Search Final Project Report

Shubhankar Mitra
Indiana University
Bloomington, IN 47408, USA
shmitra@iu.edu

Piyush Shinde
Indiana University
Bloomington, IN 47408, USA
pshinde@iu.edu

Saurabh Kumar
Indiana University
Bloomington, IN 47408, USA
kumarsau@iu.edu

ABSTRACT

Recommendation systems are an essential part of any modern day service and utility provider. It is also a big part of any information based websites like IMDB and Yelp. We take the yelp challenge data set to build a recommendation system, based on user reviews and restaurant categories. This recommendation system is specific only to restaurants. We also do topic modelling on the user reviews to tag reviews to user defined topics and extract relevant reviews.

For task two we explore the use of word2vec in combination with Latent Dirichlet Allocation (LDA) to extract reviews with user desired content and subject matter.

1 INTRODUCTION

The first task is the data prepossessing and filtering. The data was converted to csv format and three tables, business, user and review, were used for our model. We filtered only the restaurant data for 3 cities Charlotte, Scottsdale and Pittsburgh. We also removed data for users with less than 20 reviews. To evaluate our model we divided the data set into 3 parts, train, validation and test.

The recommendation system is built using 2 algorithms - Content Based Recommendation and Collaborative Filtering. In content based recommendation, we use 2 methods - user based similarity and item based similarity. The recommendation system predicts the ideal restaurants for an user, based on his/her reviews and the restaurant categories. Each content based recommendation algorithm is supported by 3 feature extraction algorithm - Non-Negative Matrix Factorization (NMF), Term Frequency Inverse Document Frequency (TF-IDF), Document to Vector (Doc2Vec). The user based model uses the reviews provided by user and the item based model uses the categories for the restaurants reviewed by the specific user.

The recommendation system built using collaborative filtering provides prediction about user's interest by finding preferences from a group of users. We use Alternating Least Squares (ALS) Matrix Factorization technique for building the recommendation system.

The evaluation is done on the data set to find the error for each recommendation model. This requires parameter tuning for the feature extraction techniques, NMF and TF-IDF. Parameter tuning is done on the train and validation data sets. The final step includes finding the mean average error and base error on the test set for the different recommendation models. We create 10 samples from the test data set using bootstrap sampling on 80% of the test data. A performance comparison for all the models is done.

The next task is to tag and retrieve customer reviews with user defined tags. This is done to easily summarize and extract reviews which are talking about the user desired aspect of the restaurant. We explore the use of word2vec in combination with Latent Dirichlet Allocation (LDA) for this task.

2 RECOMMENDATION SYSTEM

Task 1 included creating a recommendation engine to recommend businesses (restaurants) to users.

The steps involved in creating the recommendation engine were:

- Dataset Generation
- Model Creation using:
  - Content Based Recommendation
  - Collaborative Filtering
- Evaluation
- Results

2.1 Data Generation

The data was available in the dataset tab on the official website of Yelp [https://www.yelp.com/dataset]. The dataset was available in two formats, json and sql. We downloaded the files in json format and converted them to csv. We used the files, business.csv, user.csv and review.csv to create our dataset.

We used these particular files since they contained:

- business.csv: business attributes, business categories, business ids and city.
- user.csv: user id, average stars
- review.csv: user id, business id, text, date, useful, cool, funny

We selected businesses filtered by cities. We decided the cities by checking the number of business reviews for each city. Las Vegas had the most business reviews, followed by Phoenix then Toronto. We not only required the number of business reviews in each city to be high, but also to be approximately same. We decided to choose the cities Charlotte, Scottsdale and Pittsburgh. Charlotte had 7557, Scottsdale had 7510 and Pittsburgh had 5688 business reviews as seen figure 1.

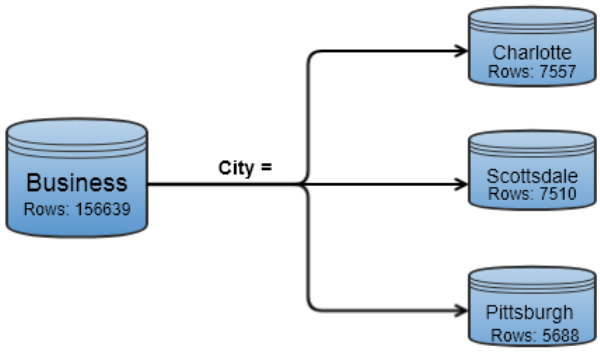

Figure 1: Cities Dataset

The business data for each city (Charlotte, Scottsdale, Pittsburgh) was then merged with review data on business ids. The resulting dataset for merged with user data on user ids. The resulting dataframe was then filtered as follows in succession:

