Final Project Report

LDA & Word2Vec on Yelp Dataset

Abstract:

Yelp is a great source of reviews for customers, which help them choose the best business in the region. The dataset is a subset of businesses, users and review data which provides a platform to learn about various techniques like Data Mining, NLP, ML etc. to discover patterns, analyze textual data and find meaningful informations.

This project will focus on providing an overview of some widely used topic modeling and document clustering techniques. Document clustering is the application of cluster analysis of textual documents. Topic modeling techniques were developed to discover different topic dimensions of a document set. This paper will document the evaluation of two different techniques LDA and Word2Vec on yelp dataset followed by analyzing the results obtained to learn which technique is better and why.

I. Introduction:

Document clustering has always been one of the widely studied field in information retrieval or text mining areas. This technique is used to group a large document corpus which contains semantically similar textual documents into clusters when either little or no information is available about the document being analyzed. For a dataset as big as yelp, there might be a case where we might end up with too many topics in a cluster. This leads to the idea of using topic modeling along with document clustering. Topic modeling is a method to find natural groups of items which might be hidden in a dataset. LDA will be representing documents as a mixture of topics that provide words with certain probabilities. Word2Vec on the other hand is used to produce word embeddings i.e. words are mapped to the vectors. Both techniques will be used to analyze yelp dataset.

II. Techniques/Approaches:

Latent Dirichlet Allocation (LDA) : It is one of the most common topic modeling techniques. It can be understood via 2 principles. First, every document is a mixture of topics and second, Every topic is a mixture of words. So basically LDA is used to estimate both the principles at the same time. LDA will be useful in determining which words exactly belong to a specific topics and then it will determine the topic of document by then examining its probabilities. In order to work with LDA, document term matrix is required. So, First a document term matrix, which is a matrix describing the frequencies of terms that occurs in a corpora, will be calculated. In DTM, rows corresponds to documents and columns corresponds to terms.

Word2Vec : The word2vec technique is used to produce word embeddings. It takes a large corpus of text as an input and produces a vector space. Each unique word in corpus is assigned a vector in space. These word vectors are positioned in such a way that words that share common contexts in corpus are in the closer proximity to one another in the space.

Word2Vec contains two distinct models which are :

- Continuous Bag of Words model
- Skip Gram model

In CBOW, model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction. In Skip-Gram, model uses current word to predict the surrounding window of context words. Thus, CBOW is faster while Skip-Gram is slower but does better job for infrequent words.

Both the techniques mentioned above will help in extracting the latent topics from the textual data and with the help of these topics documents can be clustered accordingly. Thus, using LDA and Word2Vec for evaluation will help understand the data better.

III. Packages and Dataset:

- **Programming Language/Packages :**

Programming Language - This project will be using R & Python3.6 which provides a great environment for data mining, statistical computing and graphics. R can be easily extended by using 6000+ available packages at CRAN, thus making implementation easy. We will be using Python3.6 to perform Word2Vec because of the Gensim package.

Packages - the packages which will be used are :

Text Mining(tm), Natural Language Processing(NLP), Wordcloud, jsonlite, Latent Semantic Analysis(lsa), RTextTools(machine learning package), slam, ggplot2, SnowballC, e1071(For latent class analysis, Naive-Bayes classifier, SVM etc.), Plotly, cluster, purrr, factoextra.

Link - (**https://cran.r-project.org/web/packages/available_packages_by_name.html**)

Python - Pandas, Gensim, matplotlib, nltk, sklearn, tsne, numpy.

- **Data Source :**

Link : (https://www.yelp.com/dataset/challenge)

The dataset is obtained from 10th round of 'Yelp Dataset Challenge' and contains information about local businesses in 12 metropolitan areas across 4 countries, with around 156639 businesses and 5 million reviews from over 1.1 million users. The dataset downloaded is of 5.79 GB which contains 6 files in JSON format (user, business, review, checkin, tip, photo).

This project will be focusing on 2 out of 6 files which are business.json and review.json. Using only two files will make it somewhat easy to process on the system.

