Restaurant Recommendation using Perceptron and SVM Classifiers

Chandrika Sharma and Peter Mascarenhas
College of Computer and Information Sciences,
Northeastern University
Boston, MA02115

Abstract

The following technical paper describes the process of designing an algorithm to make personalized suggestions of restaurants to the users of a restaurant search and discovery service.

This recommendation algorithm learns from user reviews and his scores to generate a list of personalized suggestions to improve and minimize their search experience for the next restaurant.

Using survey data from Zomato and Yelp that is modeled to our needs, the recommendation system learns what kind of restaurant a person likes and makes a list of recommendations to them. This can be integrated to run on data from multiple vendors such as Yelp etc. We use the Perceptron classifier with weighted features viz. a viz. Cost for two, distance, type of cuisine, Happy hours etc. to identify the unique set of features in a restaurant that appeals to our user which will be weight adjusted by learning on every iteration from a mixed collection of relevant and irrelevant restaurants. Further an SVM classifier has also been implemented to do a similar classification and the results between the two are compared.

To give the user a wholesome experience, we have used Amazon's Item-to-Item collaborative filtering to add restaurants in the suggestions that the user's most similar counterpart in his immediate friend circle prefers. We propose that with these algorithms coupled together a hybrid recommender system will make suggestions that are personal to the user.

I. INTRODUCTION

Recommendation algorithms are best known for their use on e-commerce Web sites where they use input about a customer's interests to generate a list of recommended items. [2] Recommendation systems provide personalized, relevant recommendations to users and have been used in various domains such as music (Spotify), App Stores (Apple Store and Play Store), movies (IMDB) etc. However, the fact that this very relevant feature hasn't found itself in the food domain inspired us to do something about the same. Being foodies ourselves and regulars on the food blog stack we built a one of a kind restaurant recommendation system.

Yelp and Zomato are the two leading businesses that provide a platform for users to voice their opinions about the businesses. If we could provide the users personalized suggestions as an incentive for taking time and writing a review they could use a feature like the one we're building. This will result in a win-win situation for both the users and the business.

II. DOMAIN DESCRIPTION

Our data set comprises of over 60 Restaurants and their what we felt were the most relevant binary feature set clubbed into 6 grouped features. Following are the features used:

Types of dining: Dine-in, Delivery, Drinks & Night life,

Type of Food: Fast Food joints, Dessert, Healthy Food,

Cuisines: Indian, American, Chinese,

Cost For two: < 20, 20 to 30, > 30,

Operation timing: < 9am, > 9 am < 11pm, > 11pm < 3am.
Perks: Serves alcohol, Happy Hours, Free Wi-Fi, Gluten Free food, Kids Friendly.

We have taken data from 5 food enthusiasts for 60 restaurants each. This dataset consists of restaurants and the features it offers. Each restaurant is labeled as "Like" or "Dislike" depending on the user. The perceptron and SVM classifier then train on this dataset and make a few suggestions. For testing purposes we make it predict the likability of another labeled test set of restaurants. The perceptron and SVM classifier classifies these restaurants depending on the final hyperplane from the training set of each user.
Ideally we would pick the top 'n' restaurants in a given geographical location depending on the user's preference of distance thereby making a 'n' suggestions.

III. SYSTEM DESCRIPTION/TASK DEFINITION

The general overview of our system is shown in Figure 1. The dataset is retrieved from a particular business viz. a viz. Yelp or Zomato.

