A Capstone Project Report on

SENTIMENT ANALYSIS OF CUSTOMER REVIEWS: SUMMARIZING THE CUSTOMER REVIEWS TO HELP CUSTOMERS CHOOSE A CASUAL DINING SEGMENT

Submitted to

Praxis Business School, Kolkata

(in fulfillment of the requirements for the award of the degree)

Post Graduate Program

in

Data Science

by

Akshay Tularam Bansod (**A19008**)
Hitesh Manojkumar Relwani (**A19013**)
Pratik Chakravorty (**A19023**)
Satendra Nath (**A19027**)
Sreetama Das (**A19037**)
Vivek Kumar (**A19041**)

Under the guidance of
Prof. Dr. Subhasis Dasgupta

Academic Year: 2019 - 20

Acknowledgement

We are profoundly grateful to **Prof. Dr. Subhasis Dasgupta** for his expert guidance and continuous encouragement through out to see that this project meets its target since its commencement until completion.

We would like to express deepest appreciation towards **Prof. Dr. Prithwis Mukerjee** and **Prof. Charanpreet Singh**, Founders and Directors, Praxis Business School, **Prof. Dr. Sourav Saha**, and **Prof. Dr. Gourab Nath**, whose invaluable guidance supported us in completing this project.

Lastly, We would like to express our sincere heartfelt gratitude to all the staff members of Data Science Department who helped us directly or indirectly during this course of work.

Akshay Bansod
Hitesh Relwani
Pratik Chakravorty
Satendra Nath
Sreetama Das
Vivek Kumar

Date: 31 March 2020

Contents

Abstract

With the advancement over time, the online rating and reviewing experience is becoming remarkable for both customers and restaurant owners alike. For customers it is proving to be important because product information is more valuable for services than goods as they appear riskier. Therefore, restaurant customers are more likely to seek external information sources when they have not experienced it themselves. For owners it is extremely important for them to know about the areas of improvement so that they can overcome the gaps and the areas which are doing well so that they can market them as their highlights in different channels. In this research, we intend to extend some of the existing works on sentiment analysis and suggest an effective way to cluster customer reviews for better and more informative summarization. Our work will help the owners know about the important features and the associated sentiment scores for those features and will help the customers know about the top recommended restaurants based on the requirements which they search. We have restricted this project to dealing with only casual dining segment.

Chapter 1

Introduction

During the last few years, reviews have become crucial to the success of a restaurant, as every restaurant owner is aware of the fact that good reviews can boost popularity and profitability, whereas terrible reviews even have the potential of closing businesses down. That's why it is crucial for restauranteurs to understand the impact of review websites such as Zomato and Yelp and the role they play in the success or downfall of a business.

In a recent research report published by the experts at Website Builder, approximately 61 percent of customers have read online reviews about restaurants. While such a significant number people tend to read online reviews prior to visiting a particular restaurant to dine or hosting an event, it is also worth pointing out that around 34 percent of diners currently choose restaurants based solely on information offered on peer review websites. This means that most diners disregard the restaurant's website or social media pages, preferring to rely on data present on review sites, further increasing their importance and influence on the market. Another interesting fact is that approximately 53 percent of the coveted 18 to 34-year-old demographic reported that online reviews play an important role into their dining decisions.

Fortunately, reviews can work to a restaurant's advantage as an eatery that manages to improve its rating even by one star on the scale of one to five, has a higher potential of being completely booked during the peak dining times.

Chapter 2

Methodology

2.1 Data Collection

The dataset used is "Zomato Bangalore Restaurants". In that we have only focused on the casual dining segment. The casual dining segment has around 40K rows satisfying our requirements. No data as such has been used to train the Symspell algorithm(which will be explained later) as it is already pre-trained. Restaurant data of 8 restaurants has been used for doing the project pipeline which starts with Symspell and followed by lemmatization, deep segmentation, feature extraction, dependency parsing and sentiment score respectively. For training the Non-negative matrix factorization algorithm, the dataset comprising 40K restaurant reviews has been used.
The restaurant reviews used in the project pipeline have been used in the UI in the form of restaurant names.

2.2 Symspell

It can be used for spelling correction and fuzzy string search. SymSpell derives its speed from the Symmetric Delete spelling correction algorithm and keeps its memory requirement in check by prefix indexing.