System Configuration -

OS - macOS Sierra
Processor - 2.7 GHz Intel Core i5
Memory - 8 GB 1867 MHz DDR3

- **Data Format :**

The data files were obtained in JSON format. Each file is composed of a single object type, one JSON-object per line.

Business.json contains data which includes business_id, name, location, attributes, categories and hours.

The data in JSON format -

{ "business_id": "abc", "name":"xyz", ….. , "categories":["Mexican","Burgers"], "hours":{"Monday": "10:00-21:00",…..,"Sunday":"11:00-18:00"}}

Review.json contains full review text data including user_id who wrote that review and business_id for which the review was written.

The data in JSON format -

{“review_id”:”abc”, “user_id”:”xyz”, “business_id”:“uvw”, …. , “cool”:0}

Link to the information on JSON dataset - **https://www.yelp.com/dataset/documentation/json**

IV. Experiments:

A) Data Preprocessing:

Yelp dataset is originally in JSON format which will be first converted to .csv format before preprocessing it. In order to do that, library jsonlite was used. With the help of stream_in() I was able to read JSON data, which took hours to read it. Then the JSON data was flatten into regular 2 dimension tabular structure, which was then checked if it was a data frame object using as.data.frame(). Once I was sure data is ready, business_Id and categories was extracted from business.json and business_Id and text was extracted from review.json. The data is too big to process on my laptop. So it was important to carefully decide how to reduce the data which will be used in order to get better analysis.

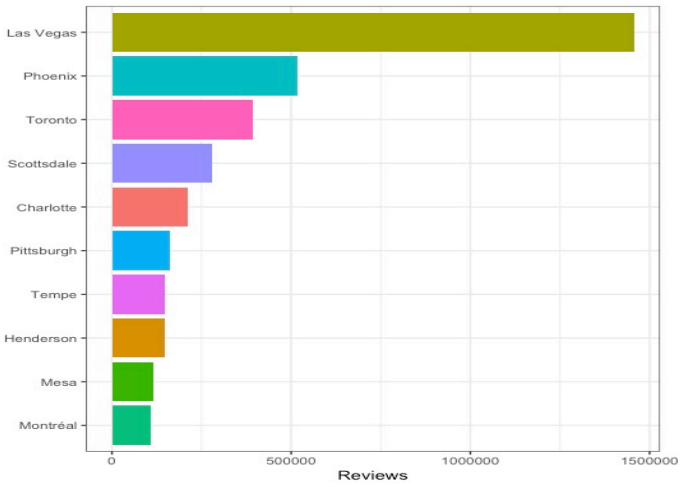


Fig-1

I decided to first know the data, figure out what I am dealing with. This led to the next step which was to find top 10 cities with highest number of reviews, which were then extracted from the dataset (Fig-1).

Based on Fig-1, I chose Las Vegas (top city with maximum number of reviews), Phoenix, and Scottsdale. Now since there are many categories, I further selected 4 out of 1240 categories which are "Restaurants","Beauty & Spas”,"Nightlife” and ”Food". I chose different categories to better group the data into clusters. I then merge the fields and grouped them by business_id. This leaves us with approximately 2 million reviews out of 5 million.

I tried using library ff to read the data but apparently reading data was not the only issue, continuing with the tasks to be performed like LDA and Word2Vec resulted into system crash if data being used increased. Since my laptop threw an error stating that the system is running out of memory after several tries, I decided to reduce the data. The following tasks were performed under data preprocessing:

- Stopwords, whitespace, numbers and punctuations were removed using the tm_map() which cleans the raw data and prepare data for further analysis.
- Document term matrix was created using the inbuilt DocumentTermMatrix(). This matrix describes the frequency of terms which occurs in a document.
  - **dtm<-DocumentTermMatrix(corp, control = list(wordLengths=c(4, Inf)))**
- Stemming was performed on the cleaned data because the classifier doesn’t understand verbs(i.e. organized & organizing) and treats them as different words.

- Pruning words by frequency: words that occurs in very few documents were removed, which is good because the entire focus is on the words which have high frequency and it also reduces computation time.
- Tf-Idf weights were used on Document Term Matrix which is one of the popular term-weighting schemes which will be used for document-term matrix. It is a statistical measure which evaluates the importance of word in the document.

