Report for Topic Extraction and Classification

-Ipshita Ghosh

Introduction

When numerous reviews are received, it becomes imperative to classify the reviews into specific tags. Tagging helps the company to understand which issues need more attention. This task can be done using machine learning algorithms; various algorithms like Latent Dirichlet Allocation can be used. In deep learning, topic modelling using BERT is one example.

In this project, we will try to extract topics from a document and using those topics; we will build a classifier.

Approach

The approach for the task is as follows,

1. **Set Up** → get all the libraries and data needed in a workable format
2. **Feature Extraction** → Extract helpful information to understand the data
3. **Data Cleaning** → Clean the data to make it workable with models efficiently
4. **Vectorizing** → Make the dataset numeric
5. **Exploration** → Get insights from the data
6. **Model 1** → Assign the topics
7. **Model 2** → Train a classifier to classify the topics

In each step, we will also see the motivation of doing a specific task, followed by the shortcomings of the particular step.

Setup

We will import the required libraries, load data, and clean the data to help us forward tasks in the setup section.

An overview of the dataset

1. The dataset consists of one column, containing the reviews
2. The dataset has no missing value
3. The dataset has 10131 rows of reviews

Feature Extraction

Feature extraction is used to extract helpful features from a document. Extracting helpful features helps us in gaining more insights into the data. One important thing to note is, the features are extracted before cleaning since cleaning and standardizing will delete many features. The features extracted are:

1. Number of words in a given review
2. Number of characters in a given review
3. Average word length of a review
4. Number of stop words used
5. Number of numeric characters present
6. Number of words, which are in UPPER CASE
7. The polarity of the review from -1(negative) to 1(positive)

In reviews, it is seen that longer reviews are usually negative, and short reviews are positive. UPPER case words show some emotion and should be taken into account. Similarly, the numeric characters can explain whether the reviews are about a date or order ids.

Data Cleaning

The data cleaning process assures that the data is in standard form so that it helps in better processing. In the data cleaning section, we have done the following:

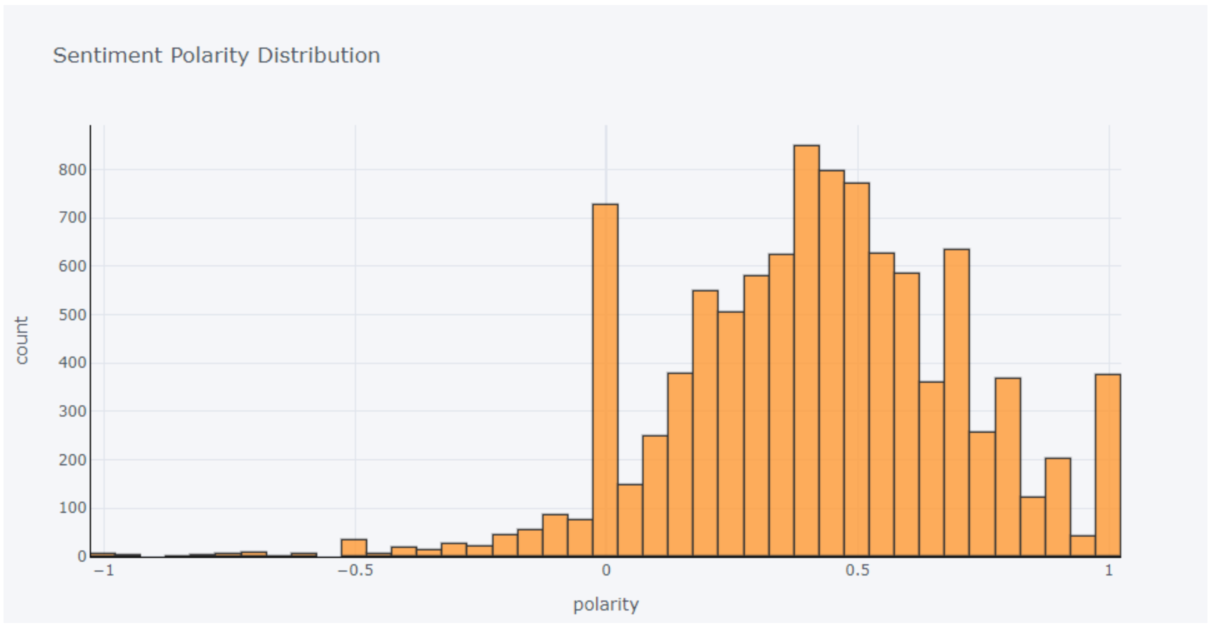
1. change all words to lower case
2. expand certain contractions
3. remove punctuations and special characters
4. remove extra spaces and trailing spaces
5. remove accented characters
6. remove stop words
7. change the words to their base form

Vectorizing

Vectorizing is a process to convert string data to numeric data. We have done count vectorization because the model used is LDA, which has TF-IDF built in it. This is the motivation for using a count vectorizer.

