Extract Salient Users Reviews Using NLP

Siyi Ge

May 5, 2021

Abstract

Reviews will efficiently reflect customers' preferences and complaints about a business, but it becomes crucial to extract useful information from tons of reviews due to the information overload problem. To deal with this situation, I will describe how to use NLP to preprocess a restaurant review dataset; build a predictive model that predicts the sentiments of each review; then compare the accuracies of those models with different algorithms by a confusion matrix. Lastly, I will evaluate the sentence score by calculating the TFIDF and extracting the top 10 sentences to form a summarization.

The main scope of the report is to help the business owner efficiently obtain important information about the restaurant from the customer reviews.

Introduction

We have already entered the age of the Internet and technology, where everyone is sharing their thoughts, ratings, and reviews online. We don't need to know everything by experiencing it in person. Checking out customer reviews is a convenient and effective way for both customers and businesses to find out what other customers have already been to that place and how they think of that place. However, as more and more people do reviews, we often may not be able to get the critical information we want quickly and accurately; therefore, we want to use Natural Language Processing (NLP) methods to extract the essential reviews, analyze the sentiment and summarize the content of a large number of reviews.

This report aims to analyze the restaurant's customer reviews to illustrate the strengths and weaknesses of this restaurant to improve the overall quality of products and services. This report also provides a detailed view of the process and procedures taken to analyze the data.

Related Work

There is a considerable and growing number of research on natural language texts and document summarization. We might notice extensive previous work focusing on sentence selection (e.g., (Radev,2004; Haghighi and Vanderwende, 2009)), sentiment analysis, and multi-document summarization a different purpose. Briefly speaking, there is a general application of Sentiment Analysis and Prediction due to the development of online social media platforms. The essential work in Natural Language Process (NLP) is to make the computer understand the importance of different words in sentences and then the grammar and structure.

The most trending research in this field would be summarization, or even multi-document summarization, which considers more on the coherence and salience of sentences, for example, Google Transformers. As for Google Transformers, some of the API focus on text generation with GPT-2 and summarization with BART. (e.g., Wolf et al.) More than what previous work has done, G-FLOW combinates the joint problems of sentence selection (e.g., (Radev,2004; Haghighi and Vanderwende, 2009)) and sentence ordering (e.g., (Lapata, 2003; Barzilay and Lapata, 2008)). For the coherent purpose, G-FLOW builds the joint model to produce more accurate summaries.

Methods

This report will include two main Experiment Sections: Part 1 Sentiment Analysis, and Part 2 Extract Salient User Reviews.

In Part 1, for Sentiment Analysis, the methods are focusing on how to let the computer learn the sentiment of different words. We use various approaches to improve the performance of sentiment analysis, including:

Feature Selection

This mechanism goes after the data preprocessing part. We use the package "CountVectorizer" to realize the feature selection. This package is designed to tokenize a collection of words and then convert them into a matrix. (e.g., (Brownlee, 2020)). This step is necessary if we are planning to build up a machine learning pipeline. Using this package, we convert the feature of reviews into a matrix of integers. This matrix of integers is the library we build up to storage and represent the feature of each reviews. For more details of this "library", we'll discuss in the next method "Bag of Words".

Bag of Words

The most important step in this analysis is Bag of Words, which is a typical natural language processing approach. Combining feather selection, we take all the different words of reviews in the dataset without repeated words and then convert into a sparse matrix, to form a "library". In this matrix, each one column is for each one word and each row is for a single review. For example, if the word "good" appears once in Review 1 (e.g., Row 1 for Review 1),

there will count once under the Column for the word “good”. The example matrix is as follow:

finish	fri	great	love	place	wow	way
			1	1	1	
	1	1				
1		2				1

In Part 2, for text summarization, we aim to efficiently extracting the core meaning of a text document. Simply speaking, we still need to create the frequency matrix of the words from the text documents, and beyond that, we rank the relevancy (for example, importance and frequency) of the words and sentences and form a summary by sentence scores. There are manifold metrics for ranking the sentences in different research. In this report, we mainly use the following approach:

TF-IDF

This metric requires to import the library: “TfidVectorizer”. This library mainly helps to explain relevance of word to a text.

