Customer review classification analysis based on Amazon, Yelp and IMDb data samples.

Nalinda Kulathunga

March 20, 2018

1 Definition

1.1 Project Overview

Natural Language Processing (NLP) provides an enormous help in understanding the communication through text messages in a machine learning perspective. Every moment of the Cyberspace is filled with uncountable number of comments, reviews and statements however, keep in track with all of them, without loosing much information is a challenge.

Today, Most consumer needs are handled by online service providers and vendors. Customer reviews play an important role in determining the level of customer satisfaction and also as a tool for choosing best places for online-shopping. This also helps the vendor to upgrade the business and to be in alert of the current market trends. Because, about 85% of the consumers trust online reviews as much as personal recommendations[1].

Machine learning techniques are used vastly in classification of reviews as a fast and reliable solution even in the presence of huge amount of data. Ability to learn from data gives an opportunity to predict completely based on artificially built models such as Nural Networks (NN). Classification problems in NLP have been solved since one of the first notably successful NLP systems were developed by Terry Winograd back in 1968[2]. For past few decades, many successful machine learning models have been developed so that NLP problems can be solved accurately and far more better than a normal human being could do. This project is based on classification of consumer reviews written on three different companies (Yelp, Amazon and IMDb) using different classification models and compare between companies for the accuracy of the predictions made by an optimized model.

Data sets were obtained from "Sentiment Labeled Sentences Data Set" at the web site of Center for Machine Learning and Intelligent systems at University of California, Irvine[3]. These data sets provide information about three types (Amazon - online shopping, Yelp - restaurants, IMDb - movies) of "good" (labeled - 1) and "bad" (labeled - 0) reviews.

[1] https://www.brightlocal.com/learn/local-consumer-review-survey/
[2] https://en.wikipedia.org/wiki/History_of_natural_language_processing
[3] https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

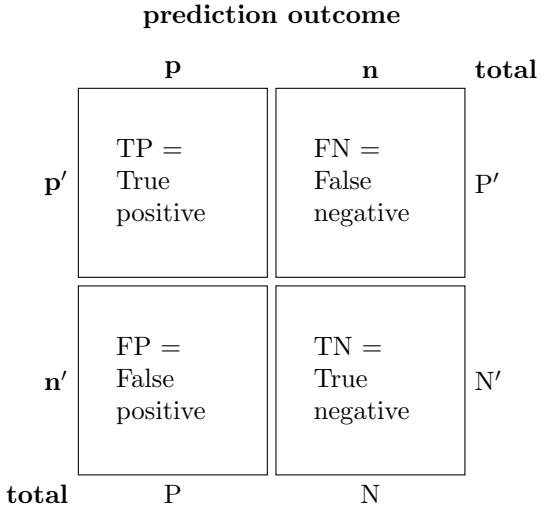
1.2 Problem Statement

Machine learning techniques are used to identify a suitable model after training, to classify and predict the labels of customer reviews as good or bad. But this analysis extend the scope to identify whether customer reviews are company specific or not. That is, customers may be using similar patterns of words like "bad", "good", "fine" etc.. and one might not need to consider reviews from a particular company to predict it's own reviews. Instead models can be trained on many reviews regardless of which company they belongs and still predict a review with acceptable accuracy. This approach lower the cost of training different models on different company reviews and may be provide one global model suitable to predict and classify most company reviews.

So, the intention is to find a model based on the test accuracy which can best predict the labels. Several classifiers will be tested (Multinomial Naive Bayes classifier, Stochastic Gradient Decent classifer and Convolutional Neural Network) and best classifier will be optimized. Once the classifier is selected, It will be trained and tested up-on different combinations of Training and testing samples. For an instance, one combination would be (Comb1); trained on a combined data set of all three companies and tested on one company's reviews, next combination would be (Comb2); trained and tested on the same company's reviews. If the test accuracy for "Comb1" is greater than or equal to that of "Comb2", then we can argue that the reviews are not company specific, when it comes to training a classification model.

1.3 Metrics

- **Confusion Matrix** [4] This is one of the detailed description of the validity of predictions against true labels. In most machine learning problems, we can categorize predictions depending on whether they are correct or wrong. We have four possible categories (Think of a data set with "yes" and "no" in labels):
  - True positives (TP): predicted "yes" and labeled "yes".
  - True negatives (TN): predicted "no" and labeled "no".
  - False positives (FP): predicted "yes" but labeled "no".
  - False negatives (FN): predicted "no" but labeled "yes".