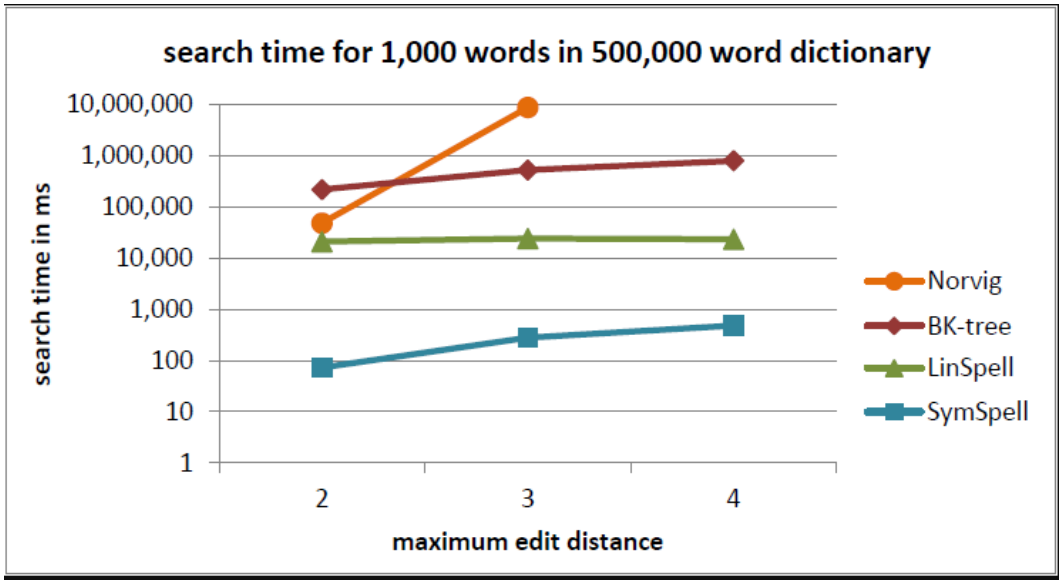
Lookup provides a very fast spelling correction of single words.

- A Verbosity parameter allows to control the number of returned results:
  - Top: Top suggestion with the highest term frequency of the suggestions of smallest edit distance found.
  - Closest: All suggestions of smallest edit distance found, suggestions ordered by term frequency.

  - All: All suggestions within maxEditDistance, suggestions ordered by edit distance, then by term frequency.

- The Maximum edit distance parameter controls up to which edit distance words from the dictionary should be treated as suggestions.
- The required Word frequency dictionary can either be directly loaded from text files (LoadDictionary) or generated from a large text corpus (CreateDictionary).

Applications include spelling correction, query correction (10–15 percent of queries contain misspelled terms),chatbots,OCR post-processing, automated proofreading and fuzzy search and approximate string matching Performance is 0.033 milliseconds/word (edit distance 2) and 0.180 milliseconds/word (edit distance 3) (single core on 2012 Macbook Pro)



LookupCompound supports compound aware automatic spelling correction of multi-word input strings.

1. Compound splitting and decompounding

Lookup() assumes every input string as single term. LookupCompound also supports compound splitting / decompounding with three cases:

- mistakenly inserted space within a correct word led to two incorrect terms
- mistakenly omitted space between two correct words led to one incorrect combined term
- multiple input terms with/without spelling errors

Splitting errors, concatenation errors, substitution errors, transposition errors, deletion errors and insertion errors can be mixed within the same word.

2. Automatic spelling correction

- Large document collections make manual correction infeasible and require unsupervised, fully-automatic spelling correction.
- In conventional spelling correction of a single token, the user is presented with multiple spelling correction suggestions.

Examples:

```
- whereis th elove hehad dated forImuch of thepast who cougdn'tread in
sixthgrade and ins pired him
+ where is the love he had dated for much of the past who couldn't read in
sixth grade and inspired him  (9 edits)

- in te dhird qarter oflast jear he hadlearned ofca sekretplan
+ in the third quarter of last year he had learned of a secret plan  (9 edits)

- the bigiest playrs in te strogsommer film slatew ith plety of funn
+ the biggest players in the strong summer film slate with plenty of fun  (9
edits)

- Can yu readthis messa ge despite thehorible sppelingmsitakes
+ can you read this message despite the horrible spelling mistakes  (9 edits)
```

WordSegmentation divides a string into words by inserting missing spaces at appropriate positions.

- Misspelled words are corrected and do not prevent segmentation.
- Existing spaces are allowed and considered for optimum segmentation.
- SymSpell.WordSegmentation uses a Triangular Matrix approach instead of the conventional Dynamic Programming: It uses an array instead of a dictionary for memoization, loops instead of recursion and incrementally optimizes prefix strings instead of remainder strings.
- The Triangular Matrix approach is faster than the Dynamic Programming approach. It has a lower memory consumption, better scaling (constant O(1) memory consumption vs. linear O(n)) and is GC friendly. SymSpell.WordSegmentation has a linear runtime O(n) to find the optimum composition.

```
- thequickbrownfoxjumpsoverthelazydog
+ the quick brown fox jumps over the lazy dog

- itwasabrightcolddayinaprilandtheclockswerestrikingthirteen
+ it was a bright cold day in april and the clocks were striking thirteen

-
itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolish
ness
+ it was the best of times it was the worst of times it was the age of wisdom
it was the age of foolishness
```

Applications:

- Word Segmentation for CJK languages for Indexing Spelling correction, Machine translation, Language understanding, Sentiment analysis
- Normalizing English compound nouns for search and indexing (e.g. ice box = ice-box = icebox; pig sty = pig-sty = pigsty)
- Word segmentation for compounds if both original word and split word parts should be indexed.
- Correction of missing spaces caused by Typing errors.
- Correction of Conversion errors: spaces between word may get lost e.g. when removing line breaks.
- Correction of OCR errors: inferior quality of original documents or handwritten text may prevent that all spaces are recognized.
- Correction of Transmission errors: during the transmission over noisy channels spaces can get lost or spelling errors introduced.
- Keyword extraction from URL addresses, domain names, hashtags, table column descriptions or programming variables written without spaces.
- For password analysis, the extraction of terms from passwords can be required.
- For Speech recognition, if spaces between words are not properly recognized in spoken language.
- Automatic CamelCasing of programming variables.