B) Clustering:

Serial No.	Word	Frequency
1	place	29447
2	good	26923
3	food	24837
4	like	22211
5	great	19942
6	time	18453
7	just	18103
8	order	14932
9	realli	13579
10	servic	13230

Table 1

Fig-2

After the data is preprocessed, I decided to perform K-Means clustering so as to determine best number of clusters to base my comparison upon. Using kmeans() for different values of k, we can get various cluster plots. But to determine optimal number of clusters, NbClust will be used. The approach of using NbClust on tf-idf will determine how well the object lies within its cluster.

The following code was used to compute NbClust:

```
#nbclust
Nb <- NbClust(as.matrix(tfidf), distance = "euclidean", min.nc = 3, max.nc = 9,
method = 'complete', index = 'silhouette')
head(n,2)
```

NbClust output is:

```
> head(Nb, 2)
$All.index
     3      4      5      6      7      8      9
0.7272 0.7334 0.6358 0.6066 0.6067 0.6069 0.6071
$Best.nc
Number_clusters     Value_Index
         4.0000          0.7334
```

So according to NbClust, best number of clusters is 4. Now we have an idea what our data looks like when grouped. Based on this I will then analyze clustering in LDA and word2vec models.

C) LDA:

After preprocessing data and getting document term matrix in previous steps, we are now ready to perform LDA. In order to do so, we used Gibbs sampling for estimation. The main parameters for LDA are as follows:

- burnin - starting period, steps which does not reflect distribution property are removed.
- iter - number of iterations
- thin - number of iteration at which correlation between samples is avoided
- seed - an integer for each starting point
- nstart - number of runs at different start points
- best - return result of best run

LDA(dtm, 4, method="Gibbs", control=list(nstart= 5**, seed** = list(2003,5,63,100001,765)**, best**= TRUE**, burnin** = 4000**, iter** = 10**, thin**= 500**))**

Clustering LDA with k-means:

D) Word2Vec:

Fig-3

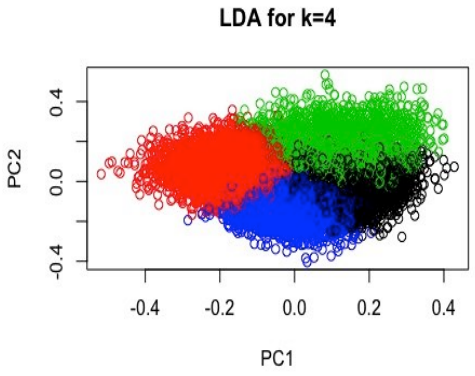
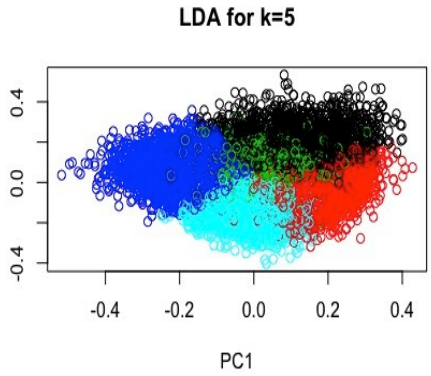


Fig-4



To perform Word2Vec, python's Gensim package is used. The data preprocessed in R was saved as a .csv file which was then used to train word2vec model using python. Initially I used both skip-gram and CBOW models when dataset was smaller in size. But as the data increased, I realized CBOW model is a better choice since it is faster than skip-gram model. Therefore, I will be using CBOW model for further analysis. The word2vec model was trained using CBOW architecture as follows:

import gensim

sg = 0 is CBOW and sg = 1 is skip gram

model=gensim.models.Word2Vec(text, min_count= 10**,window**= 5**,size**= 100**, sg = 0)**

The main parameters for Word2Vec() are as follows:

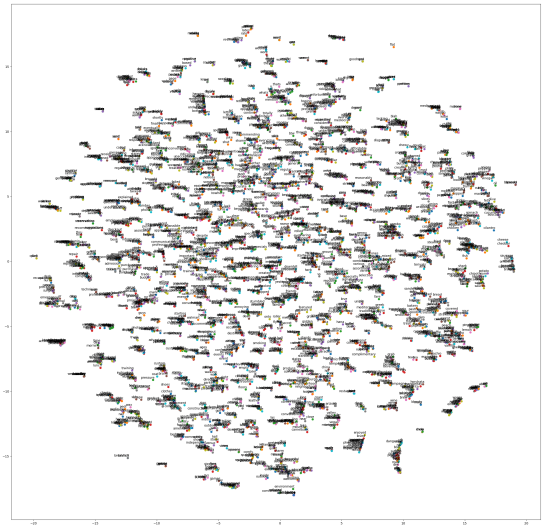
- text - is the dataset passed after preprocessing. As word2vec
- min_count - terms that occur less than min_count are ignored in the calculation. This reduces the noise and least frequent terms.
- Size - it denotes the number of dimensions which are present in the vector form of word2vec. Generally 100-200 can be assigned based upon the word2vec document.
- Sg - this defines which model will be used to evaluate word2vec. If sg=0 then CBOW model will be used and if sg=1 then Skip-Gram model will be used.
- Window - is the maximum distance between the current and predicted word within a sentence.

After the model was trained, some useful insight about the dataset can be obtained. The following summarizes how the model behaves and works:

- The word vectors are stored in keyed vectors instance in **model.wv** which provides the numpy vectors of a word. Ex: print(Model.wv['food']) will provide 'food' vectors [-0.322972 1.44059741 …… 1.451231].
- The most similar words can be found using **model.wv.most_similar**. Example: the most similar words to word sandwich are pastrami, turkey, panini, falafel, sub, gyro etc.
- In order to find similarity between two words, **model.wv.similarity** can be used. Example: similarity between sandwich and burger is 0.6477 which is quite high and true.
- To access the terms in the vocabulary matrix, **sorted(list(model.wv.vocab.keys())[:5])**is used which gives top 10 words from the vocabulary [abrupt, absenc, absolut, absurd, accept]

The entire word2vec model can be visualized by using TSNE package of sklearn.manifold. The following figure shows the visualization of trained word2vec model where nodes represent vector spaces of words.Fig-3 shows the entire visualization and image on right(Fig-4) shows a close-up view.

Fig-5



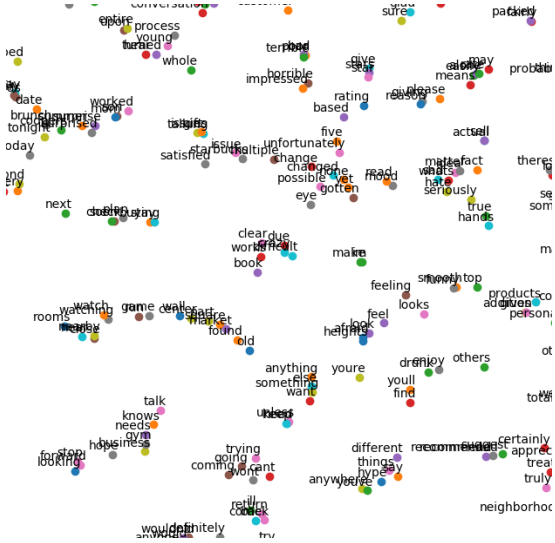


Fig-6

As we can see that word2vec does not give any topics, so I will be using clustering in order to get topics from word2vec representation. Using k-means clustering we first cluster the word vectors and then get the topics. The result is shown in Results section.

E) Parameter Tuning:

- **LDA:** For LDA we dealt with parameters mentioned above which are burnin, iter, thin, seed, nstart and best. I used default values for these parameters and the result was good. Based on K-means and keeping number of iterations as 10, I can say result obtained was good.
- **Number of Clusters:** Although I have selected 4 different categories but clustering them using k-means requires tuning the k value. Finding an optimal value of k is more like trial and error method, which can be further verified using NbClust or average silhouette method.
- **Word2Vec:** Tuning parameters was necessary in training word2vec model. Let us have a look at the code line used to train the model -

model=gensim.models.Word2Vec(text, min_count=5, window= 5, size=100, sg = 0)

Size here denotes the number of dimensions which are present in the vector form of word2vec. For a data as big as yelp, it is not possible to assume a particular value for this. Based on the documentation, ideal value to be used lies within the range of 100-200. For this project, value assigned to size is 200.
min_count is the threshold set to ignore the least terms. I set the min_count as 10, terms which occur less than 10 times will be ignored.
Window is assigned a value of 5. The terms which occur in this window size are associated together, i.e. the neighboring terms to a specific term are associated together.
Sg is used to select one of the two models based on which word2vec will be evaluated. A decision to select CBOW was already made previously.