Confusion Matrix

[4] https://doi.org/10.1016/S0034-4257(97)00083-7

- **Accuracy:** This calculates the number of items classified correctly out of the total sample. This way of measuring the reliability of a model is not suitable for most cases, specially where the labels aren't distributed equally. But in this analysis, since there are almost equal number of "positive" and "negative" reviews, accuracy will provide a reasonable insight to the model performance.

$$Accuracy := \frac{TP + TN}{Total} \tag{1}$$

- **Precision and Recall:** [5] These two metrics provide information about the non-diagonal components of the confusion matrix. Precision informs us basically what fraction of positively classified reviews are truly positive and Recall is about how many relevant items are chosen and in this problem it tell us what fraction of positively classified reviews are classified correctly. These two metrics are important since they might reflect any asymmetries in classification. Also this gives an idea about which labels are classified correctly most of the time.

$$Precision := \frac{TP}{TP + FP} \tag{2}$$

$$Recall := \frac{TP}{TP + FN} \tag{3}$$

$$\tag{4}$$

- F_β **Score:** [6] For the evaluation of model's overall performance, both precision and recall are needed with appropriate weights. This metric provides that utility. So, F_β score is a much better matrix for the evaluation of model's performance.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{5}$$

[5] http://www.flinders.edu.au/science_engineering/fms/School-CSEM/publications/tech_reps-research_artfcts/TRRA_2007.pdf

[6]

2 Analysis

2.1 Data Exploration and Visualization

This data set was obtained from "Sentiment Labeled Sentences Data Set" from the web site of Center for Machine Learning and Intelligent systems at University of California, Irvine[7]. It is consist of 3000 positively or negatively labeled customer reviews from three different companies; yelp.com, amazon.com and imdb.com. Each company has about 1000 reviews and there are no mixed reviews. That is, no neutral reviews are included. Prepossessing should be done on the data set. Given bellow (Table 1, 2 and 3) are few reviews with there labels for each company and Table 4 shows a sample of shuffled combined reviews from all three companies.

	Review	Label
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1
5	I have to jiggle the plug to get it to line up...	0
6	If you have several dozen or several hundred c...	0
7	If you are Razr owner...you must have this!	1
8	Needless to say, I wasted my money.	0
9	What a waste of money and time!.	0

Table 01: Reviews (first 10) from amazon.com.

	Review	Label
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
5	The rest of the movie lacks art, charm, meanin...	0
6	Wasted two hours.	0
7	Saw the movie today and thought it was a good ...	1
8	A bit predictable.	0
9	Loved the casting of Jimmy Buffet as the scien...	1

Table 02: Reviews (first 10) from imdb.com.

	Review	Label
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
5	Now I am getting angry and I want my damn pho.	0
6	Honeslty it didn't taste THAT fresh.)	0
7	The potatoes were like rubber and you could te...	0
8	The fries were great too.	1
9	A great touch.	1

Table 03: Reviews (first 10) from yelp.com.

	Review	Label
0	Top line: Don't waste your time and money on t...	0
1	The commercials are the most misleading.	0
2	They brought a fresh batch of fries and I was ...	0
3	This phone tries very hard to do everything bu...	0
4	This hole in the wall has great Mexican street...	1
5	The story line is totally predictable.	0
6	I would have given no star if I was able.	0
7	Oh and I forgot to also mention the weird colo...	0
8	It really is impressive that the place hasn't ...	0
9	But this movie is definitely a below average r...	0

Table 04: Reviews (first 10) from amazon.com, imdb.com and yelp.com together and shuffled.

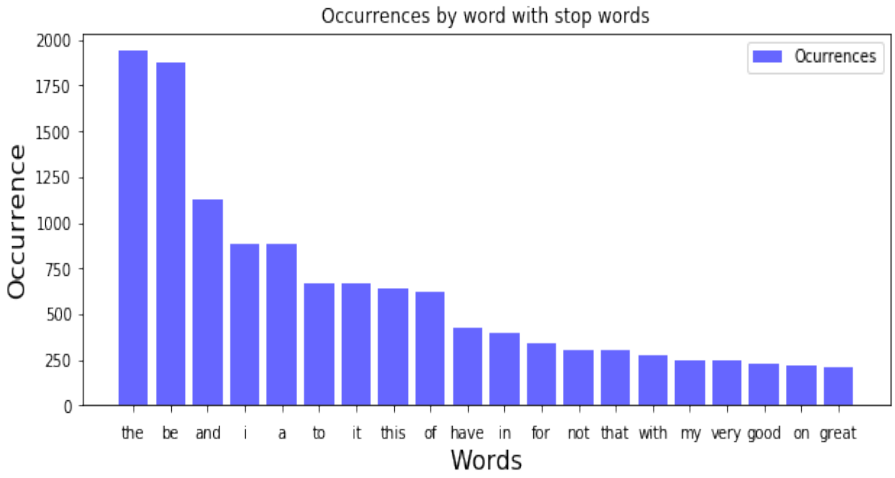


Figure 1: Occurrence of most frequent twenty words, before removal of stop-words.