The program below using TF-IDF the following function:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

$$idf(w) = \log \frac{N}{df_t}$$

$$w_{i,j} = tf_{i.j} * log \frac{N}{df_i}$$

Problem Definition

Sometimes the issue of over reliance to the internet is not safe as we already entered the age of the Internet and technology where everyone is sharing their thoughts, ratings and reviews online the hacking and non-security of the person information becomes untrustworthy to the owner as any person from the public can get the information. This is so-called information overload.

Due to the information overload problem, it's quite necessary to extract most important information from a huge number of reviews or comments.

Purpose

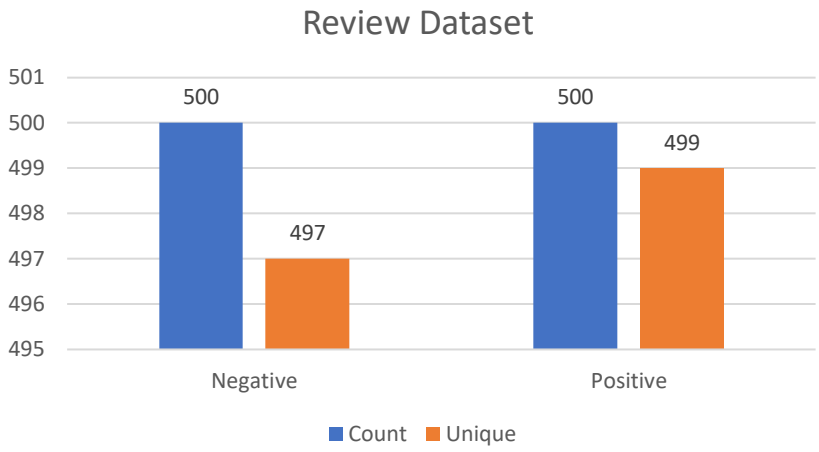
- NLP Analysis: using NLP to clean and preprocess a restaurant review dataset, and by splitting and training the review data to build a model that could predict the sentiments (positive or negative) of each review; then using the confusion matrix to test the accuracy of the model with different machine learning algorithms (random forest, multinomial, Bernoulli, logistic).
- Text Summarization: we want to evaluate the sentences score by calculating the TFIDF and make a summarization by select those sentences with highest average score.

Experiments

There are totally two main part in this report: Part 1Sentiment Analysis, Part 2 Text Summarization. Within above two parts, there are several steps used to complete the two procedures.

Part 1 Sentiment Analysis

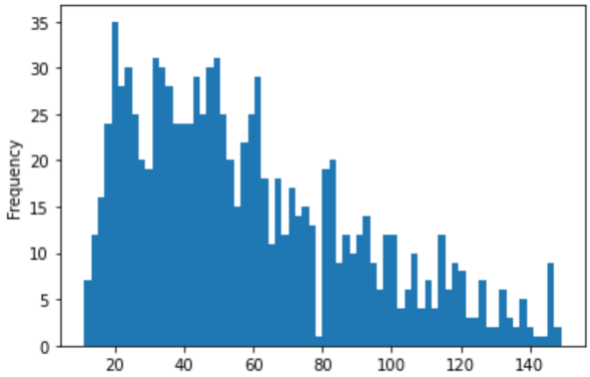
Step 1 Data Analysis



The dataset I use is from Kaggle which consist of 1000 reviews on a restaurant. Five hundred are positive reviews, and 500 are negative reviews. Most of them are unique.