- by categories that contained restaurants, and
- by users with at least 20 reviews

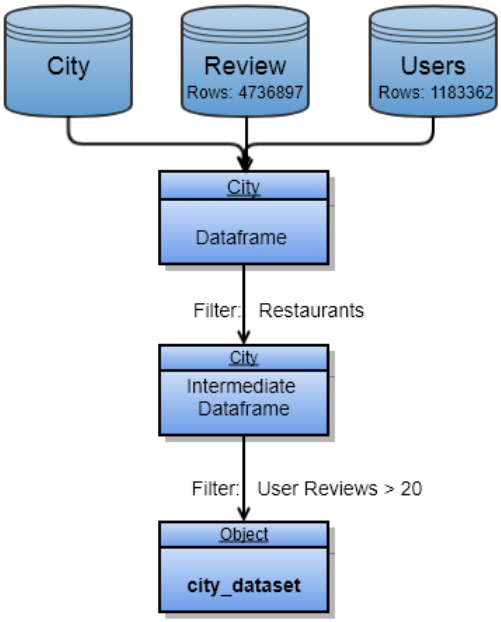

Figure 2: Final Dataset for a City

We called the resulting dataset *city_dataset* as the final dataset for a particular city as seen in figure 2. Each *city_dataset* was split into training set *city_train*, validation set *city_val* and testing set *city_test*.

The split was made with respect to date. Training set contained reviews till the year 2014 (including 2014), validation set contained reviews for the year 2015, and testing set contained reviews from 2016 as seen in figure 3.

Table 1 summarizes the number of review entries in each dataset for each city.

City	Training	Validation	Testing
Charlotte	23746	6136	11188
Pittsburgh	18570	5665	9226
Scottsdale	18387	3707	5371

Table 1: The number of reviews in each set for each city

2.2 Model Creation

Models were created to predict the reviews of a restaurant by a user, to recommend ideal restaurants to the user.

We generated multiple models using two algorithms - **Content Based Recommendation** and **Collaborative Filtering**.

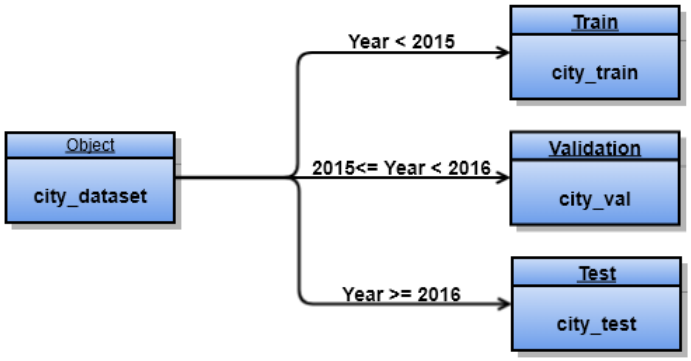

Figure 3: Training, Testing and Validation Data Split

2.2.1 Content Based Recommendation. :

Content-based recommendations analyze the content of textual information about users' preferences, and needs, and finds similarities in the content [3]. Here, it works by creating associations (similarity) with respect to either users or restaurants.

We incorporated Content Based Recommendation in two ways - User Based Similarity and Item Based Similarity. Each similarity technique was carried out using 3 feature extraction algorithms - **NMF**, **TF-IDF** and **Doc2Vec**.

These feature extraction techniques are explained in brief.

(1) Non-Negative Matrix Factorization (NMF):
Non-negative matrix factorization is a technique to decompose a matrix into two non-negative matrix factors [2].

e.g. A non-negative matrix V, is decomposed into non-negative matrix factors W and H such that:

$$V \approx WH \tag{1}$$

NMF is statistically applied in the following way:
A dataset of m multivariate n-dimensional data vectors, is placed in a n x m matrix V. It is then approximately factorized into an n x r matrix W and an r x m matrix H, where r < min(n,m) so that W and H are both smaller than the original matrix V. The resulting decomposition being two data matrices both smaller than is original matrix.