- Applications beyond Natural Language processing, e.g. segmenting DNA sequence into words

Performance:
4 milliseconds for segmenting an 185 char string into 53 words (single core on 2012 Macbook Pro)

We have used Symspell, which is a language independent spell corrector, because it would be difficult to do further analysis in the form of extraction of noun phrases and their associated adjectives and adverbs if we do not get corrected words. This would further lead to problems in calculation of sentiment scores down the pipeline, hence it was needed. Moreover, the entire process is automated and pre-trained on huge collection of english words for it to perform effectively. We also had to create a dictionary of indian dishes for them to go corrected when passed through symspell. It was possible to implement symspell correctly in our scenario with the usage of unigrams.
An example from our implementation:

```
In [37]: review = 'The food was gr8 but specially the biryni was amazing , the staff was courtious and ambience was wonderful'
         slang_corr = slang_trans(review)
         corr_string = spell_checker_eng_word(slang_corr)
         print("Incorrect English:",review)
         print("Corrected English:",corr_string)

         Incorrect English: The food was gr8 but specially the biryni was amazing , the staff was courtious and ambience was wonderful
         Corrected English: the food was great but specially the biryani was amazing , the staff was courteous and ambience was wonderfu
         l
```

2.3 Lemmatization

It involves resolving words to their dictionary form. In fact, a lemma of a word is its dictionary or canonical form.

Because lemmatization is more nuanced in this respect, it requires a little more to actually make work. For lemmatization to resolve a word to its lemma, it needs to know its part of speech. That requires extra computational linguistics power such as a part of speech tagger. This allows it to do better resolutions (like resolving is and are to “be”)

To get the best results we have to feed the part of speech tags to the lemmatizer, otherwise it might not reduce all the words to the lemmas which we desire. It is based on the WordNet database.We have used lemmatizer present in nltk.

Our code for for the same is as follows:

```python
def lemmatize_sentence(sentence):
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    wn_tagged = map(lambda x: (x[0], nltk2wn_tag(x[1])), nltk_tagged)
    res_words = []
    for word, tag in wn_tagged:
        if tag is None:
            res_words.append(word)
        else:
            res_words.append(lemmatizer.lemmatize(word, tag))
    return " ".join(res_words)
```

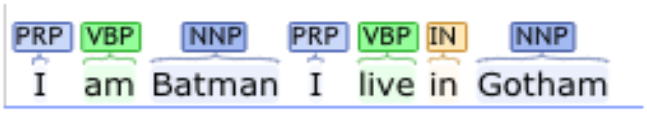
2.4 Deep Segment

There are various libraries including some of the most popular ones like NLTK, Spacy, Stanford CoreNLP that that provide excellent, easy to use functions for sentence segmentation.These libraries work exactly as they are supposed to i.e: work near perfectly on perfectly formatted text and fail miserably on text with bad punctuations, wrong capitalisations etc.

Sample example:

```
>>> nltk.sent_tokenize('I am Batman I live in Gotham')
['I am Batman I live in Gotham']
>>> doc = nlp('I am Batman I live in Gotham')
>>> print(list(doc.sents))
[I am Batman I live in Gotham]
>>>
```

NLTK and Spacy fails (to no one's surprise)



and so does CoreNLP

A simple logic is used that combines a random number of sentences from the corpus and removes or changes the punctation and casing of the text to generate the text.To implement this, sequence to sequence models or sequence tagging models for the task at hand is used i.e: Perform sentence segmentation on the unpunctuated text and use seq2seq for punctuation correction at a sentence level instead of on the whole text.

Since the idea behind this is to get a baseline model as quickly as possible, it was started with Glove + BiLSTM CRF implementation.

Absolute Accuracy	DeepSegment	Spacy	NLTK punkt
Correct punctuation	97.139	96.039	98.789
Partial punctuation	71.096	23.477	19.379
No punctuation	52.637	11.758	9.89
Average	73.35	43.4	42.5

Comparison of absolute accuracy

A screenshot of deep segment implemented by us is below:

```
Input : a very nice and good place with best buffet in town The staff be very humble and polite
Output: ['a very nice and good place with best buffet in town', 'The staff be very humble and polite']
```

2.5 Feature Extraction

Chunking is the first step towards information extraction from unstructured text. It basically means extracting what is a real world entity from the text (Person, Organization, Event etc). We use it to map it against a knowledge base to understand what the sentence is about or intending to extract relationships between different named entities (like who works where, when the event takes place etc).Chunking can be reduced to a tagging problem.

