MWSUG 2016 - Paper AA05

Hybrid recommendation system to provide suggestions based on user reviews

Ravi Shankar Subramanian, Oklahoma State University
Shanmugavel Gnanasekar, Oklahoma State University

ABSTRACT

If you have ever shopped on Amazon, Pandora or Netflix, you would have probably experienced recommendation systems in action. These systems analyze the historical buying behavior of their customers and make real time recommendations to them. The back end of these systems contain data mining models that make predictions about the product relevant to you. We plan to build a similar hybrid recommender system to suggest restaurants. We intend to combine content from Yelp reviews, users profile, and their ratings/reviews for each restaurant visited, restaurant details and tips provided by the user.

To implement our idea, we downloaded 2.2M reviews and 591K tips by 552,000 users from the Yelp® website. The dataset for 77,000 restaurants contain information such as user profile information. Traditional systems utilize only user's ratings to recommend new restaurants. However, the system we propose will use both user's reviews or content and ratings to provide recommendations. The content based system is modeled by identifying the preferences for each user and associating them with key words such as cuisine, inexpensive, cleanliness and so on by constructing concept links and association rules based on their past reviews. The collaborative based system is modeled through clustering by aggregating a particular user with other peer users based on the ratings provided for restaurants.

INTRODUCTION

The objective of the paper is to build a recommender system that identifies the preferences of a user to provide individualized suggestions that make their experience enjoyable. Recommender systems are an integral part of social platforms such as Facebook and LinkedIn as well as a part of Ecommerce websites such as Amazon and EBay. The traditional recommender system uses only the ratings to understand the preferences of the user. With the availability of abundant user reviews describing their experience with a particular business, we tried to leverage the reviews along with the user ratings to provide effective recommendations. To build the recommendation engine, we used the data from Yelp - the popular search and review service about local businesses.

RECOMMENDATION TECHNIQUES

Recommender systems are majorly developed based on the two techniques: Content-based filtering and Collaborative-filtering.

CONTENT-BASED FILTERING TECHNIQUE

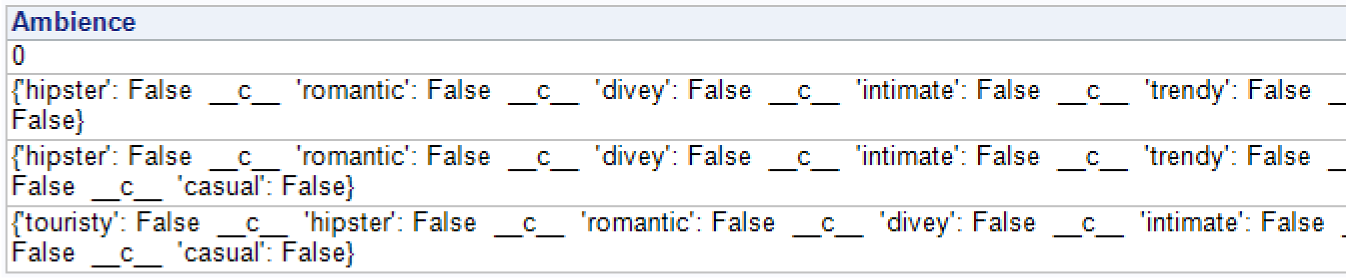
With the content-based filtering technique, the user profile is analyzed to generate the recommendations. For example, if a user prefers restaurant that serve wine and also provide valet parking service, then the recommendation engine provides searches for restaurants with similar attributes such as wine serving and valet parking, from the database and provide those businesses to the user.

COLLABORATIVE-FILTERING TECHNIQUE

The collaborative-filtering systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users. If a user has given a particular rating to a restaurant, the user will get suggestions based on the preferences of other users who have given same rating to that particular restaurant.