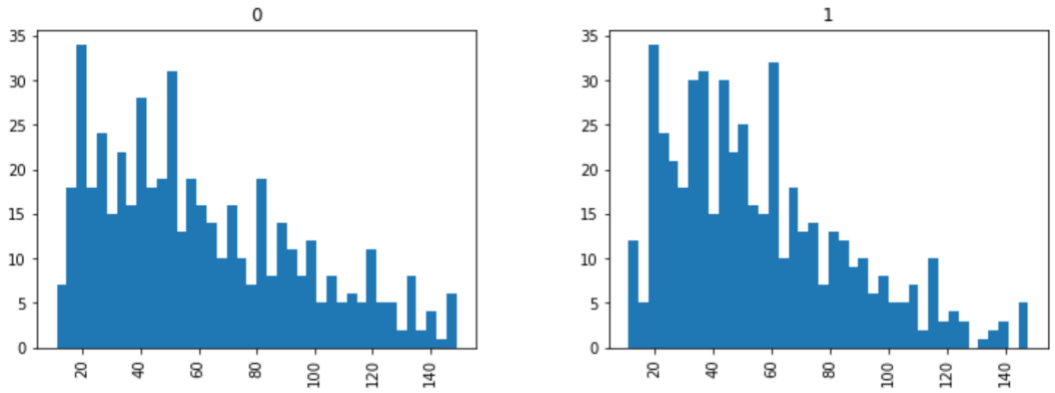
	Review	Liked	Length
0	Wow... Loved this place.	1	24
1	Crust is not good.	0	18
2	Not tasty and the texture was just nasty.	0	41
3	Stopped by during the late May bank holiday of...	1	87
4	The selection on the menu was great and so wer...	1	59
...	...	...	...
995	I think food should have flavor and texture an...	0	66
996	Appetite instantly gone.	0	24
997	Overall I was not impressed and would not go b...	0	50
998	The whole experience was underwhelming, and I ...	0	91

Sample reviews from the dataset with their sentiment 0 and 1 and length.



This histogram is created for the Length column where the number of discrete intervals is 70. The average review length of this dataset is around 58, the shortest review has 11 words, and the longest review is about 149 words long. the most frequent size is about 20 words, and most reviews are from 20 terms to 80 terms.

Below are two histograms created for both positive and negative sentiments. The positive is '1' while the negative is '0'.



Step 1.1 Word Cloud

Use word cloud to find out what are customers mainly concern about this restaurant.

Above are word cloud for Negative Reviews

Above are word cloud for Positive Reviews

These word clouds created after simple text cleaning; we could find out some negative terms such as ‘don’t’ ‘bad’ ‘never’ and ‘disappointed’ and some positive terms like ‘good’ ‘great’ ‘delicious’ ‘amazing’, and both word cloud has significant show people talks about the place, service, and food.

Step 2 TEXT CLEANING OR PREPROCESSING

In the data preprocessing part, by removing stop words, we remove the low-level information from our text to focus on the critical data. We remove Punctuations, Numbers, and special characters since they will increase the size of the bag of words and decrease the efficiency.

We also want to do stemming, which takes roots of the words; and we want to convert all letters to lower case.

Sample Before Text cleaning:

```
                                Wow... Loved this place.
                                      Crust is not good.
          Not tasty and the texture was just nasty.
Stopped by during the late May bank holiday of...
The selection on the menu was great and so wer...
   Now I am getting angry and I want my damn pho.
             Honeslty it didn't taste THAT fresh.)
The potatoes were like rubber and you could te...
                           The fries were great too.
```

Results After Text Cleaning:

```
'wow love place',
'crust good',
'tasti textur nasti',
'stop late may bank holiday rick steve recommend love',
'select menu great price',
'get angri want damn pho',
'honeslti tast fresh',
'potato like rubber could tell made ahead time kept warmer',
'fri great',
```

Some main Codes I use for Text Cleaning with their function:

- **Review** = re.sub ('[^a-zA-z]',' ', dataset ['Review'] [i]) this cleans the data.
- **Review** = review.lower () this converts the text into lower case.
- **Review** = review. Split () this aims to split the dataset using the separator as space.
- **ps** = PorterStemmer() this is as a constructor which is called by ps
- **Review** = [ps.stem (word) for word in review if not word in set (stopwords.words ('english'))] this loops the stemming words in the array.
- **Review** = ' '. join (review) this rejoins the string in the array elements.

- **corpus.append (review)** the append function appends the strings in order to create a new array of the clean text.

Step 3 TOKENIZATION:

Tokenization is splitting lines, sentences and words from the text body.