V. Results/Evaluation:

Evaluation: In order to evaluate the the results obtained from LDA and Word2Vec, confusion matrix will be used. From confusion matrix, precision, recall, F1 score and overall accuracy will be calculated. Confusion matrix contains actual and predicted class. Based on this true positive, true negative, false positive and false negative observations are made. True positive and negative are correctly predicted values whereas false positive and negative are the values obtained when actual class contradicts with predicted values.
Precision - TP/TP+FP
Recall - TP/TP+FN
F1 Score - 2*(Recall * Precision) / (Recall + Precision)
Based on these parameters we will evaluate our models.

- **LDA:** Top 10 topics with top 10 terms is shown in the figure below.

Table 2

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	vegas	hair	good	place	like	food	time	room	great	like
2	bar	great	order	great	just	restaur	back	hotel	drink	just
3	show	job	chicken	good	realli	menu	wait	one	night	dont
4	night	massage	littl	love	place	servic	tabl	night	hour	know
5	music	work	salad	friend	dont	best	first	park	friend	want
6	las	done	tast	alway	look	price	went	stay	happi	think
7	fun	best	lunch	pizza	think	will	order	good	nice	thing
8	casino	good	also	nice	know	well	take	servic	beer	feel
9	game	servic	chees	food	thing	dinner	will	nice	enjoy	look
10	love	price	sauc	hour	want	ever	never	food	taco	make

Confusion Matrix for LDA:

Actual	Predicted			
Label	**1**	**2**	**3**	**4**
Restaurants	**4030**	**124**	**873**	**84**
Beauty & Spas	**190**	**4437**	**24**	**53**
Food	**436**	**103**	**3697**	**341**
Nightlife	**102**	**93**	**261**	**4504**

Table 3

Label	Precision	Recall	F1
Restaurants	0.7687	0.806	0.7869
Beauty & Spas	0.9371	0.8874	0.9116
Food	0.7773	0.7394	0.7578
Nightlife	0.9055	0.8912	0.8983

Table 4

Overall Accuracy = 74%

Summary :

Table-2 depicts the LDA topic modeling result for top 10 topics. From the table we can see that the top words are somewhat informative. Topic 1 may be about nightlife in Las Vegas since it does includes terms like casino, game, fun etc. Whereas topic 2 can be about a hair salon and topic 3 can be about a restaurant and topic 8 about a hotel. Although topic 10 does not really make any sense. The top 10 topics contain quite informative topics and they seem to be right since we selected only 5 categories while preprocessing. The overall accuracy of 74% so obtained from confusion matrix is good.

The confusion matrix does agree with the choice of k value selected previously. The documents from each groups are well separated from each other. Therefore, we can say that our approach was right. Table 3 and Table 4 shows confusion matrix and values for precision, recall and F1 score respectively.

- **Word2Vec:**
  - The numpy array gave an output with approximately 53489 instances and 37422 number of features. This array will now be used for further analysis. To train the model, the dataset was split into training and testing sets. The size of training set is kept much larger as compared to testing set. This numpy matrix representation is the testing set for yelp with 53489 instances and 37422 features. The training dataset is kept higher than test set in order to perform better analysis.
  - Once the word2vec model was trained, I tried to find the confusion matrix for the same. The rows represent the actual values and columns predicted values. The following is the result obtained after the word2vec model was trained.
  - Obtaining the confusion matrix will ultimately help us evaluate the results.