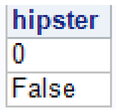
DATA PREPARATION

1. The data for our recommender system comes from yelp website. For the purpose of this paper, we have filtered restaurants in Las Vegas and used them in building the model.
2. Data for restaurants were provided in JSON format. Python script was used to convert the data to csv, filter restaurants in Las Vegas and imported into SAS®.
3. Base SAS code was used in data exploration and data understanding. Variables like "full_address, latitude**,** longitude, attributes, neighborhoods, state, city" were analyzed.
    a. Latitude and longitude details are ignored as we consider only the restaurants that are located in Las Vegas. Since the restaurants are within city limit, we ignored distance between the customer and the hotel.
    b. Variables like "Saturday", "Sunday" containing the opening and closing timings of the restaurants. In this project, we are not going to match restaurants without considering if it is open or not. Our motive is not to make recommendation system perform in real time instead our goal is to model recommender system with the existing rating data and enhance it by mining text reviews. Therefore, we do not take opening and closing times under consideration. In the future, we could use our framework and model it to consider restaurants open hours and customer search time to give real predictions.
    c. Full Address variable is not useful as such. We only need that variable for providing the location details in the output and thus we have removed it from dataset during initial data preparation.
    d. Attributes, categories are nested variables, i.e., it has different attributes variables. These variables are extracted as separate variable, therefore this is a redundant column and has been dropped.
    e. State and city variables are dropped as their values are Las Vegas and NV respectively for the entire subset and they did not contribute any valuable information to the model.
4. Boolean variables are coded as integer values. For example, FALSE is coded as 0 and TRUE is coded as 1.
5. Nested Variables like ambience and Hipster are removed from the dataset.



Ambience
0
{'hipster': False __c__ 'romantic': False __c__ 'divey': False __c__ 'intimate': False __c__ 'trendy': False _ False}
{'hipster': False __c__ 'romantic': False __c__ 'divey': False __c__ 'intimate': False __c__ 'trendy': False _ False __c__ 'casual': False}
{'touristy': False __c__ 'hipster': False __c__ 'romantic': False __c__ 'divey': False __c__ 'intimate': False _ False __c__ 'casual': False}

Various parameters inside these nested variables are extracted into separate variable as shown below.

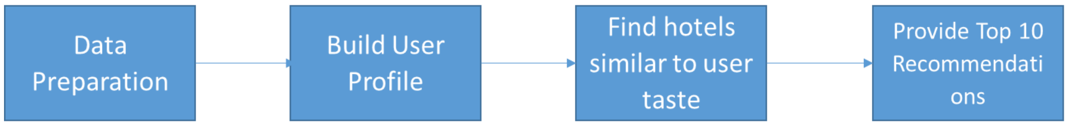


hipster
0
False

6. Attributes which are TRUE for all restaurants or FALSE for all the restaurants are removed from datasets. And any redundant or duplicate variables in the dataset was cleaned as well.

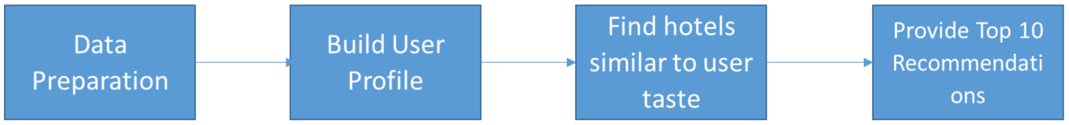
CONTENT RECOMMENDER SYSTEM

We followed below flow to build our Content Recommendation system.


Data Preparation
Build User Profile
Find hotels similar to user taste
Provide Top 10 Recommendati ons

CONTENT-BASED FILTERING TECHNIQUE

The process starts with data preparation, then a user profile is built based on their ratings to different restaurants. After building the user profile, we can suggest restaurants that match their tastes using the formula provided below. We recommend top 10 restaurants that match with user's taste.


Data Preparation
Build User Profile
Find hotels similar to user taste
Provide Top 10 Recommendati ons

$$\boxed{\begin{aligned} Prediction &= user\ preference\ of\ the\ attribute \\ &* Idf\ of\ the\ attribute * business\ attribute \end{aligned}}$$

Next, we have to decide on the weights for each attribute and have used TF-IDF(inverse document frequency) for this purpose. The IDF is used as a way to assign weights for the attributes, so that common attributes are given less importance, while rare attributes are given high importance.