Text chunking, also referred to as shallow parsing, is a task that follows Part-Of-Speech Tagging and that adds more structure to the sentence.The result is a grouping of the words in "chunks".The most obvious advantage of shallow parsing is that it's an easier task and a shallow parser can be more accurate. Also, working with chunks is way easier than working with full-blown parse trees.

spaCy(which we have used) automatically detects noun phrases.The chunker computes the root of the phrase, the main word of the phrase.For example:

```
doc = nlp("Wall Street Journal just published an interesting piece on crypto currencies")
for chunk in doc.noun_chunks:
    print(chunk.text, chunk.label_, chunk.root.text)

# Wall Street Journal NP Journal
# an interesting piece NP piece
# crypto currencies NP currencies

```

An example of chunking performed by us:

```
Input : the food is good and the biryani is awesome
Noun Chunks : [the food, the biryani]
Cleaned Chunks : ['food', 'biryani']
```

The process goes further with extraction of top noun chunks and their associated frequencies as shown below in the code below:

```
# Counter is used to get key,value pair for each review.
# Only those features are considered which appears in more than 5% of reviews

for value in list(rest_dict.keys()):

    freq = Counter(rest_dict[value])
    no_of_review = df[df.restaurant == value].shape[0]
    most_common={key:[] for (key,val) in dict(freq).items() if (val/no_of_review)>0.05}
```

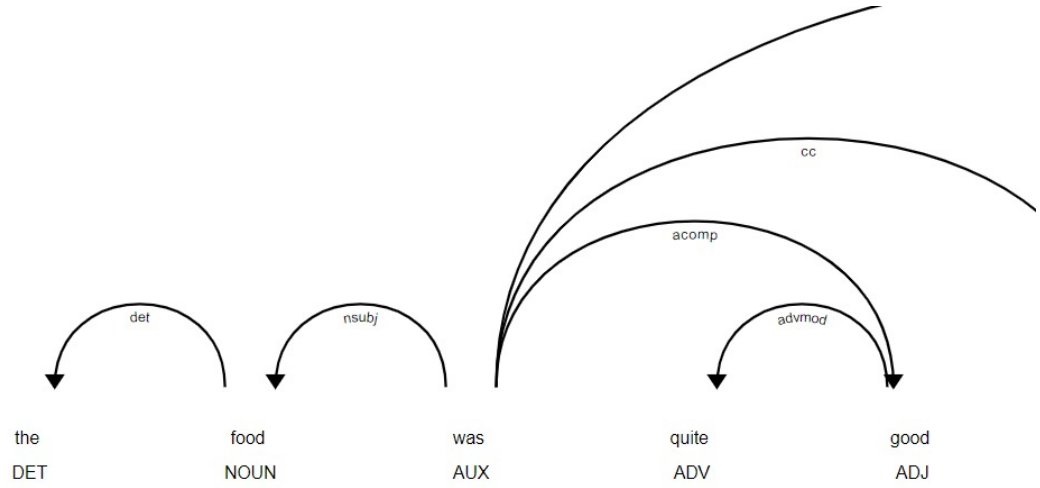
The output of the code shown above is as follows:

```
[('main course', 6),
 ('starters', 5),
 ('options', 4),
 ('staff', 4),
 ('desserts', 4),
 ('food', 4),
 ('quality', 3),
 ('service', 3),
 ('place', 3),
```

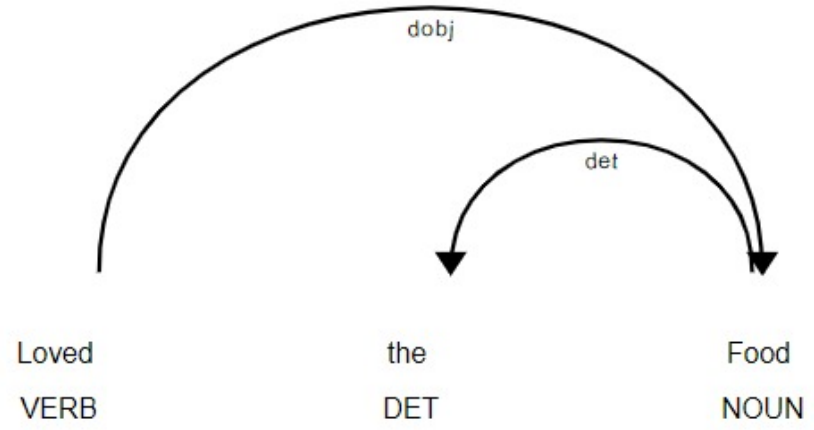
2.6 Dependency Parsing

Dependency parsing is the process of extracting the dependency parse of a sentence to represent its grammatical structure. It defines the dependency relationship between headwords and their dependents. The head of a sentence has no dependency and is called the root of the sentence.We are doing this so that we get proper association of the noun/noun phrases with the adverbs and/or adjectives(whether it be positive or negative), which would further enable us to calculate the sentiment score of the important words associated with a restaurant.