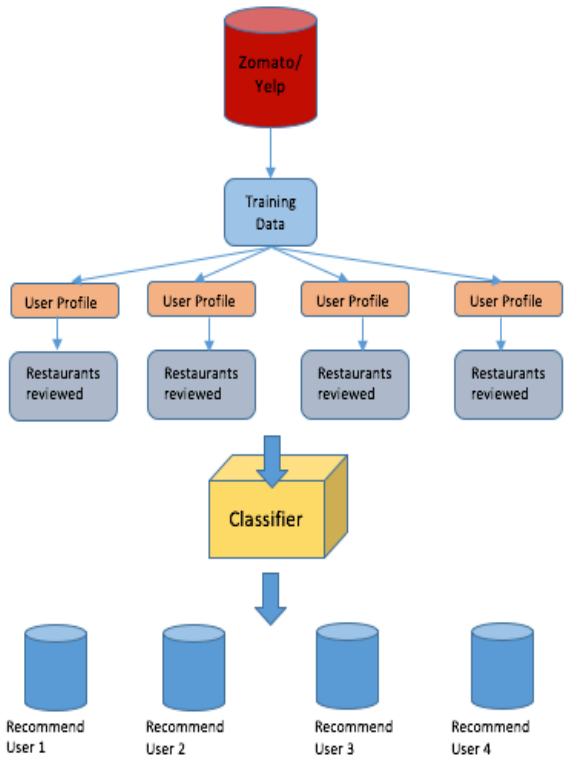

Figure 1

After preprocessing the data and modelling to our system specific input. this will be our initial training data. Weights are initialized at 0 for every feature. This weighted vector will form our hyperplane for classification. This will then be tested on a test set and the labels of user likability will be predicted.
Ideally we would want to break down a restaurant review to know which particular features need to be changed specifically. For example, a user who loves Chinese food might not necessarily like the next best Chinese restaurant. He would probably want to dine at a fancier Chinese place than a cheaper one irrespective of the quality of food. We would ideally need to use an NLP parser (would require greater knowledge in NLP) that could break down a user review along the dimensions of cuisine, cost for two, happy hours, service etc. This will be part of our future work.

IV. PROBLEM DEFINITION AND METHODS

Using the dataset rated by users as "Liked" or "Disliked" to rate whether they like a restaurant with certain features or not, each review is inputted as a row into the classifier with binary labels for each feature with a label for likability of the restaurant. Initially the weight vector is zero for all features. The purpose of the weight vector is to learn the most relevant features to the user depending on his likability. As the system learns, the weight vector will shift to categorize the restaurants according to patterns created in the data set. The final weight vector obtained after running the training set will form our hyperplane that will be our boundary classifier for relevant and irrelevant restaurants. These feature weights are bound to change on learning more reviews.

ALGORITHMS USED AND MODELS:

1) PERCEPTRON

2) SUPPORT VECTOR MACHINE

3) ITEM TO ITEM COLLABORATIVE FILTERING

Perceptron

We started by building a recommendation system for individuals using Perceptron.

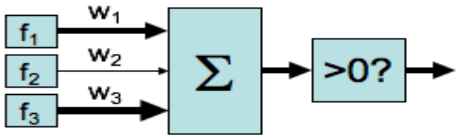

Figure 2

A perceptron classifier is a simple model of a neuron. It has different inputs (f1, f2...fn) with different weights (w1, w2...wn). [Figure 2] The weighted sum of these inputs is then passed through a step function. [Figure 3]

$$\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)$$

Figure 3

We started by initializing the weight vector to zero for each feature. The weights are adjusted depending on the likability labels of the restaurants. Our initial weights for all the features would look like w1[0 0 0], w2[0 0 0] etc. Therefore, in our first learning attempt we might get a lot of suggestions wrong as there maybe multiple irrelevant restaurants that will match the formula below. This is how we achieve our initial hyperplane that will move around as we learn using the perceptron. On iterating over the other reviews we learnt a hyperplane that reflects their preferences. Hence, how happy the user is about the restaurant corresponds to the distance of the restaurant to their hyperplane.