Steps

Data Preparation

a. We transformed the rating scale of 1 star – 5 stars to 0 and 1. This transformation resulted in an interesting behavior (i.e.) It forces user to have only two choices, he can like it or dislike it. This simplifies some of the calculation in the later steps. For simplicity sake, if the user has rated a restaurant with less than 3 stars, then we consider it as negative experience or in other words we can say he dislikes it. Rating that is greater than 3 are considered as positive and equal to 3 as Neutral. This is implemented in SAS as shown below, where rating less than 3 as negative, rating equal to 3 as 0 and greater than 3 as positive.

```
PROC SQL;
   select *, case ratings
              when ratings < 3 then -1
              when ratings > 3 then 1
              else 0
              end from yelp.review;
quit;
```

b. The data provided contains multiple attributes for each restaurant. The attributes are in Boolean format and we have used it as it is in the model, so this implementation ignores the quality of the attributes. For example, if a reputed restaurant theme is local culture and provides rich immersive cultural experience, it is represented as 1 in "Cultural" attribute. In contrast let us consider a boutique restaurant which also has cultural theme and provides limited cultural experience. The attribute for both the restaurant is represented as 1, and there is no way in our model that accounts for the level or quality of the attribute. If the hotel has particular attribute, then it is 1, else 0.

Attribute Weights

a. Next step in the process is to calculate IDF for each attribute. This is calculated as given in below equation. The maximum in the formulae is to account for an unusual case. Suppose an attribute has 0 for all restaurants, then denominator becomes 0 and IDF goes to infinity. To prevent such scenarios, maximum function is introduced in the formulae.

$$IDF = \frac{1}{\max(no.of\ hotels\ that\ has\ the\ attribute, 1)}$$

```
/*Calculate IDF*/
%macro idf();

   PROC SQL;
      create table idf as
            select %do i = 1 %to &cnt; %IF &I > 1 %THEN ,;
1/MAX(sum(&&col&i),1) as idf_%substr(&&col&i,1,10) %end AS COLNAME;
                  from business ;
   quit;

%mend;
%idf();
```

b. Next step in the flow is to build a user’s profile. If user has previously rated any restaurants, we use that knowledge to build the user’s profile. If user has rated a restaurant positively, then we increment all the attributes associated with the restaurant in the user’s profile by 1 or if a user has negatively rated the restaurant, then we decrement all the attributes associated with the restaurant in the user’s profile by 1 using the below mentioned macro.

```
   /*Create User Profile*/
   %macro userprofiles();
   data yelp.userprofiles_v1(KEEP=USER_ID BUSINESS_ID RATING
      %DO i = 1 %TO &CNT; &&COL&I %END;
                                          );
      set yelp.userprofiles_v1;
      %do i=1 %to &cnt;
            &&col&i = rating * &&col&i ;
      %end;
   run;
   PROC SQL;
      create table userprofiles_v2 as
            select user_id %do i = 1 %to &cnt;
                  ,sum(&&col&i) as sum_%substr(&&col&i,1,10)
                  %end; from yelp.userprofiles_v1 group by user_id;
      quit;
   %mend;

   %userprofiles();
```

Score Prediction

c. After building IDF and user’s profile, we calculated the similarity between the user’s taste and restaurants profile as shown below. To do this we use two pieces of SAS code. First piece of code transposes the IDF into a single column. The second part of the code uses the transposed IDF and user profile to compute prediction as shown below.

```
PROC TRANSPOSE data=idf out=idf_v2;
run;

%macro buildRecommendations(usrid="9A2-wSoBUxlMd3LwmlGrrQ");
/*Extract the given user’s profile*/
   data user_profile1;
   set business;
   if _n_ = 1 then do;
      set idf;
      set userprofiles_v2(where=(user_id=&usrid));
   end;
   run;

   PROC SQL;
      SELECT COUNT(NAME) INTO :CNT FROM DICTIONARY.COLUMNS WHERE
memname='USERPROFILES_V1' and libname="YELP"
                  and type="num";
      SELECT NAME INTO :COL1 - FROM DICTIONARY.COLUMNS WHERE
memname='USERPROFILES_V1' and libname="YELP"
                  and type="num";
   quit;
/*Calculate predicted rating*/
   data user_recommendation;
      set user_profile1;
      %do i=1 %to &cnt;
            rat_%substr(&&col&i,1,10)= max(sum_%substr(&&col&i,1,10),0) *
max(&&col&i,0) * idf_%substr(&&col&i,1,10) ;
      %end;
            score = 0
      %do i = 1 %to &cnt;
            + max(rat_%substr(&&col&i,1,10),0)
      %end;
   ;
   run;

   %mend;

%buildRecommendations(usrid="9A2-wSoBUxlMd3LwmlGrrQ");
```