An example of right side dependency parsing is as below:



An example of left side dependency parsing is as below:



We found out the adjectives associated with the features extracted as follows:

```
{'staff': [' courteous', ' extremely polite', ' very polite', 'friendly'],
 'main course': ['avoided',
  'not acceptable',
  ' disappointing',
  'not that good'],
 'options': ['courteous', 'consider', 'nice', 'many'],
 'service': ['courteous',
  ' very courteous',
  'Absolutely loved',
  ' upto',
  ' disappointing'],
```

2.7 Sentiment Score

The AFINN lexicon is perhaps one of the simplest and most popular lexicons that can be used extensively for sentiment analysis. The current version of the lexicon is AFINN-en-165. txt and it contains over 3,300+ words with a polarity score associated with each word.

After retrieving the adjectives associated with the nouns, the mean sentiment score associated with the important features associated with a particular restaurant was dumped in the excel file as follows:

Restaurant	Word	Mean	Freq	-5.0	-4.0	-3.0	-2.0	-1.0	1.0	2.0	3.0	4.0	5.0
Barbeque Nation	place	2.75	80	0	0	1	0	1	2	13	56	7	0
Barbeque Nation	taste	1.22	18	0	0	1	5	0	0	3	7	2	0
Barbeque Nation	order	2	3	0	0	0	0	0	1	1	1	0	0
Barbeque Nation	main course	0.6	5	0	0	0	2	0	1	0	2	0	0
Barbeque Nation	service	1.09	47	0	0	7	9	0	0	3	28	0	0
Barbeque Nation	ambiance	2.5	16	0	0	0	0	0	1	6	9	0	0
Barbeque Nation	restaurant	2.15	13	0	0	1	0	0	0	5	7	0	0

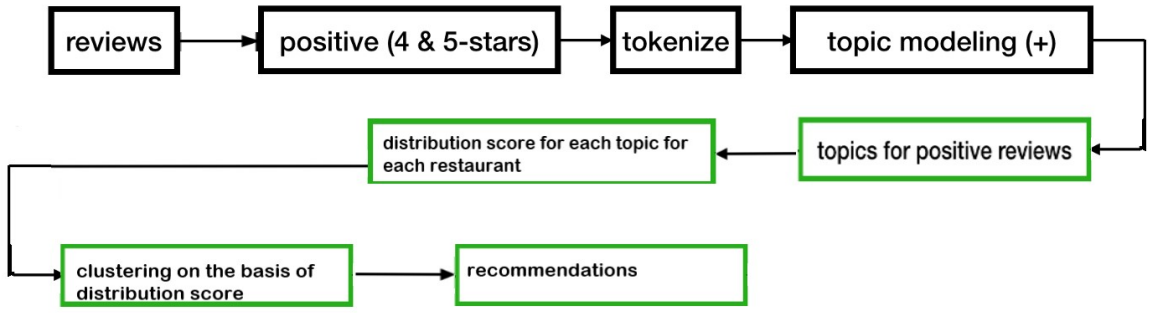
For the UI created, we have selected 8 restaurants whose insights would be shown in visual forms with the process starting with symspell and leading all the way to sentiment score calculation. The integration of this pipeline with the UI has been made possible with the usage of the python application flask.

2.8 Topic Modeling

Topic modeling is the process of identifying topics in a set of documents. This can be useful for search engines, customer service automation, and any other instance where knowing the topics of documents is important.

This section aims to analyze restaurant reviews via topic modeling to determine main topics for positive and negative reviews. Once each review was assigned its topic weights, that information was averaged to assign topic weights for each restaurants.

The flow chart for the same is as follows:



TF-IDF:

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

Documents (restaurant reviews) were tokenized using TFIDF and 1-grams, which gave the most consistent results and distinct topics.

LDA:

The 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. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics.

We thought LDA could be a useful technique since it assumes that each document (yelp review, in this case) is a mixture of a small number of topics. However, our LDA topics had lots of overlap in meaning among each other, which could be the result of relatively short document size.

Non-negative matrix factorization:

NMF (Nonnegative Matrix Factorization) is a matrix factorization method where we constrain the matrices to be nonnegative. In order to understand NMF, we should clarify the underlying intuition between matrix factorization

Suppose we factorize a matrix X into two matrices W and H so that X WH, there is no guarantee that we can recover the original matrix, so we will approximate it as best as we can.

Now, suppose that X is composed of m rows x_1,x_2,....x_m , W is composed of k rows w_1,w_2,...w_k , H is composed of m rows h_1,h_2,...h_m. Each row in X can be considered a data point. For instance, in the case of decomposing images, each row in X is a single image, and each column represents some feature.

