Capstone Report

March 31st, 2019

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Avnish Srivastava March 31st, 2019

1.1.1 Domain Background

Natural language processing (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages and concerned with programming computers to fruitfully process large natural language corpora.

The project predicts whether reviews are positive or negative also considered as "Sentiment Analysis". Sentiment Analysis is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude towards a topic, product, etc. is positive, negative, or neutral through ML-based and Lexicon-based. In ML-based sentiment analysis algorithms, it requires you to create a model by training the classifier with a set of examples. This ideally means that you must gather a dataset with relevant examples for positive and negative classes, extract these features from the examples and then train your algorithm based on these examples. These algorithms are essentially used for computing the polarity of a document.

1.1.2 Problem Statement

It is important for companies to understand the sentiments of their customers or clients to remain competitive and grow their businesses. Traditionally this depended upon feedback surveys and informal, small-scale and often inefficient methods. Today, however, there is great deal of information on social media and e-commerce platforms related to customer's experiences with products and services. This is available for companies to analyse and better understand their customer's experience and satisfaction.

Accurately identifying the sentiment in a body of text requires accuracy (to be useful) and automation (to deal with the scale and nature of data available today). Machine learning models provide one solution to this problem.

Build a machine learning model using reviews about companies' products and services that will predict whether reviews are positive or negative.

1.1.3 Datasets and Inputs

Here we have collected reviews for various products and services from different sources written between December 2016 and March 2017.

The training dataset is divided into positive and negative reviews:

Positive: 32470 reviews

Negative: 7256 reviews

Data Dictionary:

Each review is in JSON format and contains the following fields:

- 'author': author of this review
- 'crawled': site from where it is crawled
- 'entities': if any
- 'external_links': if any links
- 'highlightText': if any
- 'highlightTitle': if any
- 'language': language of the review
- 'locations': if tracked of the author
- 'ord_in_thread': if any
- 'organizations': organization of author
- 'persons': if any
- 'published': if any
- 'text': review written by author
- 'thread': thread of review
- 'title': title of review
- 'url': link of review
- 'uuid': id of review

1.1.4 Evaluation Metrics

Project will be scored using **Matthews Correlation Coefficient (MCC)**

MCC is defines as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

MCC ranges from -1 (worst, complete disagreement between prediction and observation) to +1 (complete concordance between prediction and observation) with a value of zero indicating that the model is no better than random prediction. Read more about Mathews Correlation Coefficient **here**

Why MCC?

1. MCC is considered one of the best single number representations of the confusion matrix - the formula for MCC considers all four cells in the confusion matrix.

2. MCC is particularly well-suited to unbalanced data sets.

3. MCC does not depend on which class is defined as positive and which one as negative.

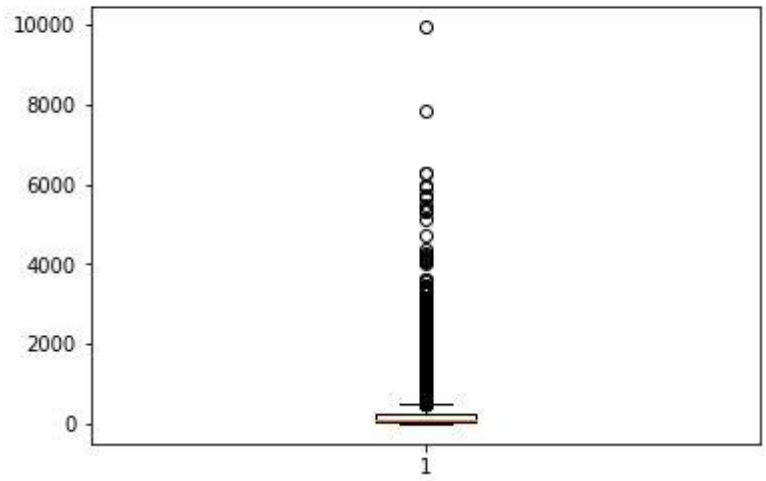
1.1.5 Analysis

The data was present in JSON format originally with 17 columns had lots of missing values in around 13 columns. There are 2 redundant columns as well. Data first read from JSON and made pandas data frame with “text” and “reviews” as columns. The “reviews” is encoded as “1” for positive reviews and “0” for negative reviews. Once the data frame is prepared, it is then exported to CSV so that it increases the speed of data loading process faster.