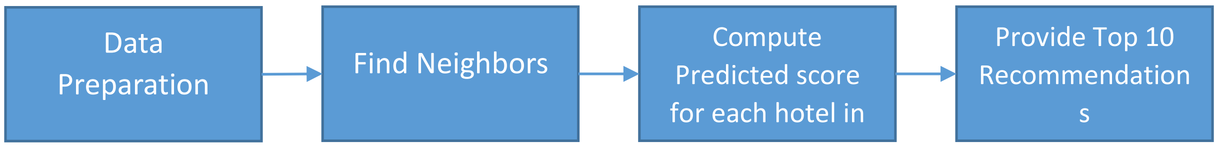
Recommend Top 10 items

d. After calculating the score, we return the top 10 restaurants that match the user profile using the below code.

```
/*Give Me Top 10 Recommendations*/
PROC SQL outobs=10;
   select user_id, score, rating from user_recommendation order by score
desc;
quit;
```

COLLABORATIVE-FILTERING SYSTEM

In the collaborative system, to recommend a restaurant to user, we use other users with similar taste (a.k.a neighbors) and make recommendations based on the feedback provided by others users. For example, it is very common to get recommendation from your friends and family who has similar taste for suggestions to find new restaurants. Collaborative recommender system works in a similar fashion.


Data Preparation
Find Neighbors
Compute Predicted score for each hotel in
Provide Top 10 Recommendation s

To build Collaborative recommender system, we first decide on number of neighbors for a user. We have used 18 neighbors in our model and this number can be decided based on domain knowledge. After determining the number of neighbors, we use correlation method to find 18 closest neighbors. We also decide on a threshold and if any user is farther than the distance decided by threshold, then we don’t include him as our neighbor. After selecting neighbors, we compute the predicted score for each restaurant by using below formula and recommending top 10 restaurants from our recommendation.

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|}$$

In the above formula, u index represents user for whom we're giving recommendations and u' are user's neighbor. s(u,u') – is the similarity co-efficient that is calculated using neighbors distance. If neighbors taste is very close to user's taste, then similarity co-efficient is high and if the neighbors taste is further from user's taste then similarity co-efficient is low. In above formulae, we calculate weighted average of neighbors rating to predict user's rating.

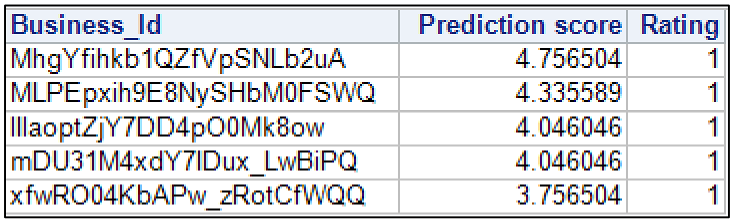
$$s(u,u') = \frac{1}{Distance}$$

1. In data preparation stage, we build a user profile similar to one built in content based recommendation system.
2. Then we compute predicted score based on the formulae given above

```
%MACRO PREDICTSCORE(USRCNT=15);
%DO I = 1 %TO &USRCNT;
   PROC SQL;
      CREATE TABLE USERUSERPREDICTION&I AS
            SELECT BUSINESS_ID, USER_ID, &&RU&I AS RU,
                        "&&USER&I" AS USER, CASE USER_ID
                        %DO J = 1 %TO &&NCNT_&I;
                              WHEN "&&&&NNAME&I._&J." THEN
&&&&NEIGHBOR&I._&J. * (STARS - &&RN_&I._&J)
                        %END;
                        END AS WEIGHTEDSCORE,
                        CASE USER_ID
                        %DO J=1 %TO &&NCNT_&I;
                              WHEN "&&&&NNAME&I._&J." THEN &&NEIGHBOR&I._&J.
                        %END;
                        END AS W
                              FROM YELP.USER_PROFILE_V1 WHERE
                        USER_ID IN (
                                    %DO J = 1 %TO &&NCNT_&I;
                                          "&&&NNAME&I._&J" %IF &J<&USRCNT
%THEN ,;
                                    %END;
                                           );
   QUIT;
   PROC SQL OUTOBS=10;
            SELECT BUSINESS_ID, SUM(WEIGHTEDSCORE)/SUM(W) + RU AS PREDSCORE
FROM USERUSERPREDICTION&I GROUP BY BUSINESS_ID, RU ORDER BY PREDSCORE DESC ;
   QUIT;
%END;
%MEND;

%PREDICTSCORE();
```