(2) Term Frequency Inverse Document Frequency (TF-IDF):
TF-IDF is a technique to calculate the importance of each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. [5]

TF-IDF can be statistically applied in the following way:
Given a document corpus D, a word w, a document d ∈ D, w_d can be calculated as follows:

$$w_d = f(w,d) * log(|D|/f(w,D)) \tag{2}$$

where, function f(w,d) calculates the frequency of w in d and $|D|$ is the size of corpus.

(3) Document to Vector (Doc2Vec):

Doc2Vec is a technique of representing text documents as a vectors. It is based on word2vec. word2vec gives a numeric representation of each word.

[1] achieved dov2vec by adding a paragraph vector to word2vec thereby accomplishing representation of sentences, paragraphs and documents.

The graphical representation of the paragraph vector can be seen in figure 4.

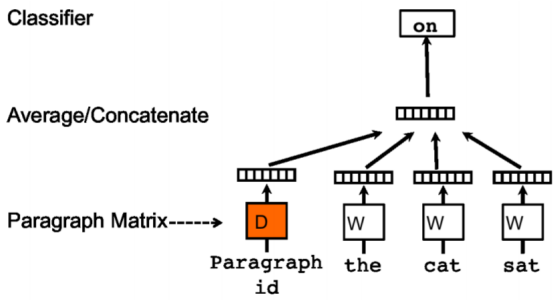

Figure 4: Graphical representation of the paragraph vector

These feature selection techniques were implemented using both **User Based Similarity** and **Item Based Similarity**.

(1) User Based Similarity -
In this approach we predicted the ratings given by a user to a restaurant (entries of test dataset) by creating a model to learn the users preferences from the training dataset.

Ratings based on User similarity were calculated using algorithm 1.

Algorithm 1 Calculate Ratings

procedure CALC_RATING($city\_test, city\_train, vector$)
  $pred\_ratings \leftarrow 0$ ▷ vector = feature extraction vector: NMF, TF-IDF or Doc2Vec
  $user\_id \leftarrow$ first user id of city_train
  $business\_id \leftarrow$ first business id of city_test
  $user\_rating\_shift \leftarrow$ calc_sim_user_rating(user_id, business_id,city_train,vector)
  $pred\_ratings \leftarrow$ user_rating_shift + (average ratings by user_id in city_train)
  return pred_ratings ▷ pred_ratings of business_id by user_id

The user based similarity model was created using the algorithm 2.

Algorithm 2 Calculate User Based Similarity

procedure CALC_SIM_USER_RATING(target_user_id, target_business_id,city_train,vector)
  $test\_user\_id \leftarrow$ all businesses in city_train where business_id = target_business_id
  $user\_reviews\_train \leftarrow$ all unique user ids in city_train with text combined with respect to corresponding user_id
  $list\_i \leftarrow$ list of indices of user_reviews_train where user_id = target_user_id
  $target\_user\_vector \leftarrow$ vector with indices in list_i
  $user\_sim \leftarrow []$ ▷ Calculating user similarity
  for user_id in test_user_id.user_id, **do**
    $list\_j \leftarrow$ indices of user_reviews_train where user_id = user_id
    $user\_vector \leftarrow$ vector with indices in list_j
    $cos\_sim \leftarrow$ cosine_similarity(target_user_vector,user_vector)
    $user\_sim \leftarrow$ append cos_sim to list user_sim
  $test\_user\_id.similarity \leftarrow$ user_sim
  $wa \leftarrow []$ ▷ Calculating weighted average
  for user_id in test_user_id.user_id, **do**
    $list\_k \leftarrow$ list of indices of test_user_id where user_id = user_id
    $test\_user\_similarity \leftarrow$ first similarity value of test_user_id.similarity filtered by indices in list_k
    $rating\_diff \leftarrow$ first ratings value of test_user_id.similarity filtered by indices in list_k - (average ratings by user_id in city_train)
    $dot\_prod \leftarrow$ dot product of test_user_similarity and rating_diff
    $sim\_sum \leftarrow$ sum(similarity values of test_user_id.similarity with indices in list_k)
    $wa \leftarrow$ append (dot_prod/sim_sum) to list wa
  return mean(wa)

(2) Item Based Similarity - -
In this approach we predicted the ratings given by a user to a restaurant (entries of test dataset) by creating a model to find restaurants that are most similar to the ones rated by user.

The item based similarity method uses an algorithm that is similar to the one used for user based similarity method. Ratings based on item similarity were calculated using algorithm 1.