NMF gave the most interpretable results and separable topics,for example:

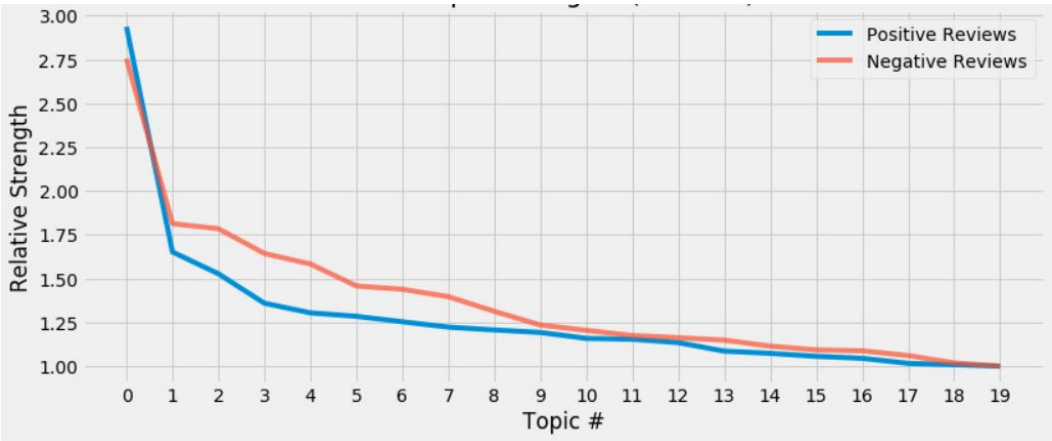
- 3.364*biryani
- 0.393*mutton
- 0.317*chicken

- 0.275*best
- 0.264*dum

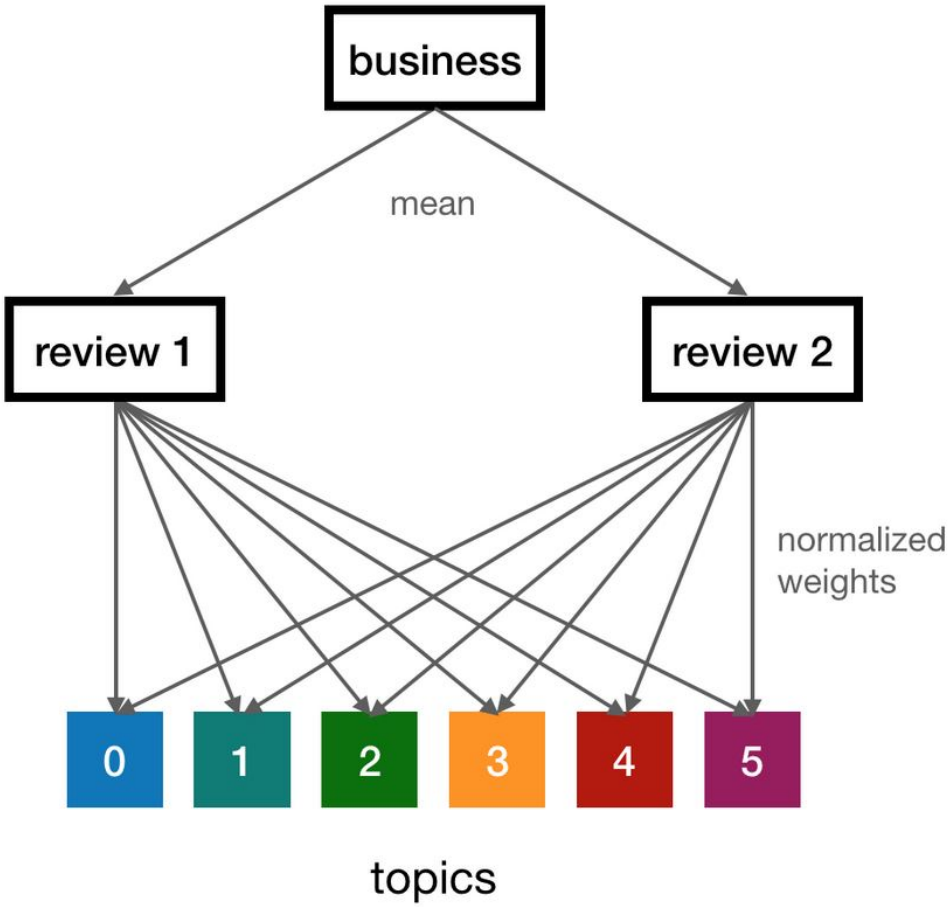
```
1 no_topics = num_topics
2 no_top_words = 5
3 print('Topics for POSITIVE reviews')
4 print('-'*39)
5 display_topics(nmf_pos, tfidf_pos.get_feature_names(), no_topics, no_top_words)
```

```
Topics for POSITIVE reviews
---------------------------------------
Topic 0:   3.467*chicken, 1.218*biriyani, 1.136*order, 0.871*rice, 0.720*tasty
Topic 1:   3.364*biryani, 0.393*mutton, 0.317*chicken, 0.275*best, 0.264*dum
Topic 2:   3.883*nice, 0.155*ambience, 0.143*tasty, 0.114*friends, 0.107*staff
Topic 3:   1.842*service, 1.540*great, 1.281*ambience, 0.932*staff, 0.884*visit
Topic 4:   2.723*taste, 1.912*awesome, 0.737*quality, 0.668*quantity, 0.521*delivery
Topic 5:   2.275*veg, 1.258*non, 0.534*starters, 0.365*buffet, 0.324*main
```

With those word weights shown above, we can assume that this topic had something to do with a restaurant's Biryani and its happy hour specials. Therefore, rest of the analysis was done using NMF. The chart below shows the elbow plot of relative topic strengths which was used to settle on 6 topics for the rest of the project.



