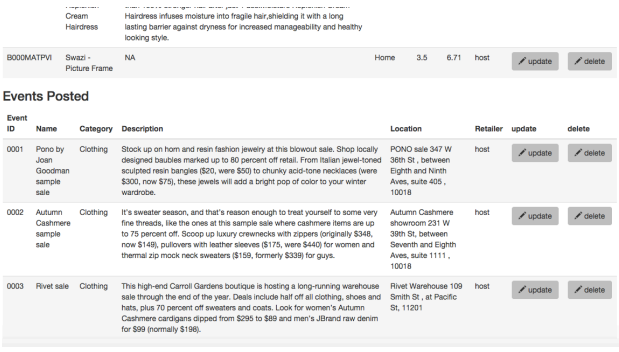

Figure 14 Retrieve/Update/Delete Histories (2)

VI. Experiment Resultsz

To test the application, we invited several test users for our recommendation system. When test users post tweets to express their purchase demand, they also tag tweets to allow us calculate the classification accuracy. The overall accuracy of the classification is 78.6%.

For Twitter users, one obvious advantage for them to use our application is that they can get the desired product hyperlink without going to an e-commerce website.

VII. Conclusion

In general, we have built two related recommendation systems in this project. Moreover, there are still lots of improvements that can be made to this system.

Currently, we only use Amazon as our recommendation resource. However, for the resource of retailers, we have more choices. Retailers including Macy, eBay, Groupon have great API for developers to utilize. By doing this, we can attract more users in the beginning because of enriched recommendation contents thus decrease the "cold start" effect that many collaborative filtering recommendation systems suffer from in the starting point.
Aside from that, for the web application, we are still working on the feedback of retailers based on application users. However, given the limited amount of users at this moment, the feedback report is still meaningless and we would keep work on this aspect and plan to publish it with visualization in D3 when we have enough users.

In addition, we can expand our coverage of social media. Social networks such as Facebook also have sufficient information for us to build the system and may also improve the performance of the recommendation system if these social networks have the similar expression pattern. In fact, social media such as Facebook and blogs are more suitable for accurate purchase demand detection since the text contents in this kind of social networks are of larger length, which would make the sentiment analysis more accurate and improve the accuracy of prediction afterwards.

Acknowledgment

We would like to express our tremendous gratitude to Professor Ching-Yung Lin. He provided us with lots of creative ideas and great inspirations in our project. Also, we appreciate the help from the TA group, especially Bhavdeep Amarjeet Singh Sethi. Their patience and devotions help us overcome dozens of difficulties and finish this project.

APPENDIX

REFERENCES

[1] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[2] How Computers Know What We Want — Before We Do

[3] Hosein Jafarkarimi; A.T.H. Sim and R. SaadatdoostA Naïve Recommendation Model for Large Databases, International Journal of Information and Education Technology, June 2012

[4] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. (2000). "Application of Dimensionality Reduction in Recommender System A Case Study"

[5] Sanghack Lee and Jihoon Yang and Sung-Yong Park, Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem, Discovery Science, 2007.

[6] Collaborative Recommendations Using Item-to-Item Similarity Mappings

[7] Peter, Brusilovsky (2007). The Adaptive Web. p. 325. ISBN 978-3-540-72078-2.

[8] http://en.wikipedia.org/wiki/Recommender_system#cite_note-24

[9] Adomavicius, G.; Tuzhilin, A. (June 2005). "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". IEEE Transactions on Knowledge and Data Engineering 17 (6): 734–749. doi:10.1109/TKDE.2005.99.

[10] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.

[11] Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider and Noah A. Smith. In Proceedings of NAACL 2013.

[12] Owoputi et al. (2012). Technical Report, Machine Learning Department. CMU-ML-12-107. September 2012.

[13] Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments Kevin Gimpel, Nathan Schneider, BrendanO'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan,and Noah A.Smith. In Proceedings of ACL 2011.

[14] Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. InProceedings of EMNLP 2014.