A Simple Example:

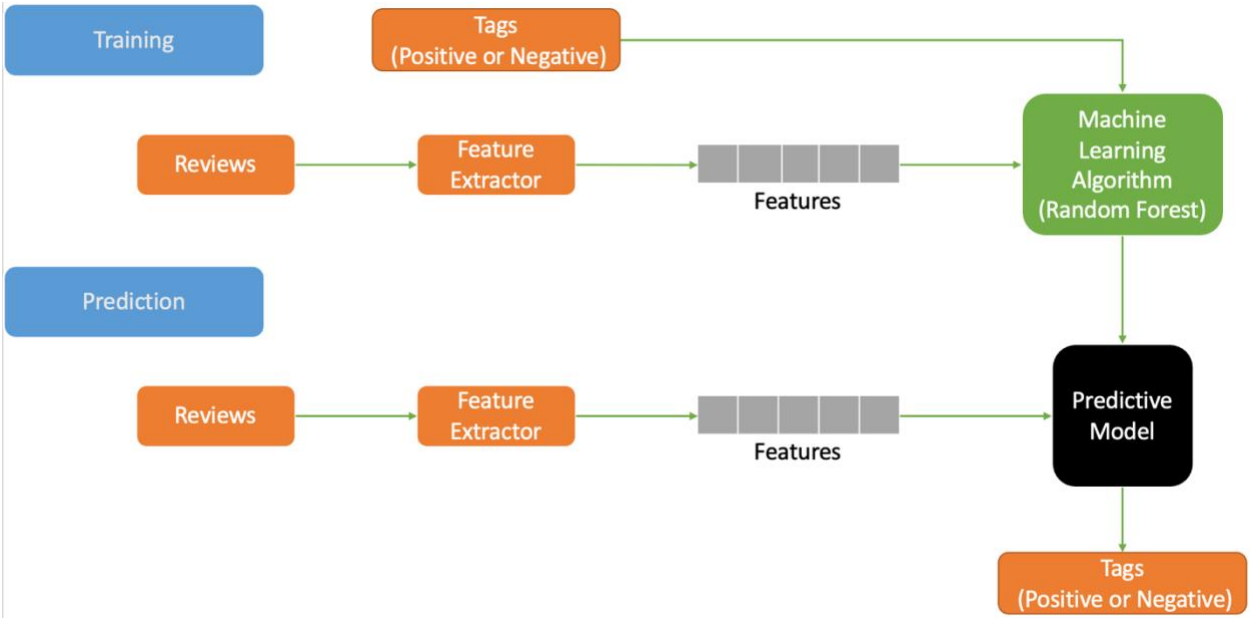
```
print(sent_tokenize(corpus[149]))
print(word_tokenize(corpus[149]))
```

```
['watch waiter pay lot attent tabl ignor us']
['watch', 'waiter', 'pay', 'lot', 'attent', 'tabl', 'ignor', 'us']
```

Step 4 MAKING THE BAG OF WORDS

For feature selection, we want to first take all the different words of reviews in the dataset without repeating words, and to make the bag of words, we want each column for each word and each row for each review. If one word appears once in a review, then it will be counted as one under the column of the word.

Step 5 SPLITTING TRAINING AND TESTING SET



This step involves splitting the dataset into Training set and Testing set. The Training set is used to train the model while the Testing set is used to test the properties of the model. The dataset is split, and the test size is 30% for testing while the rest 70% is for training.

In the training process, we take our review data as an input, and its corresponding output tags, which indicates positive and negative, after the feature extractor transfers the review text input into a feature vector. Pairs of feature vectors and tags are fed into the machine learning algorithm to generate a model.

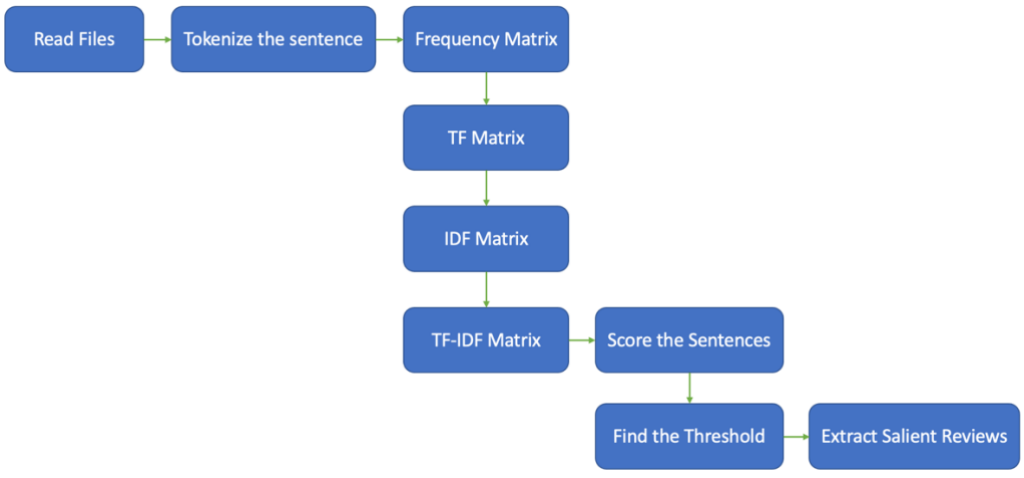
Step 5.1 FITTING A PREDICTIVE MODEL

In the prediction process, the feature extractor is also used to transform text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags about positive and negative information.