In the figure above we are seeing distribution of topics among restaurants.



The diagram above is an illustrated overview of few steps:

- Mapping all reviews to topics using Non-negative Matrix Factorization (NMF).
- Normalizing topic weights to sum to 1 to increase interpretability.
- Averaging the topic distributions for all reviews of a restaurant to map that restaurant to the topic space.

Now that all restaurants had been mapped to the topic space, we can do four things:

- Use cosine similarity to find restaurants similar to each other based not on attributes or definitions, but its user reviews. This is analogous to getting restaurants that elicit a similar sentiment from reviewers.
- Recommending restaurant based on the distribution of review not on attributes or definitions. for example, If 5 restaurants have high distribution score for biryani, so we can recommend those restaurants to public for biryani.

- Visually show to a restaurant owner what topics their positive and negative reviews reside in. This makes it clear if a restaurant needs to improve its service quality, or if its much more popular for its pizza than its bar.
- Take a mean of all the restaurants and create an "average restaurant" in particular city or area. This will allow comparison of any restaurant against this calculated mean.

For now we are moving forwards with on the first two bullet points above i.e. finding similar restaurants and recommending restaurants.The last two bullet points are kept in the future scope of this project.

Chapter 3

Conclusion and Future Scope

3.1 Conclusion

In this report we have proposed a technique to analyze the customer reviews of casual dining restaurants and based on the sentiments derived have put up a suggestion in the User Interface which would enable customers to search similar restaurants based on different attributes such as Similar Restaurants, Ambience, Service and Menu. We have used Symspell library for grammatical correction of the reviews and Spacy to create noun chunks. Then by using spaCy dependency parsing we have found adjectives associated with the common features derived and the associated sentiment scores with the AFINN library.