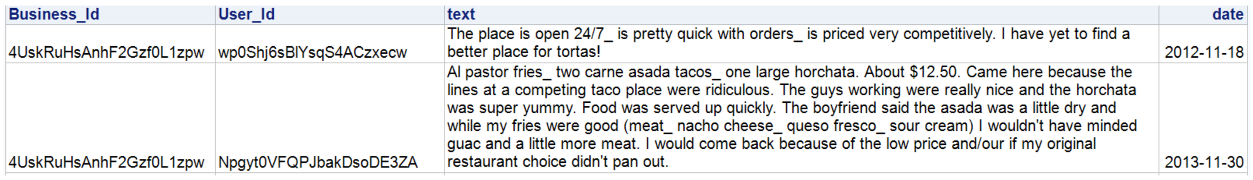
3. After computing ratings for all restaurants, we then output top recommendations to the user.



Business_Id	Prediction score	Rating
MhgYfihkb1QZfVpSNLb2uA	4.756504	1
MLPEpxih9E8NySHbM0FSWQ	4.335589	1
IllaoptZjY7DD4pO0Mk8ow	4.046046	1
mDU31M4xdY7lDux_LwBiPQ	4.046046	1
xfwRO04KbAPw_zRotCfWQQ	3.756504	1

RECOMMENDATION BASED ON USER REVIEW

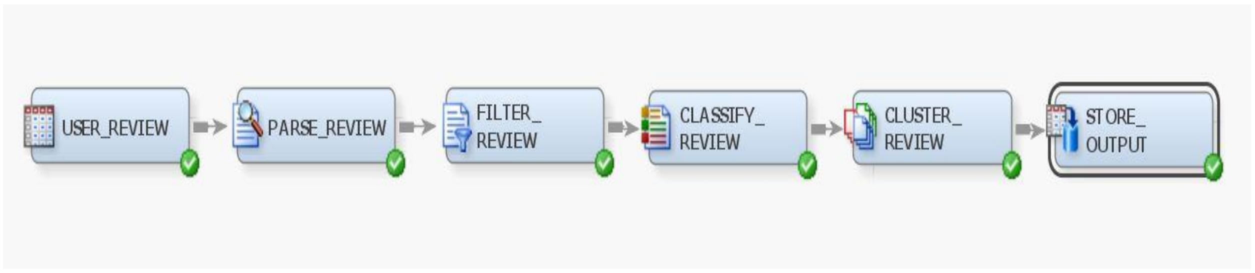
The Yelp dataset contains reviews given by a user for various restaurants that can be analyzed to find the preferences of the user. The User profile dataset contains reviews for all types of businesses such as food chains,grocery, and clothes stores. We narrowed our scope to only the restaurant business places and took only the reviews corresponding to the restaurants for consideration.



Business_Id	User_Id	text	date
4UskRuHsAnhF2Gzf0L1zpw	wp0Shj6sBlYsqS4ACzxecw	The place is open 24/7_ is pretty quick with orders_ is priced very competitively. I have yet to find a better place for tortas!	2012-11-18
4UskRuHsAnhF2Gzf0L1zpw	Npgyt0VFQPJbakDsoDE3ZA	Al pastor fries_ two carne asada tacos_ one large horchata. About $12.50. Came here because the lines at a competing taco place were ridiculous. The guys working were really nice and the horchata was super yummy. Food was served up quickly. The boyfriend said the asada was a little dry and while my fries were good (meat_ nacho cheese_ queso fresco_ sour cream) I wouldn't have minded guac and a little more meat. I would come back because of the low price and/our if my original restaurant choice didn't pan out.	2013-11-30