Part 2 Extract Salient User Reviews

In this part, we use the mechanism what we discussed in the Related Work and Methods, which is TF-IDF.

Pipeline:



Step 1 TF Matrix

We firstly count the frequency of each word in each review, then we calculate the term frequency of each word.

```
def _create_tf_matrix(freq_matrix):
    tf_matrix = {}

    for sent, f_table in freq_matrix.items():
        tf_table = {}

        count_words_in_sentence = len(f_table)
        for word, count in f_table.items():
            tf_table[word] = count / count_words_in_sentence

        tf_matrix[sent] = tf_table

    return tf_matrix
```

Step 2 IDF Matrix

Before calculating the IDF Matrix, we need to count how many times a word appears in the whole document; then we follow the equation to get the log of the total number of reviews divided by the count of each term.

```
def _create_idf_matrix(freq_matrix, count_doc_per_words, total_documents):
    idf_matrix = {}

    for sent, f_table in freq_matrix.items():
        idf_table = {}

        for word in f_table.keys():
            idf_table[word] = math.log10(total_documents / float(count_doc_per_words[word]))

        idf_matrix[sent] = idf_table

    return idf_matrix
```

Step 3 TFIDF Matrix

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

Step 4 Calculate Sentence Score

By calculating the average TFIDF score of all words in a review, we could get the sentence score.

Step 4.1 Calculate Threshold

We also calculate the average score of all sentences in the document; we called that average score our threshold.

Step 5 Generate Salient User Reviews

Select a sentence for a summarization if the sentence score is more than the threshold times k; and by time k with the threshold, as K gets bigger, we will get a shorter summary with more salient reviews.

Results and Analysis

Sample Result for the Predictive Model:

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

The following summarizes the accuracy of previous approaches on Experiment Part 1 Sentiment Analysis:

We mainly use two formula here to calculate Precision and Recall.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Random Forest

```
Confusion Matrix:
 [[131  21]
 [ 59  89]]


Accuracy is  73.33 %
Precision is  0.81
Recall is  0.6
```

Multinomial Naïve Bayes

```
Confusion Matrix:
 [[119  33]
 [ 34 114]]


Accuracy is  77.67 %
Precision is  0.78
Recall is  0.77
```

Bernoulli Naive Bayes

```
Confusion Matrix:
 [[115  37]
 [ 32 116]]


Accuracy is  77.0 %
Precision is  0.76
Recall is  0.78
```

Logistic Regression

```
Confusion Matrix:
 [[125  27]
 [ 43 105]]


Accuracy is  76.67 %
Precision is  0.8
Recall is  0.71
```

Results for TFIDF:

```
{'Wow...': {'wow': 1.2618732334057823, '...': 0.7847519786861198},
 'Loved this plac': {'love': 0.38093661670289114,
  'thi': 0.22862697271411966,
  'place': 0.25464912212291463,
  '.': 0.02887662537492873},
 'Crust is not go': {'crust': 0.8999459086224153,
  'good': 0.35714959860564477,
  '.': 0.038502167166571635},
 'Not tasty and t': {'tasti': 0.40017354430624535,
  'textur': 0.5047492933623129,
  'wa': 0.12341047111015913,
  'nasti': 0.5047492933623129,
  '.': 0.023101300299942985},
 'Stopped by duri': {'stop': 0.21807343001847862,
  'dure': 0.209263428835928,
  'late': 0.24543979326065873,
  'may': 0.209263428835928,
  'bank': 0.2728061565028388,
  'holiday': 0.2728061565028388,
```