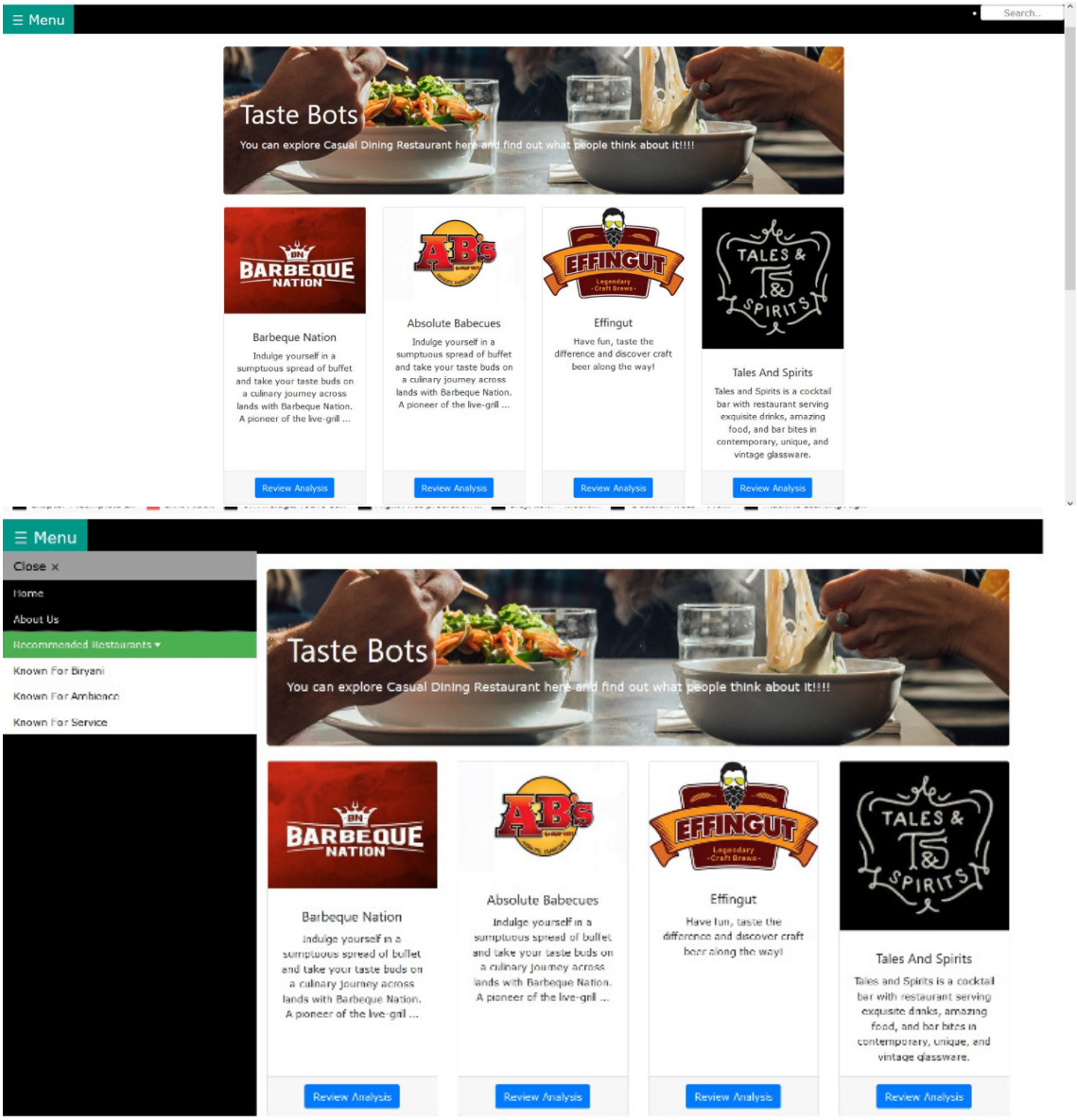
3.2 Future Scope

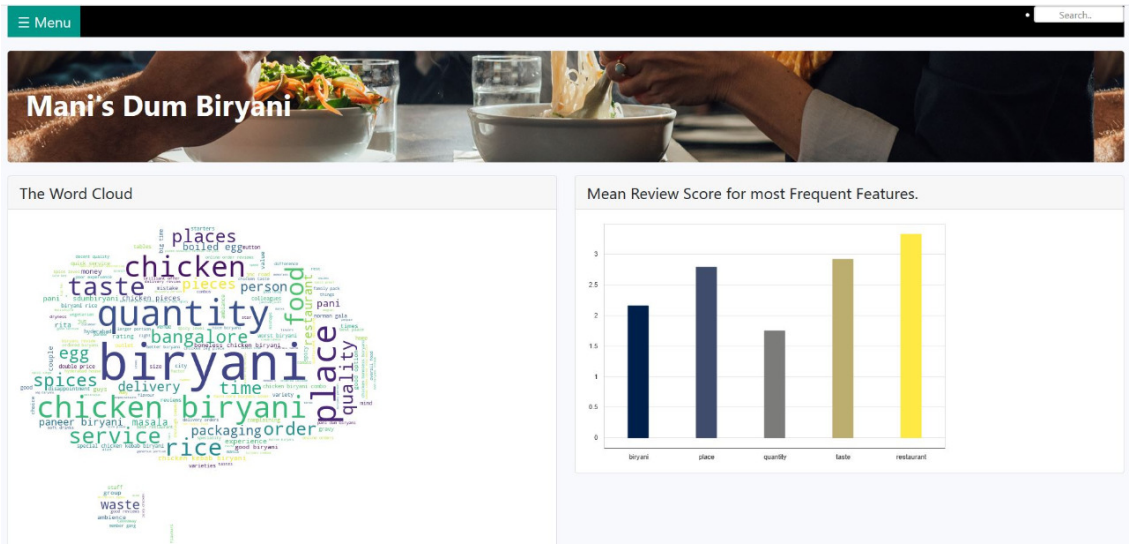
In this we have used Zomato dataset and stored the file in local systems and after preprocessing we have analyzed the data from the system.

But in the future we plan to create a pipeline that would scrap the data from multiple APIs, simultaneously would preprocess the data and henceforth would reflect the analysis on the website.It would be a dynamic website.

Chapter 4

UI Snapshots




☰ Menu
Search..
Mani's Dum Biryani
The Word Cloud
places
boiled egg
chicken
taste
pieces
person
food
restaurant
quantity
bangalore
biryani
place
quality
egg
spices
delivery
time
chicken
biryani
paneer biryani masala
packaging
order
service
rice
waste
Mean Review Score for most Frequent Features.
biryani
place
quantity
taste
restaurant


The Positives
restaurant
3.33
The Negatives
time
1.42
experience
0.5
order
0.0
Sentiment Distribution of Features
Score Distribution for biryani
Score Distribution for place
Score Distribution for quantity
Score Distribution for taste

Bibliography

[1] Nath, Gourab, Randeep Ghosh, Retail Risk, and Rishav Nath. "Cluster Analysis of Customer Reviews: Summarizing Customer Reviews to Help Manufacturers Identify Customer Satisfaction Level."

[2] Hu, Minqing & Liu, Bing, "Mining Opinion Features in Customer Reviews," Proceedings of AAAI, 2004

[3] https://github.com/wolfgarbe/SymSpell

[4] https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8

[5] https://medium.com/@praneethbedapudi/deepcorrection-1-sentence-segmentation-of-unpunctuated-text-a1dbc0db4e98

[6] https://nlpforhackers.io/complete-guide-to-spacy/

[7] https://nlpforhackers.io/text-chunking/

[8] https://www.datacamp.com/community/tutorials/introduction-t-sne