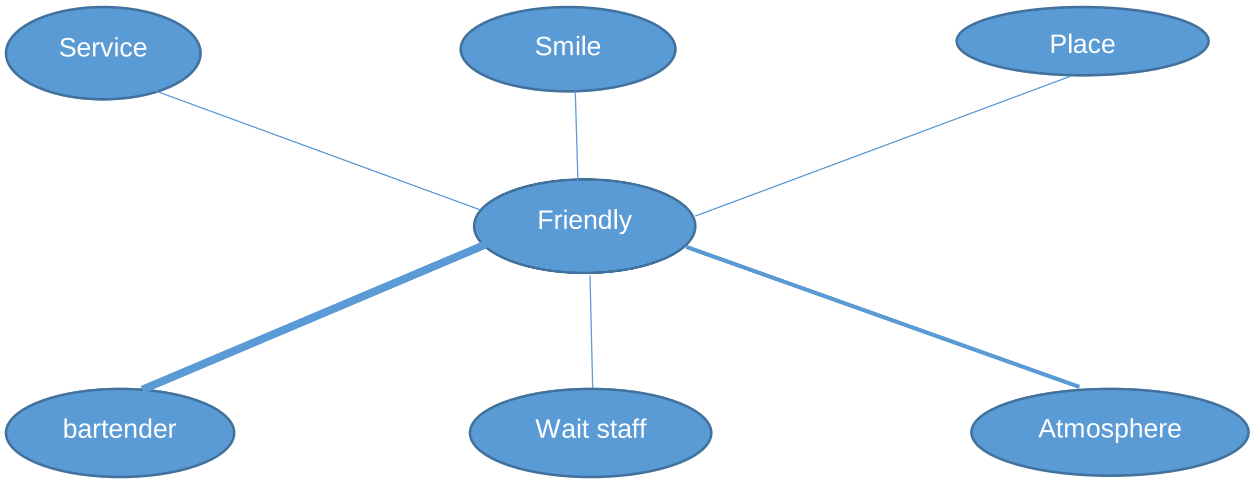
TEXT MINING THE USER REVIEWS

The filtered user review dataset is fed as an input to the text mining model created using SAS Enterprise Miner 14.1. The review is fed into a text parsing node where the some of the common data cleaning techniques such as filtering the common words such as the,an,I,be etc were removed from the data using the SAS default Stop List. Stemming of the words is carried out to group different words with same origin together. Certain parts of speech namely conjuction,preposition, and auxillary verbs that are not going to provide valuable information were removed from the reviews.



After parsing the user review, the output of the text parsing node is fed as an input to the Text Filter Node where the importance terms were filtered based on Inverse document frequency (IDF). Reviews are prone to spelling errors hence the spell-check was appied on the review text using a custom english dictionary.

Term Association
Text filter node helps in identifying the terms associated with the key terms. For example, the term friendly is associated with various terms as show below. This will help in content based filtering to provide recommendations.

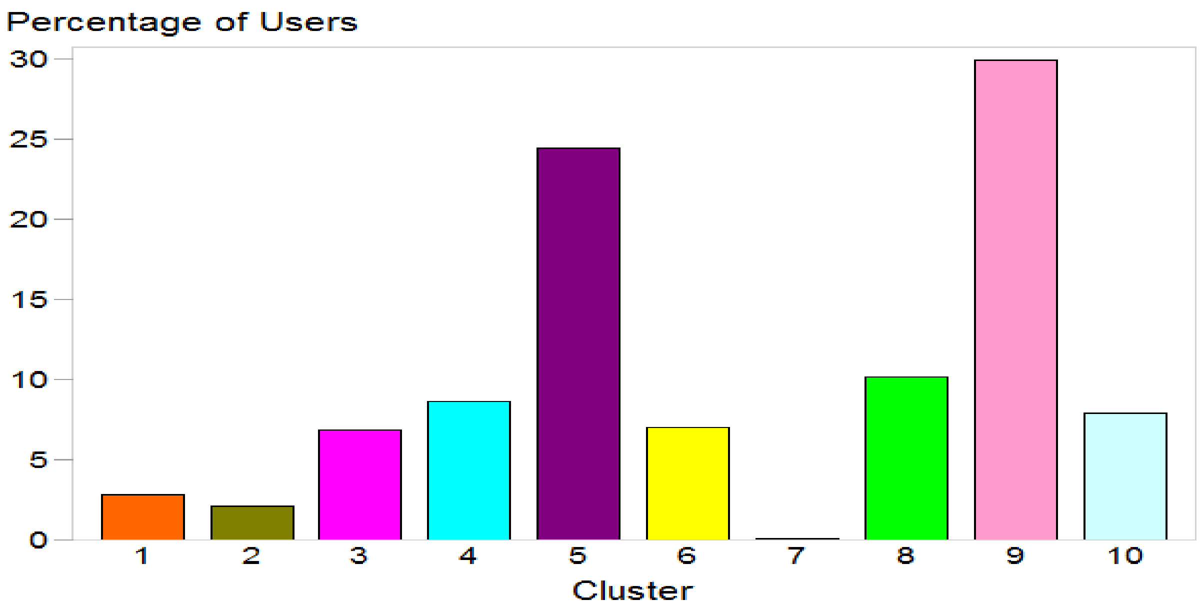


After filtering the reviews, the output of the text filter node is fed as an input to the text topic node to classify the reviews as positive and negative based on their sentiment. The default user topic table provided by SAS is used to calculate the sentiment probability for each review.