Results for Sentence Score:

```
'Wow...': 1.023312606045951,
'Loved this plac': 0.22327233422871354,
'Crust is not go': 0.43186589146487725,
'Not tasty and t': 0.31123678048819464,
'Stopped by duri': 0.207784400692334,
'The selection o': 0.21099462779546863,
'Now I am gettin': 0.31379532766314394,
'Honeslty it did': 0.26440803864224527,
'The potatoes we': 0.1808858114574137,
'The fries were ': 0.20634293011179136,
'A great touch.': 0.4520026579879986,
'Service was ver': 0.24358173133605984,
'Would not go ba': 0.25768483057512137,
'The cashier had': 0.22524797708364894,
'I tried the Cap': 0.2180327529051731,
'I was disgusted': 0.2422925793857991,
'I was shocked b': 0.24993478294544802,
'Highly recommen': 0.4790068856198091,
'Waitress was a ': 0.22656588559949128,
```

GENERATE THE SUMMARY:

ALGORITHM: Select a sentence for a summarization if the sentence score is more than the average score (threshold).

Code:

```
def _generate_summary(sentences, sentenceValue, threshold):
    sentence_count = 0
    summary = ''

    for sentence in sentences:
        if sentence[:15] in sentenceValue and sentenceValue[sentence[:15]] >= (threshold):
            summary += " " + sentence
            sentence_count += 1

    return summary
```

OUTPUT FOR THE SUMMARY:

Threshold * 2:

```
" Wow... The Burrittos Blah! I could care less... The interior is just beautiful. So they per
formed. Omelets are to die for! Sooooo good!! Check it out. I'll leave it at that... Penne vo
dka excellent! Don't do it!!!! Avoid at all cost! DELICIOUS!! Interesting decor. Extremely Ta
sty! Service stinks here! I waited and waited. The burger... They dropped more than the ball.
No complaints! We loved the biscuits!!! So absolutley fantastic. I LOVED it! And then tragedy
struck. Thoroughly disappointed! Just spicy enough.. Eew... Waited and waited and waited. Ove
r rated. I will not return. Very disappointing!!! What a mistake. Eclectic selection. A fanta
stic neighborhood gem !!! 2 Thumbs Up!! A FLY was in my apple juice.. A FLY!!!!!!!! The nacho
s are a MUST HAVE! The chips and salsa were really good, the salsa was very fresh. This place
is great!!!!!!!!!!!!!! It was packed!! Do not waste your money here! The chips and sals a her
e is amazing!!!!!!!!!!!!!!!!!!!"
```

Threshold * 5

```
' The chips and salsa were really good, the salsa was very fresh. The chips and sals a here i
s amazing!!!!!!!!!!!!!!!!!!'
```

For different purpose, we could adjust k with threshold to get the most proper result.

Conclusion

This project is using NLP to clean and preprocess a restaurant review dataset, and by splitting and training the review data to build a model that could predict the sentiments of each reviews; then using the confusion matrix to test the accuracy of the model with different machine learning algorithms. Lastly, we want to evaluate the sentences score by calculating the TFIDF and make a summarization by select those sentences with highest average score.

By doing sentiment analysis and extracting salient user reviews, we want to help businesses and customers to make decisions and improve efficiency.

References

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. (n.d.). Transformers: State-of-the-art natural language processing. Retrieved April 23, 2021, from https://www.aclweb.org/anthology/2020.emnlp-demos.6/

Brownlee, J. (2020, June 27). How to encode text data for machine learning with scikit-learn. Retrieved April 28, 2021, from https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

il.ion(2014, January 29). A simple java class for tf*idf scoring [Web log post]. Retrieved from http://filotechnologia.blogspot.com/2014/01/a-simple-java-class-for-tfidf-scoring.html

Generating word cloud in python. (2020, April 18). Retrieved May 03, 2021, from https://www.geeksforgeeks.org/generating-word-cloud-python/

Jagtap, R. (2020, June 11). Abstractive text summarization using transformers. Retrieved May 03, 2021, from https://medium.com/swlh/abstractive-text-summarization-using-transformers-3e774cc42453

Panchal, A. (2021, February 08). Text summarization using tf-idf. Retrieved May 03, 2021, from https://towardsdatascience.com/text-summarization-using-tf-idf-e64a0644ace3

Malik, F. (2019, July 04). Nlp: How to evaluate the model performance. Retrieved May 03, 2021, from https://medium.com/fintechexplained/nlp-how-to-evaluate-the-model-performance-7e1820cdb759