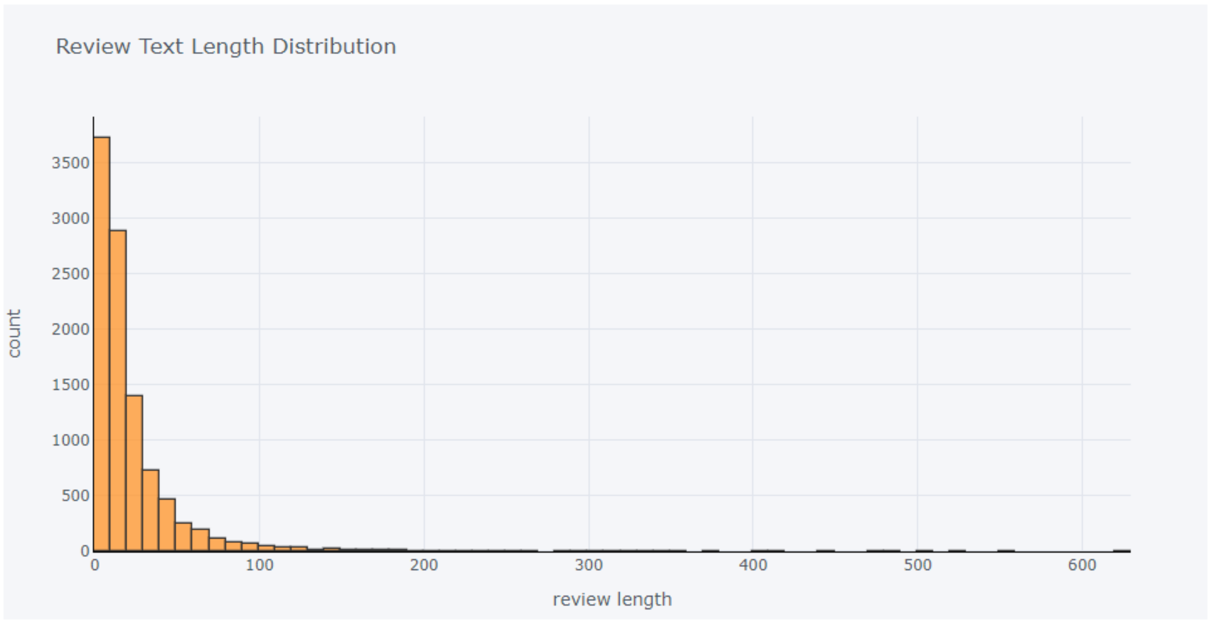
Exploration

To explore the data, we have used graphical representations. Following are few graphs and the observations gathered from them:



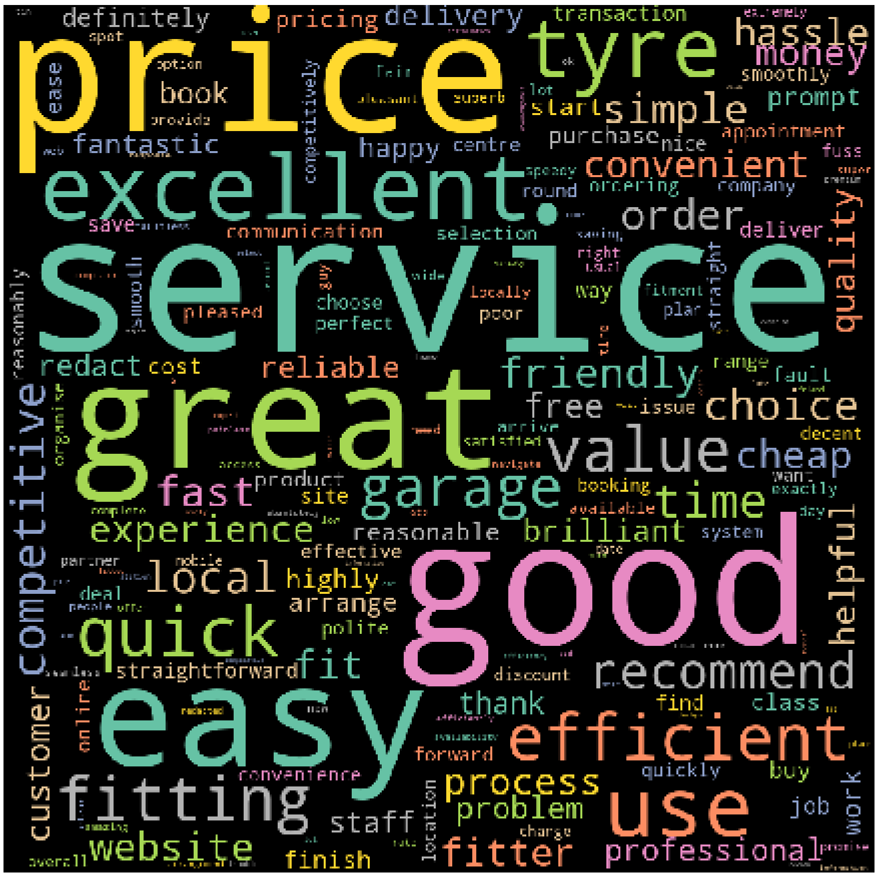
Observation:

- Most of the data lie between being neutral and positive
- Significantly fewer data have negative sentiments
- The number of neutral comments is also very high



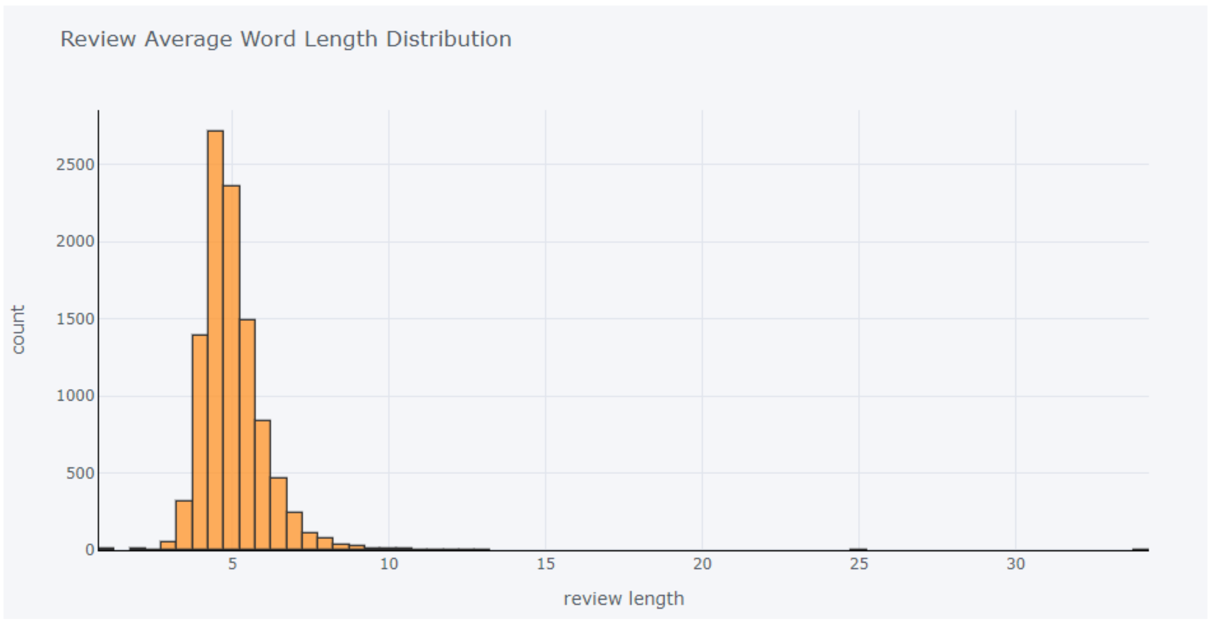
- Most of the reviews have ~10 words, i.e. number of short reviews is at the majority
- There is one review that has around 620-629 words.
- Few reviews do have a tremendous amount of words

Next, we tried to see what the reviews with 0 to 10 words contain.



Observation:

- most of the short reviews are good



Observation:

- Most of the words fall in the range of 3 to 10 in length, with five being the most common
- One of the reviews has an average word length of ~35

Model 1 (Allocating topics) ~ Latent Dirichlet Allocation

1. Since we know the number of topics, we will be using Latent Dirichlet Allocation with the number of topics at 12.
2. We will also not be needed to compare different models to get the best number of topic
3. We will use random_state so that the results can be reproduced
4. We will be fitting the model into the vectorized data and transform it on the same
5. After fitting the model, we will print the top 10 words of each topic
6. We will be assigning the provided topic, based on the closest topic given by LDA, manually
7. After getting the topics, we will be creating a new column and assign the topic

Motivation for the model:

- Ease of implementation
- It can be used for multi labelling
- Small dataset

Hyper-parameters:

- N_topics → 12
- Random_state → 0 12

Time:

- To get the time, we used the %time magic command, which gave an error. We also used %timeit, which also gave an error. On trying to find the reason, it seems to be an issue on Jupyter.