The dataset is imbalanced with 32470 positive and 7256 negative reviews so “Mathews coefficient” is considered as evaluation metrices since it is well suited to unbalanced data sets.

Data cleaning in natural language processing is a very important process since sometimes if the data is cleaned enough, a benchmark model outperforms machine learning advance model, which you will be noticing in the below steps but before that let’s see how many words customers have used to write reviews in general.

```
Review length:
Mean 199.56 words (median - 108.000000)
```



As expected, customers reviews are positively skewed that is mean of review length is greater than median which suggest that most customers used around 100 words while writing reviews, but some have used high number of words which increases the mean to around 200 words.

Data Cleaning When dealing with numerical data, data cleaning often involves removing null values and duplicate data, dealing with outliers, etc. With text data, there are some common data cleaning techniques, which are also known as text pre-processing techniques.

With text data, this cleaning process can go on forever. Here are a bunch of things you can do to clean your data. We're going to execute just the common cleaning steps here and the rest can be done at a later point to improve our results.

- Remove HTML Tags
- Remove accented characters
- Make text all lower case, remove punctuation, Remove numerical values
- Tokenize and Remove stop words
- Stemming / lemmatization
- Expand Contractions
- Create bi-grams or tri-grams

Data Exploration We have plotted the word cloud of 500 positive and 500 negative reviews which gives us an idea that what words are customer using mostly for positive reviews and what for negative reviews. Based on the figure below, we can see that for positive reviews, the customers have mostly used the word “good”, “great”, “time”, “easy”, “Service”, “price”, etc. while for negative reviews the customers have used word like “order”, “call”, “say”, “pay”, etc.

Positive Review:

Negative Review:

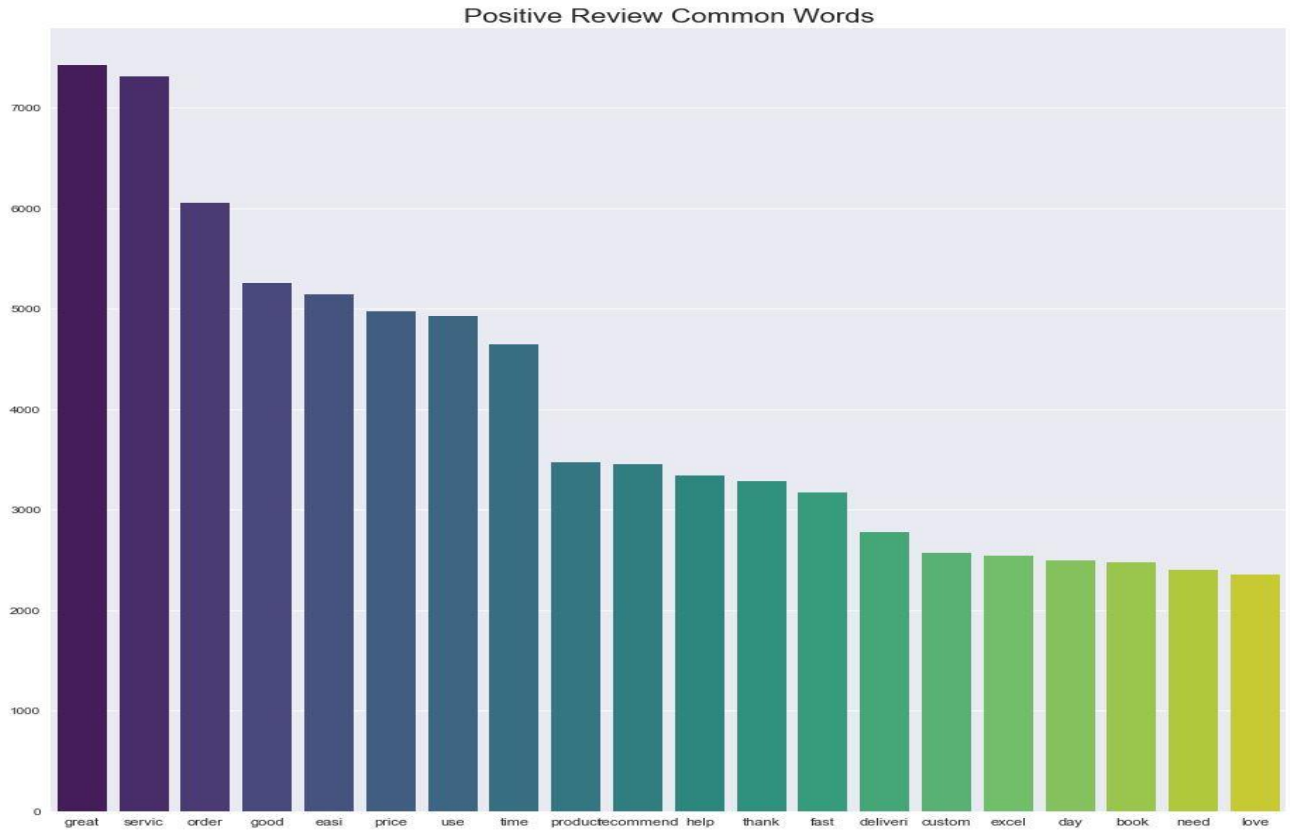
Now, let's tokenize the reviews and remove the English stop words and use nltk PortStemmer for stemming the word and plot the 20 most common positive words and 20 most common negative words.