Summary:

Table 5 shows top 5 topics extracted from word2vec model. Topic 1 may be about a nail salons or hair salons or it can be both. Topic 2 can be about a pizza place whereas topic 4 sort of hint towards the staff service. The confusion matrix thus obtained for k-means clustering on word2vec for k = 4, agrees with our first clustering experiment. It shows that NbClust provided an optimal value for k. Although we selected only 4 categories for this dataset, our results agreed with what we expected.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	nail	pizza	would	great	absolut
2	just	cream	time	place	just
3	dont	time	car	good	drink
4	realli	wait	servic	food	place
5	color	ice	back	always	dont
6	much	tabl	wait	friend	liquor
7	think	bread	get	staff	think
8	shav	came	driv	servic	expens
9	pretti	good	Custom	best	Beer
10	trim	cheese	work	love	jazz

Table 5

Actual	Predicted			
Label	**1**	**2**	**3**	**4**
Restaurants	**3513**	**169**	**1442**	**219**
Beauty & Spas	**115**	**4912**	**73**	**89**
Food	**1510**	**98**	**4178**	**871**
Nightlife	**341**	**129**	**349**	**3971**

Table 6

Label	Precision	Recall	F1
Restaurants	0.6325	0.6409	0.5651
Beauty & Spas	0.9352	0.9200	0.9116
Food	0.6164	0.6903	0.597
Nightlife	0.8117	0.7654	0.7439

Table 7

Table 6 shows confusion matrix for the word2vec model and Table 7 shows corresponding values for precision, recall and F1 score. The overall accuracy obtained by word2vec model was of 78% which is good enough.

VI. Analysis:

- The dataset used for the analysis was yelp dataset which is bulky and with different characteristics. The raw data was preprocessed at the initial stage.
- Pruning words by frequency i.e. removing words which occur less frequently is a better way to prepare the data for analysis. This will reduce the size of data and will make it easier for computation.
- The top 10 topics obtained from LDA shows that the topics actually did make some sense. Terms in the topics indicated that they belonged to one or the other category which were selected at the beginning.
- Finding an optimal value for k was a challenge, which was then verified by using NbClust.
- LDA did perform well on the dataset and represented topics in a fairly better way. Clusters of LDA were distinct and accuracy was better.
- Word2vec was trained and confusion matrix was formed. The overall accuracy for word2vec model was good. From my point of view a difference of 4% is not that big of a difference as

yelp data is quite diverse and if this analysis is performed on a stronger machine, it will be interesting to see how does the model behave when the entire dataset is considered.

- Although word2vec model does not provide topics, with the help of python's k-means function I was able to extract the topics.
- The most important work was to label the data, without which getting confusion matrix was not possible.
- During the entire procedure, upon comparing the influence of simplest of procedures like pruning, labeling, tf-idf weighting, formation of document-term matrix etc. provided a deeper understanding of the concepts.
- Although the topics I discovered may not be very clear but they seemed reasonable.

VII. Challenges/Learnings:

- Yelp dataset is bulky, so loading entire dataset, converting it to .csv format and reducing the data took time and effort.
- Once dataset was obtained, due to its large volume it became practically impossible to train the models and get the result as system keeps crashing.
- Parameter tuning for word2vec model was the main challenge as once I fixed the size I had to tune the parameters accordingly. Since the size of dataset was increased gradually, it was challenging to figure out right values for parameters to train the models. Also, the idea of using default values for the parameters definitely did not work well in word2vec case.
- Clustering word2vec model again was a challenge, initially I was not getting result because somehow labels were not assigned. After figuring out the error, confusion matrix was obtained. Also writing a proper algorithm to perform k-means took time.
- I started with only review.json and performed LDA and Word2Vec, but later on when it came to evaluating the result it was not possible because the data was not labeled. Due to this I had to again start from the scratch and merge categories while merging business_Id and text.
- Doing the entire project by myself, working on the dataset from the scratch helped me learn how to preprocess the dataset, get data ready to perform analysis, tune parameters as per needs and analyze what I see from the results obtained. Apart from this, I get to learn two widely used techniques, LDA and Word2Vec. Doing research on them made me familiar with the concept and logic and working on them made me familiar with their practical implementation.
- The project provided me with a good opportunity to understand and apply data mining algorithms with a hand on experience using both python and R language.