Item similarity is calculated using the categories column in the business table. For each restaurant, category is a list of the keywords associated with that restaurant. Item based similarity is calculated using algorithm 2. Instead of the user reviews, we use business categories.

2.2.2 *Collaborative Filtering.* :

Collaborative filtering (CF) is a technique to make recommendations or predictions (filtering) about a user's interests by compiling preferences from a group of users (collaborating). [6]

- Alternating Least Squares (ALS) Matrix Factorization- A matrix V is factorised into two matrices W and H. The condition used in doing this is that none of the 3 matrices have non negative elements. The approximation V, given by V ≈ WH, is achieved by minimizing the error function given in equation 3.

$$min_{W,H}||V - WH||_F, \ \ Subject \ \ to \ \ W \geq 0, \ \ H \geq 0 \qquad (3)$$

2.3 Evaluation

The evaluation process was performed in two steps.

(1) Parameter Tuning -
Parameter Tuning was performed to find the ideal values of parameters to generate an optimum model that had the least mean average error. It was carried out on the validation dataset for the feature extraction techniques NMF and Doc2Vec.

Parameter tuning was executed using the algorithm 3.

Algorithm 3 Parameter Tuning

parameter_value_matrix ← [m x n] matrix of m parameters and their corresponding n values
ideal_parameter_value ← [m x 2] matrix of m parameters and their corresponding ideal values
for i in 0 to m-1, **do**
    val_mean_avg_error ← []
    for i in 0 to n-1, **do**
        vector ← feature extraction vector : NMF, Doc2Vec vector with parameter[i] and value[j]
        predict_ratings ← calc_rating(city_val,city_train,vector) ▷ From Algorithm 1
        normalized_ratings_diff ← (predict_ratings - city_val.stars_review)/city_val.stars_review
        val_mean_avg_error ← append mean(normalized_ratings_diff) to list val_mean_avg_error
    index ← index of list val_mean_avg_error where val_mean_avg_error is minimum
    ideal_parameter_value[*i*][1] ← parameter_value_matrix[i][index]
ideal_vector ← generate ideal_vector using the ideal_values of parameters

Using algorithm 3 we tuned a few parameters, the results of which are summarized in the table 2.

(2) Variance of Error
To be sure of our results, we executed algorithm 1 on 10 samples of test data for each city. Each sample was generated through bootstrap sampling on 80% of the test data.

Bootstrap sampling is a statistical technique of resampling a big sample into smaller equisized samples with replacement. We used scikit-learn's [4] sklearn.utils.resample() method to incorporate resampling in our code.

Sampling was followed by calculating mean average error and base mean average error for the 10 samples. We calculated the base mean average error for understanding whether our algorithm was performing better than the base ratings. The base ratings were obtained by calculating the average ratings given by a user in the training data.

Mean average error and base mean average error for the samples was calculated from algorithm 5. Algorithm 4 was internally used to execute algorithm 5.

Feature Extraction Technique	Parameter	Ideal Value	Definition
Doc2Vec	window	10	Maximum distance between the current and predicted word within a sentence
Doc2Vec	size	100	Dimensionality of the feature vectors
NMF	n_samples	100	Number of samples in consideration

Table 2: Parameters and Ideal Values obtained through Parameter Tuning

Algorithm 4 Calculate Average Error

procedure CALC_AVG_ERROR(*sampled_test*, *pred_ratings*)
    normalized_ratings_diff ← (pred_ratings - sampled_test[i].stars_review)/sampled_test[i].stars_review
    avg_error ← append mean(normalized_ratings_diff) to avg_error
    return *avg_error*

Algorithm 5 Mean Average Error

ideal_vector ← from Algorithm 3
mean_avg_error ← []
base_mean_avg_error ← []
for i in 1 to 10, **do**
    pred_ratings ← calc_rating(sampled_test[i],city_train,ideal_vector)
    mean_avg_error ← calc_avg_error(sampled_test[i],pred_ratings) ▷ from Algorithm 4
    pred_base_ratings ← []
    for id in indices of sampled_test[i], **do**
        user_id ← user_id of sampled_test[i] where index of sampled_test[i] = id
        pred_base_ratings ← append (average ratings by user_id in city_train) to list pred_base_ratings
    base_mean_avg_error ← calc_avg_error(sampled_test[i],pred_base_ratings) ▷ from Algorithm 4
final_mean_avg_error ← mean of the values in list mean_avg_error
final_base_mean_avg_error ← mean of the values in list base_mean_avg_error

The resulting final mean average error and base mean average error were obtained for each city.