Support Vector Machine

In machine learning, Support Vector Machine (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification. Given a set of training data, marked clearly to belong to one of the two labels in a classifier, SVM learns itself such that it can assign any new training examples to the right label depending on the weighted feature set. SVM creates a clear boundary between the two categories which separates the data. It aims to maximize the margin between them and makes it more accurate than a Perceptron classifier.[10]

Classifier Margin of a linear classifier is defined as the width of the boundary that could be increased before hitting a data point. An SVM only takes into account the support vectors. Support Vectors are only those data points that the margin pushes up against. By maximizing the margin and keeping the support vectors in mind, the rest of the data points can be ignored. Figure 4 gives a general idea of the linear classifier with the weight vector and bias and the margins with support vectors. Figure 5 and 6 define the Margin of the SVM.

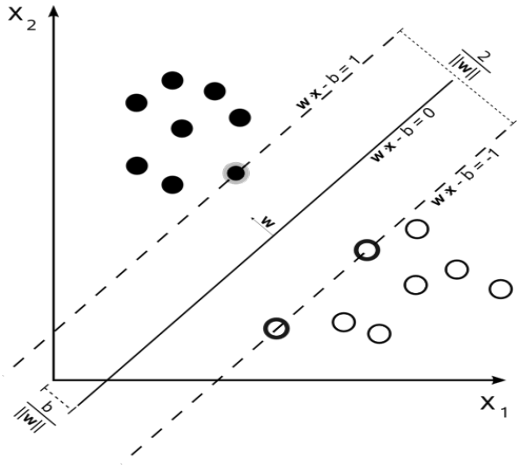

Figure 4[10]

$$w \cdot x^+ + b = +1$$
$$w \cdot x^- + b = -1$$
$$w \cdot (x^+ - x^-) = 2$$

Figure 5[1]

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Figure 6[1]

Above is an instance of a maximum-margin hyper-plain trained by an SVM.
Training dataset of 60 restaurants were given of the form:
X = {list of feature values (0/1) for restaurant characteristics}
Y = {Label of restaurant likability (0/1)}

```
array([[ 1.,  1.,  0.,  1.,  1.,  1.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,
         1.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,  0.,
         1.,  1.,  1.,  0.,  0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         0.,  0.,  0.,  0.,  1.,  1.,  1.,  1.,  0.],
```

Figure 7

Training data for each restaurant was an array. Each value depicts if the feature is present in the restaurant or not. [Figure 7] A list of such arrays are inputted into the machine for training as X. Y are the label for each restaurant quantifying if the user liked the restaurant or not.
With use of libraries like Sklearn and Numpy, standard SVM algorithm was run to train the system.

Item–item collaborative filtering

Combining a feature of the Amazon recommendation algorithm that personalizes the online store for each customer. The store radically changes based on customer interests, showing programming titles to a software engineer and baby toys to a new mother. [5] It measures the similarity of two customers, A and B, in various ways; a common method is to measure the cosine of the angle between the two vectors. There exist such features on Yelp and Zomato where users are connected together as "followers" or "friends". If a user highly reviewed a place, there is a chance a similar user who hasn't been to the place might like it as well. This can be used as a separate "Restaurants your friends liked" feature or could be integrated with the suggestions made from the perceptron. The similarity is calculated as follows:

```
features are cuisine, cost, distance, ambience, serves alcohol etc

weights are [0 0 0 0]

for each restaurant in user_reviewed_list
    for each friend_of_user
        if user_review_Label = friend_of_user_review_label
            Record friend_of_user

for each recorded_friend_of_user
    Record latest_highly_reviewed_restaurant

Add latest_highly_reviewed_restaurant to Suggestions
```

Figure 8

V. EXPERIMENTAL EVALUATION

We used data set from survey of 5 food enthusiasts. These people carefully studied each feature of 60 restaurants to rate if they liked the restaurant or not. This data was fed into the machine as training data.

Dine-in
Delivery
Drinks & Nightlife
Sandwich+FastFood+Coffee+Pizza
Dessert
Healthy Food
Indian
American
Chinese
Less than $20
$20 to $30
$30 +
< 9 am
> 9 am < 11 pm
> 11 pm < 3 am
Serves Alcohol
Happy Hours
Wi-Fi

The features are clubbed under similarity as Dining, Type of Food, Cuisine, Cost, Operating hours, Perks. Each restaurant has a different feature set depending on their business.