Clustering the similar users based on review
Clustering similar users based on their preference is essential to build a collaborative-filter recommendation engine. The Text cluster node helps in decomposing the term document matrix and create cluster based on the distance between each term. Expectation-Maximization algorithm is used to cluster terms. The goal of the clustering algorithm is to maximize the overall probability or likelihood of the data, given the final clusters. By doing this, similar users are grouped under same cluster.

Below are the clusters created as a result of text clustering in SAS EM.

Cluster	Descriptive Terms	Frequency
1	always+beer+cheese+Crust+good+order+pepperoni+pizza+place+salad+sauce+slice+taste+thin	8192
2	Appetizer+happy hour+beer+bar+great+half+night+roll+price+special	6055
3	back+bread+cheese+chicken+delicious+french+line+long+lunch+minute+night+order	19853
4	beef+chicken+chinese+dis+egg+food+fry+good+lunch+noodle+pork+portion	24855
5	buffet+desert+diner+potato+sauce+meal	70604
6	bacon+breakfast+coffee+egg+french+friendly+hash+pancake+place	20222
8	burger+bar+beer+cheese+chicken+fry+lunch+potato+place+order	29425
9	bar+beer+back+dinner+drink+experience+great+food+meal+man+look	86368
10	bar+eat+fish+fresh+mexican+love+roll+salsa+spicy+sushi+taco	22874



The above graph shows the population distribution of users across each clusters. Users in same clusters are assumed to have similar preferences. Hence the recommendation based on collaborative filtering can be done by providing the suggestions to a user based on the neighboring users in same cluster.

FIT STATISTICS

The fit of our model is calculated using the two metrics namely RMSE (Root Mean Square Error) and MAE (Mean Absolute Error). Those metrics for the top 10 recommendations were calculated using formulae given below

$$RMSE = \sqrt{\frac{\sum(pred - rating)^2}{N}}$$

$$MAE = \frac{\sum|pred - rating|}{N}$$

The RMSE and MAE for both content based recommender system with and without review is 0.447 and 0.2. The RMSE and MAE for collaborative based recommender system with and without review is 0.316 and 0.1.

The new method (including the review along with the rating for recommendation) we proposed did not add any significant improvement on RMSE and MAE rating. But we calculated RMSE and MAE on only 10 users. If we automate calculation of RMSE and MAE using macros and calculate for a larger holdout sample, then we would likely get a better honest assessment.

CONCLUSIONS

- The new method we proposed did not improve our chosen fit statistic, RMSE and MAE. Secondly, we calculated RMSE and MAE on only 10 users. Therefore calculating RMSE and MAE on large population will be a better assessment of the model.

- With certain improvements, this method could be used as an alternative way to build a recommender system in SAS other than the usual way of using PROC RECOMMEND which requires LASR server.

REFERENCE

- The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review, Ivens Portugal.
- Kamba, T., Bharat, K., & Albers, M. C. (n.d.). The Krakatoa Chronicle - An Interactive, Personalized, Newspaper on the Web. Retrieved from http://www.w3.org/Conferences/WWW4/Papers/93/
- Ekstrand, M. D., Riedl, J., & Konstan, J. (2011). Collaborative filtering recommender systems. Hanover, MA: Now.
- "Discounted Cumulative Gain." Wikipedia. Wikimedia Foundation, n.d. Web. 26 July 2016.

ACKNOWLEDGEMENTS

- We thank Dr. Goutam Chakraborty, Director, Master of Science in Business Analytics – Oklahoma State University and Dr. Miriam Mcgaugh – Oklahoma State University for their continuous support throughout our research.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Ravi Shankar Subramanian
Oklahoma State University
405-762-3625
Ravishankar.subramanian@okstate.edu

Shanmugavel Gnanasekar
Oklahoma State University
813-810-5630
Shanmugavel.gnanasekar@okstate.edu