[7] https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

This data set is manually labeled and is a small sub set of a very large data set. These reviews are inspected thoroughly in order to get rid of any neutral reviews. These steps have been already taken for the purpose of testing a new classification model in an earlier NLP analysis[8]. There fore I have tried not to exclude any sentences since they carry valuable information about the corresponding label. Also I wanted to include many reviews as possible in the analysis to avoid running out of statistics.

However, random check on the sentences reveled that they need to be further processed before training is done. This included creating a bag of words in preparation for the vectorization, removing stop-words which do not provide valuable information about labels, lemmatization to find the stem of similar words and also the removal of accent characters. Most of the techniques for basic text data pre-processing was obtained from this online document[9].

For an instance, with the presence of stop-words, effects of the true feature words (like "good", "great", "bad", etc..) are hidden behind as shown in Figure 1. That means, in Figure 1, most occurred five words are "the", "be", "and", "i" and "a". These words carry no information about the labels. Once the stop words are removed, a model can easily recognize the key feature words needed in predicting the labels. Also, the most frequent word "the" occurred 2000 times. So, Removing those words also make process faster. It is a plus point that, if you see beyond the word "for" in "Words" (positive X) axis, most words are quite informative, like "not", "very", "good", "great", etc.. . We can use those feature words to get better idea about the labels.

2.2 Algorithms and Techniques

This supervised learning problem can be addressed in many different approaches. However in this project, the algorithms are chosen in order to get a better accuracy and also can be handled with simplicity. Firstly Multinomial Naive Bayes (MultinomialNB[10]) was fitted with data and then stochastic gradient descent classifier (SGDClassifier[11]) was used. Both models were tested varying parameters and optimal model was selected. Finally several architectures of convolutional neural network (CNN) with hidden layers were tested. When selecting a model/classifier, it is impotent to consider the fact that labels of the reviews are discrete and sparse arrays of data can also be present after pre-processing. The performance of these models were tested with respect to a benchmark model and best classifier was selected. A rough cartoon of CNN structure is shown in Figure 2[12]. All the techniques and algorithms used in this analysis are listed bellow.

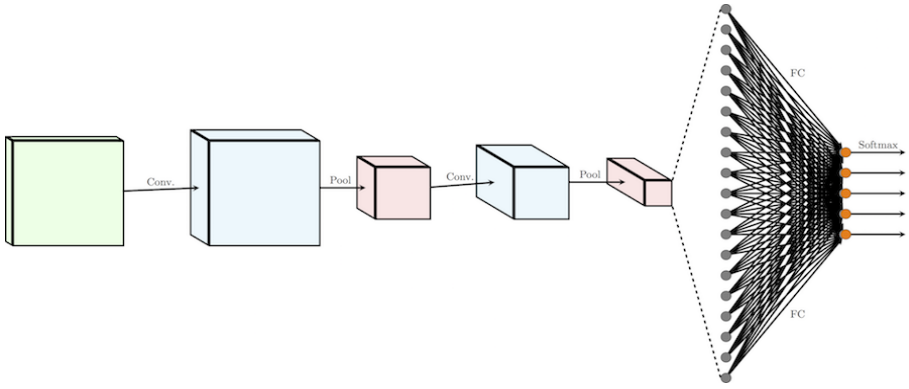


Figure 2: CNN architecture cartoon.

[8] http://mdenil.com/media/papers/2015-deep-multi-instance-learning.pdf
[9] http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
[10] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[11] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[12] https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

1. **Vectorization/Bag of words:** [13] Idea behind bag of words (BOW) id fairly simple but powerful. We can not use words themselves as input for a certain classification problem. So we can first find all possible words in a certain data set (BOW) and give an index (integer values) to each word in the sentence. This vectorizes the word but wouldn't capture the meaning of the text accurately.