2.4 Results

The Mean average error for user based similarity, calculated across the 3 cities and the 3 different algorithms, is shown in figure 5. This figure also shows the base error and the comparative graph for the 3 algorithms, TF-IDF, NMF and Doc2Vec. Pittsburgh provides the lowest mean average error across all algorithms.

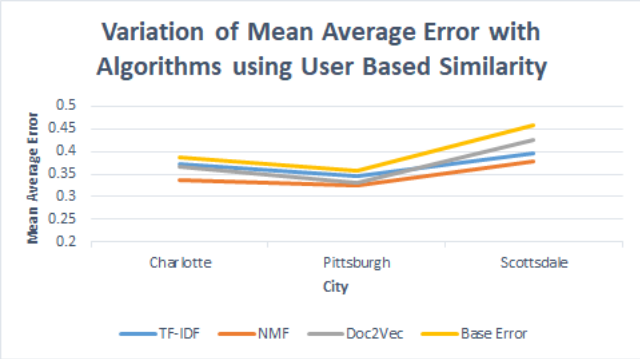

Figure 5: User Based Similarity Results

The Mean average error for item based similarity, calculated across the 3 cities and the 3 different algorithms, is shown in figure 6. This figure also shows the base error and the comparative graph for the 3 algorithms, TF-IDF, NMF and Doc2Vec. Pittsburgh has the lowest mean average error across all algorithms.

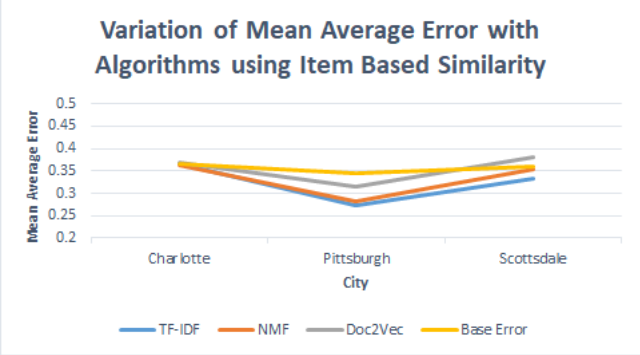

Figure 6: Item Based Similarity Results

The TF-IDF algorithm works best for item based similarity and NMF works best for user based similarity. The comparison graph is shown in figure 7.

The performace comparison for the 2 content based recommendation methods, item based and user based, and the collaborative filtering method is shown in figure 8.

3 REVIEW IDENTIFICATION FOR USER SUPPLIED WORD USING LDA AND WORD2VEC

Customer reviews on Yelp act as a significant tool for customers to research about the food quality, service, dishes etc. of a restaurant.

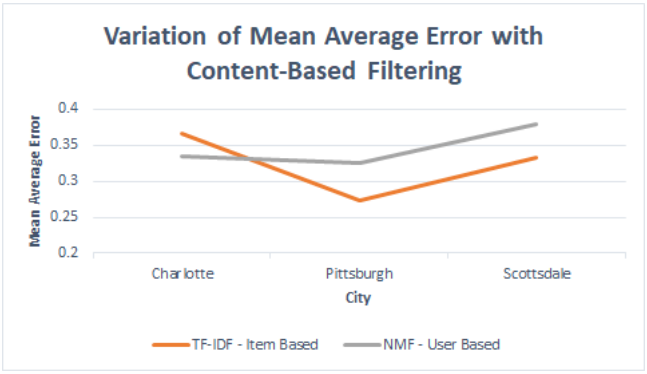

Figure 7: Content Based Recommendation Best Algorithms

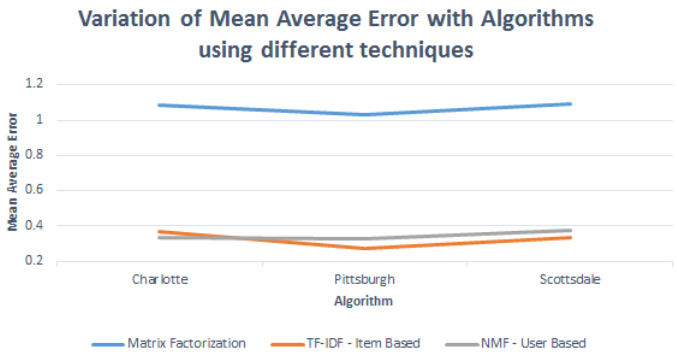

Figure 8: Performance of Different Algorithms

In order to understand restaurant quality on customer desired topics, customers have to read through individual reviews. To enable quicker understanding of individual reviews we explore the use of word2vec with LDA in order to easily tag reviews to user desired topics and extract relevant reviews.