Test Results and Discussion

We classified 60 restaurants into our binary feature set for 5 users. To ensure a metric for performance we took 10 of the restaurants from the training data to see if the suggestions matched with label given by each user. Once we iterate over the training data we would have computed the hyperplane for each user and hence we can run over the test set to make a list of restaurants that could be suggestion. These suggestions depend on the distance of the restaurant from the hyperplane obtained during the training set.

We use the following formula to compute accuracy.
X=No. of right predictions from test set

Y=No. of restaurants the user would visit from the test set

Percentage of Accuracy= (X/Y)*100

Using the Perceptron algorithm, we got an average accuracy of (4/5)80% for the 5 users that were suggested restaurants. One of the reasons for a lower accuracy could be the size of the test data set. The restaurants that were wrongly predicted had most of the features that the users loved if not all.

Coupling these results with the item to item collaborative feature that we integrated we achieved a higher accuracy of 85.7%. We noticed that the users would have a higher chance of visiting a similar user's (friends) restaurants if this was maintained as a separate feature instead of it being slipped in with the suggestions. This is more of a social factor that will result in more hits than a miss. This is probably a note the product team could take notice of.

We know that SVM works really well on a small dataset. We have used standard libraries in our implementation of the SVM. Using these standard libraries in our code we achieved a 100% accuracy on the prediction of labels for our users on test data. However grand this result might seem the probability of such high accuracy could be because of the relatively small dataset.

Algorithms	**Average Score (X/Y)**	**Accuracy**
Perceptron	6/7	85.7%
SVM	7/7	100%

VI. FUTURE WORK:

Improving the weights by using an NLP parser that tells us exactly which weight needs to be reduced or increased instead of adding/subtracting the feature vector every time we encounter a label mismatch. For example, consider a yelp review about a restaurant which has 4 stars: "*They have the best happy hours, the food is good, and service is even better. When it is winter we become regulars". [9]*

If we look at only the rating, it is difficult to guess why the user rated the restaurant as 4 stars. However, after reading the review, it is not difficult to identify that the review talks about good "food", "service" and "deals/discounts" (happy hours). This will enable us to have a weight vector with high precision.

This idea has great potential to be integrated with the existing Yelp and Zomato platforms as recommendations haven't ventured into the food business yet. This would also work as a perk for the users taking their time off to review a business on a social platform.

collaborative filtering? different features?

VII. REFERENCES

[1] Mingyue Tan- www-labs.iro.umontreal.ca/~pift6080/H09/documents/papers/svm_tutorial.ppt

[2] https://www.cs.cornell.edu/~rahmtin/files/YelpClassProject.pdf

[3] Prem Melville and Raymond J. Mooney and Ramadass Nagarajan- http://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf

[4] Pieter Abbeel – UC Berkeley - https://inst.eecs.berkeley.edu/~cs188/sp12/slides/cs188%20lecture%2021%20--%20perceptron%202PP.pdf

[5] Greg Linden, Brent Smith, and Jeremy York, Amazon.com- https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

[6] KhushbooShah - http://www.eater.com/2015/2/24/8100527/luka-artificial-intelligence-restaurant-recommendations-sf-nyc-app

[7] https://www.quora.com/What-recommendation-system-algorithm-is-used-by-big-social-giants-like-Facebook-and-YouTube

[8] Danilo Bargin - https://blog.dbrgn.ch/2013/3/26/perceptrons-in-python/

[9] Hitesh Sajnani, Vaibhav Saini, Kusum Kumar, Eugenia Gabrielova , Pramit Choudary, Cristina Lopes http://www.ics.uci.edu/~vpsaini/

[10] https://en.wikipedia.org/wiki/Support_vector_machine