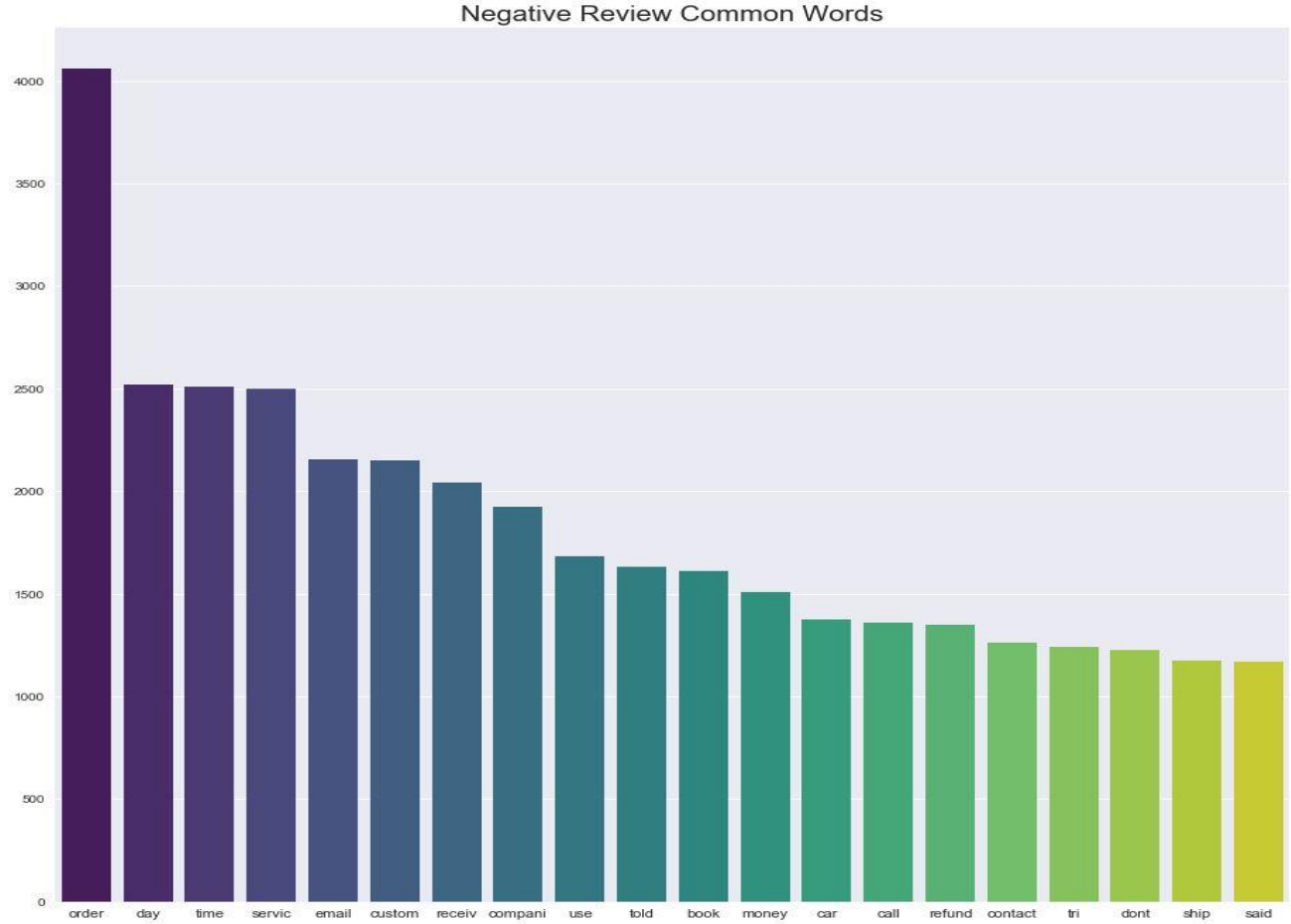
Tokenization of reviews means it is taking a text or set of text and breaking it up into its individual words and stemming through PortStemmer which means it we can create new words by attaching affixes to them in a process known as inflection. Word stems are also known as the base form of a word. Consider the word JUMP. You can add affixes to it and form new words like JUMPS, JUMPED, and JUMPING. In this case, the base word JUMP is the word stem. But Stemming has its disadvantages, it sometimes fails to find the stem of the word correctly, but it takes less time for processing.

The alternative option for PortStemmer is Lemmatization. Lemmatization is very similar to stemming, where we remove word affixes to get to the base form of a word. However, the base form in this case is known as the root word, but not the root stem. The difference being that the root word is always a lexicographically correct word (present in the dictionary), but the root stem may not be so. Thus, root word, also known as the lemma, will always be present in the dictionary. Both NLTK and spacy have excellent lemmatizers. We will be using **nltk** here.

Do note that the lemmatization process is considerably slower than stemming, because an additional step is involved where the root form or lemma is formed by removing the affix from the word if and only if the lemma is present in the dictionary.

NOTE: If you don't have time then go for stemming else lemmatization is the better option


Positive Review Common Words
7000
6000
5000
4000
3000
2000
1000
0
great
servic
order
good
easi
price
use
time
productecommend
help
thank
fast
deliveri
custom
excel
day
book
need
love


Negative Review Common Words
4000
3500
3000
2500
2000
1500
1000
500
0
order
day
time
servic
email
custom
receiv
compani
use
told
book
money
car
call
refund
contact
tri
dont
ship
said

Based on these plots we can clearly see that there are certain words which are present in both positive and negative reviews so will not make much sense, thus we will be deleting those words which have high frequency and some word which have rare frequency i.e., only used once in the reviews.