3.1 Usage and significance

Topics generated by LDA usually required human interpretation of the word distribution of each topic. This methodology seeks to extend LDA topic extraction with word2vec to enable automated tagging of review text with user supplied tags. We use word2vec vectors to find LDA topics most similar to user supplied topic and extract reviews having high weight for these similar topics. This methodology has the following goals:

- Review topic tagging
- Quicker understanding of LDA topics
- Review retrieval for user supplied word

3.2 Background

3.2.1 LDA. In natural language processing,latent Dirichlet allocation(LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. In LDA, each document

may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. LDA is a generalisation of the pLSA model, which is equivalent to LDA under a uniform Dirichlet prior distribution.

3.2.2 Word2Vec. word2vec is widely featured as a member of the machine learning algorithms based on neural networks, commonly referred to as "deep learning" (though word2vec itself is rather shallow). Using large amounts of unannotated plain text, word2vec learns relationships between words automatically. The output are vectors, one vector per word, with remarkable linear relationships that allow us to do things like vec("king") − vec("man") + vec("woman") = vec("queen"), or vec("Montreal Canadiens") − vec("Montreal") + vec("Toronto") resembles the vector for "Toronto Maple Leafs". We seek to use this property of word2vec vectors in identifying semantically similar words to user supplied word to identify LDA topic most similar to the user supplied word.

3.3 Algorithm Details

The goal of our algorithm is to rank reviews according to their relation to the user supplied word. An overview of the steps involved are shown below:

3.3.1 Preprocessing.

- Lower case all reviews, lemmatize review text using Wordnet lemmatizer, and remove punctuation from text reviews. Create bag of words of review texts after removing stop words.
- Extract LDA topic word vector and review text LDA topic vector.
- Train word2vec vector on review text

3.3.2 Review Identification to user supplied topic.

- Calculate word2vec vector of each LDA topic by calculating the weighted average of word2vec vector for each topic with weights being the value of word importance for the topic.
- Calculate cosine similarity between word2vec representation of user supplied word and word2vec vector of each topic.
- Take cosine similarity between vector of similar topics and topic distribution of reviews to extract top relevant reviews for the user supplied word.

The complete algorithm is described as a flowchart in Fig 9.

We use word2vec to understand the similarity between user supplied topic and word distribution of LDA generated topic. Using word2vec helps by taking into account semantic similarity of user supplied word and highly weighted words in an LDA topic. In comparison, if we only use LDA word distribution for a topic to ascertain relevance of an LDA topic to the user supplied word we might end up ignoring importance of semantically similar words to user supplied word.