Problems:

- We had to assign the topics with the provided topics manually, which can cause errors

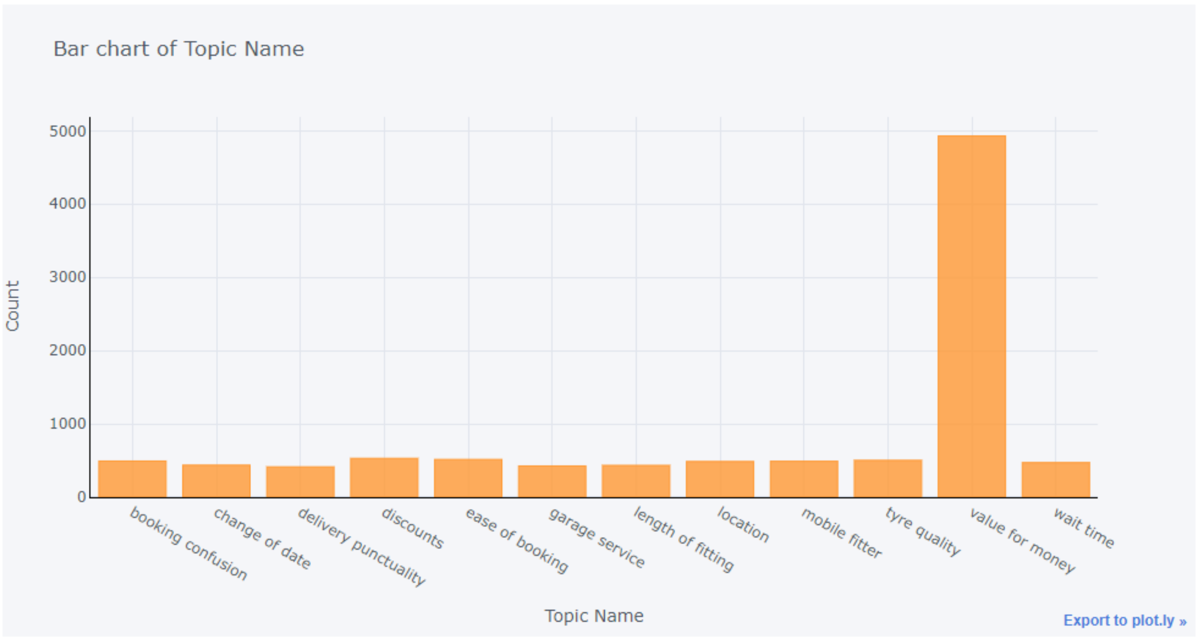
- We could not check if the topics assigned is correct or not
- Only one topic is assigned, while ideally, it should depend on what matches the best.
- In some documents, all the topics have the same probability, which will cause problems, as we are selecting only the max
- Some of the words had no relation with the topic, such as discount, date change.

While reading why the results were not so great, we came across a few blogs and threads. In conclusion, we can say that the reasons are:

1. LDA performs poorly on small texts; most of our data was short.
2. Since the reviews are not coherent, LDA finds it all the more difficult to identify the topics
3. Since the reviews are mainly context-based, hence word co-occurrences based models fail.

Better Approach:

- Use of context-based models



On seeing the assignments, we can see that the dataset is imbalanced, for which we will be using oversampling.

Model 2 (Classifying) ~ Multinomial Naïve Bayes masked with OneVsRest

1. We store the original data into another dataset and do encoding
2. The encoding is done for the Naive Bayes classifier to perform.
3. Pipeline from imbalanced-learn instead of sklearn because the pipeline from sklearn does not support smote.
4. The model used is Multinomial naive Bayes, as is masked by oneVsRest classifier to enable multi-label classification
5. To deal with the imbalanced class, oversampling is done
6. The dataset is passed via TF-IDF vectorizer, as Naive Bayes needs vectorized data

Motivation for the model:

- Fast and easy to implement
- Useful for multiclass classification
- Works well with OneVsRest and language data

Hyper-parameters:

- All set to default

Problems:

- Since it works with probabilities, if the values are not present in training, it assigns 0 probability. This can cause a problem for imbalanced datasets.

Resources:

1. Choosing the right number of topics for scikit-learn topic modeling | Data Science for Journalism (investigate.ai)
2. Contextual Topic Identification. Identifying meaningful topics for… | by Steve Shao | Insight (insightdatascience.com)
3. sklearn.naive_bayes.MultinomialNB — scikit-learn 0.24.2 documentation
4. sklearn. decomposition.LatentDirichletAllocation — scikit-learn 0.24.2 documentation

5. Latent Dirichlet Allocation (Part 1 of 2) - YouTube
6. YouTube
7. NLP Tutorial 13 - Complete Text Processing | End to End NLP Tutorial | NLP for Everyone | KGP Talkie - YouTube
8. Organizing machine learning projects: project management guidelines | by Gideon Mendels | Comet.ml | Medium
9. And numerous StackOverflow solutions…