1.1.6 Benchmark Model

For the benchmark model, the dataset was first converted to Document term matrix using CountVectorizer, simply known as bag of words model with certain parameters which have been derived by cross validation using sklearn GridsearchCV method.

min_df = 5 (at least words appears in 5 document)

max_df = 0.25 (words appear in 25% of document max)

max_features = 5000 (maximum number of features)

n_gram = (1,3) (create tri grams as well)

The document term matrix is then converted to pandas dataframe with 39726 rows and 5000 columns. Since the dataset was imbalanced, we have divided data into train and validation set using stratified sampling which makes sure that equal percentage of positive and negative reviews are present in train and validation set. Based on this and with a multinomial naïve Bayes model we got a Mathews correlation coefficient of 0.7537 which was not that bad as expected.

Let us see our model performance on the data set with all some data cleaning and some more features added to data model.

Performance on data with more cleaning and more features We have removed the contractions from the reviews as one more step of cleaning process

Contractions are shortened version of words or syllables. They often exist in either written or spoken forms in the English language. These shortened versions or contractions of words are created by removing specific letters and sounds. In case of English contractions, they are often created by removing one of the vowels from the word. Examples would be: do not to don't and I would to I'd. Converting each contraction to its expanded, original form helps with text standardization.

Also, as discussed above, we have used lemmatization instead of stemming (definition above) to get the base form of word. We have also removed top 15 common words which are "I", "the", "then", "is", "of", "as", etc which is present in almost every review but makes no sense so removing shall help, similarly we have removed rare words present in the reviews to reduce the review length since they will not be helpful as well.

There are possible to more pre-processing steps can be included in this is "spelling correct" using Textblob and "language translator" since there are lots of words which are either in Dutch language or in German. These processing steps takes lots of time to process so haven't applied in the project but can be used if we don't have time issue and enough memory to handle.

We engineered multiple features like word_count, avg_word and then scaled those values with MinMaxScaler function to keep it in same range.

Once the data is cleaned we have converted the reviews to document term matrix using TF-IDF Vectorizer with certain parameters which was obtained using cross validation.

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(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

Logistic Regression model with L1 regularization technique is applied on this document term matrix after dividing the dataset to train and test using stratified sampling technique and the MCC score obtained is around 0.76 which is better than benchmark model performance but not high increase in MCC suggest applying some other more advanced machine learning models.

Xgboost , randorm forest and stochastic gradient descent algorithm has been also tried on the same dataset but MCC score has not been improved which suggest that Logistic regression will be right choice to choose.

Here we can see that, sometimes with good data cleaning and data pre-processing, simple model can outperform the complex model.

Let's try with some deep learning algorithms:

As suggested in the proposal feedback about applying LSTM algorithm and also used Medium article as reference for applying the algorithm on NLP. We have used max number of words to 10000 and max length sequences to 1000.

Embedding the text data using Keras embedding layers which offers an Embedding layer that can be used for neural networks on text data. It requires that the input data be integer **encoded**, so that each word is represented by a unique integer.

We are using a directional 64 layers and dropout of 0.5 which is used to avoid overfitting. Then in the last, 1 layer has been added with "sigmoid" as activation function.

"binary_crossentropy" and "adam" has been used as loss function and optimizer respectively.

Model has been trained only for 4 epochs with batch size = 32. The MCC score we get using LSTM model with just 4 epochs is around **0.78** which suggest that with more fine tuning the parameters or adding running for more epochs will increase the MCC score significantly.

1.1.7 Conclusion

With lots of data cleaning and adding features, we saw that simple machine learning model can outperform the complex models. We have also seen that using LSTM the result can be improved with just 4 epochs. I believe that running for more epochs or fine tuning the parameter will improve the MCC score but might not that much. The other solution would be to use pre-trained model or pre-trained embedding vector like “GloVe6B” which is “Global Vectors for Word Representation” which will help model to achieve better accuracy and MCC score. We can also use the features of fastai library which I have seen that having better functionalities. There is so much learning in this project 😊

1.1.8 Acknowledgements

This capstone would not have been possible without the extremely helpful Kaggle community for their kernels and discussion forums and some great article on medium.com for NLP and Keras implementation on NLP. Finally, Big thanks to Udacity for the learning experience during the ML NanoDegree program.