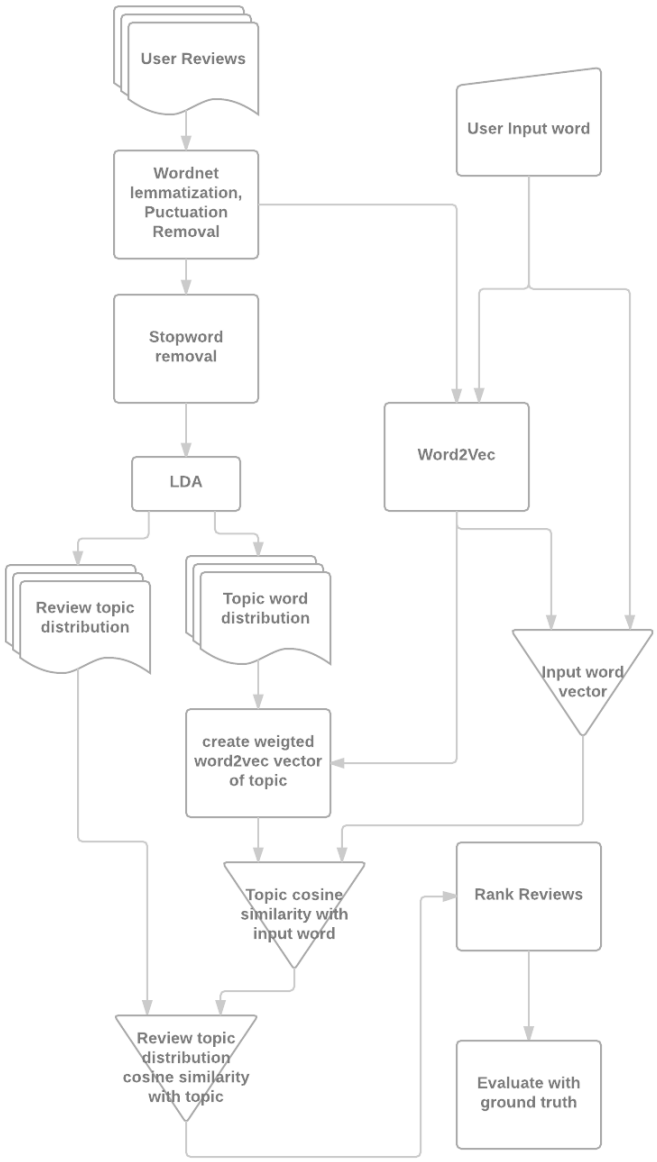

Figure 9: Algorithm Flowchart

3.4 Evaluation

For evaluation, we use reviews from 1200 Pittsburgh restaurants with varying cuisines of Yelp dataset. We also filter for reviews having greater than 50 words. Total reviews taken into consideration 45182.

3.4.1 Creating ground truth for evaluation. For creating ground truth, we use the category column to classify a review having a particular topic For example, all reviews of the restaurants having Italian or Pizza in their category classification are classified to be talking about pizza. Hence, if the user supplied word is 'pizza'. The top reviews are obtained from our algorithm are evaluated against the column having restaurant with Pizza or Italian in their category column. The user supplied word and the restaurant category classification are shown in Table 4.

3.4.2 Evaluation results. We used accuracy, precision, area under ROC curve, and recall measures to evaluate results and compare against LDA results. We obtain LDA results by using user supplied word weights as a measure of LDA topic similarity to the user supplied word. The results are shown in Table 3.

3.4.3 Discussion around evaluation results. From Table 3 we can see that combining word2vec with LDA leads to an increase in overall accuracy but lower recall and similar precision. This may be due to word2vec not being trained on enough data. Also, our ground truth might not be accurate as we are using a proxy in the form of categories column. For example, an American diner might have pizza in its categories column but a reviewer might only be talking burgers.

3.5 Conclusion and Future enhancements

Using word2vec with LDA shows some theoretical advantages of being able to use use contextual similarity of words from word2vec but this is reflected in our evaluation results when compared to just using LDA. For further research, we may need to use a more accurate proxy for evaluation and optimise parameters for word2vec and LDA.

Table 4: User supplied word and the restaurant category classification they were evaluated against

User Supplied Word	Category column classification
pizza, italian	Pizza,Italian
beverage, tea	Tea Rooms, Wineries, Wine Bars, Wine & Spirits, Pubs, Juice Bars & Smoothies, Bars, Coffee & Tea, Beer, Beer Bar, Bubble Tea, Cafes, Cocktail Bars, Coffee & Tea, Gastropubs
sweet	Waffles, Ice Cream & Frozen Yogurt, Gelato, Desserts, Creperies
beer	Wineries, Wine Bars, Wine & Spirits, Pubs, Bars, Beer, Beer Bar, Cocktail Bars, Gastropubs

REFERENCES

[1] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[2] Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.). MIT Press, 556–562. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[3] Michael J Pazzani. 1999. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review* 13, 5-6 (1999), 393–408.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[5] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. 133–142.

[6] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. 2009 (2009). https://doi.org/10.1155/2009/421425

Table 3: Evaluation results for user supplied topics

Algorithm	Metric	pizza	italian	beer	sweet	show	entertainment	Metric Average
LDA with word2vec	Accuracy	85%	85%	59%	90%	26%	21%	61%
LDA with word2vec	Recall	3%	2%	0%	8%	81%	85%	30%
LDA with word2vec	Precision	97%	95%	19%	8%	5%	5%	38%
LDA with word2vec	Area under ROC curve	79%	79%	46%	56%	56%	56%	62%
LDA	Accuracy	85%	87%	41%	79%	8%	9%	51%
LDA	Recall	4%	14%	95%	20%	96%	95%	54%
LDA	Precision	98%	95%	40%	6%	5%	5%	42%
LDA	Area under ROC curve	81%	79%	55%	51%	56%	56%	63%