2. **tf-idf:** [14] Term frequency–inverse document frequency (tf-idf) is a technique which reflects the importance of a certain word in a sentence. It takes the frequency of a certain word in a sentence in to account and does a weighing accordingly. Importance of the word increases proportionally to the number of times a word appears in the document but it is offset by the frequency of the word in the corpus. This ranking can be calculated by the following approach:

   TF (Term Frequency):

$$TF = \frac{Numberoftimestermtappearsinadocument}{Totalnumberoftermsinthedocument} \tag{6}$$

$$\tag{7}$$

   IDF (Inverse Document Frequency)

$$IDF = \log \frac{Totalnumberofdocuments}{Numberofdocumentswithtermtinit} \tag{8}$$

$$\tag{9}$$

3. **Embedding:** [15]: This is also a popular vectorization technique which maps words or phrases in to vectors of real numbers. In NLP, words are more treated as discrete symbols. So, on the case of to distinct symbols that have some common feature, this approach fails to extract the stem. For and instance, "CAR" and "BUS" both are in the vehicle class even though they have distinct character symbols. To overcome this issue, techniques like Vector space models (VSMs) embed words in a continues vector space where semantically similar words are mapped on to locally nearby points. Embedding is really useful as a input layer to a convulsional neural network (CNN) to represent text in vector form without loosing the core meaning of the sentence. One could refer to the keras documentation for further details about embedding[16].

4. **Padding**: This technique is applied to make finite fixed size vectors for the analysis. This is applied after an embedding is done. After the embedding, vectors are of variable lengths. This make it difficult for a classifier to work with. So throwing zeros without changing the initial meaning (pattern), until all the vectors are in fixed size is crucial.

5. **MultinomialNB**:
   The Naive Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem with naive but strong and meaningful independent assumptions. In Multinomial Naive Bayes, it estimates the conditional probability of a particular word after it's been tocanized, given a class as the relative frequency of a term in a certain documents which belonging to a particular class[17]. This approach is most suitable for this analysis because of the limited number of reviews present. Also this takes small CPU time for the process.

[13] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[14] https://en.wikipedia.org/wiki/Tf-idf
[15] https://en.wikipedia.org/wiki/Word_embedding
[16] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding
[17] https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

6. **SGDClassifier**:
   In stochastic gradient decent, it uses a stochastic approximation of the optimal decent of the gradient and tries to find a maxima or minima by iteration. This is also a much faster algorithm and can also be used to learn online. Finding a learning rate which gives an optimized results is the key factor in SGD classifier which will be implemented later in this analysis. SGD cassifier can be used in the presence of dense or sparse arrays which make this suitable for this analysis and also this can be easily implemented. But it is impotent to shuffle the data before use this technique for better results. Further information can be found in this really helpful sklearn documentation[18].

7. **Convolutional Neural Networks (CNNs)**:

   CNNs are widely used in image classification problems. But with the help of embedding in put layer, we can implement this model in text classification problems. I wanted to try a bit complex model in this analysis since previous two methods (MultinomialNB and SGD Classifier) are relatively simple and easy to implement.

   As shown in Figure 2, normally a CNN is consist of several convolution layers and pooling layers and fully connected layers. But in this special case of text classification we had to include a embedding layer as the input following the standard architecture. Convolution layer works by using a weight matrix which we define by ourselves, and scanning the image matrix (In our case it is the embedded vector representation) to extract certain features from it.Sometimes when the images (matrices) are too large, we need to reduce the number of trainable parameters. That is the reason we periodically introduce pooling layers between subsequent convolution layers. However all these above layers are able to extract features and reduce the number of parameters in the problem. We still needs the classification. So we also introduce fully connected at the end to get our desired output. Drop out technique is also used to prevent or reduce over-fitting. Dropping-out means skipping certain units in given intervals in a neural network. Important detailed description about CNNs and hidden layers can be found in this useful document [19]

   My CNN architecture consist of following layers to obtain the above mentioned feature extraction and classifications;

   - Input - embedding layer
   - Convolution 1D layer
   - Convolution 1D layer
   - Flattern layer
   - Drop-out
   - Dence layer
   - Drop-out
   - Dence layer

2.3 Benchmark

- **Naive Guesser**: In this analysis data set comprise of approximately equal number of "positive" labels and "negative" labels. So, the naive classifier would be a random guesser. Here the predicted labels are assumed to be all "positive" and based on that assumption, accuracy is calculated. Since almost 50% of the labels are truly positive, accuracy is also 50%. Later implementations are done not only looking at the accuracy metric but also taking confusion matrix, precision, recall and F-score in to account. But, accuracy value can still be a good measure of the model performance since positively and negatively labels are equally distributed.

[18] http://scikit-learn.org/stable/modules/sgd.html
[19] https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/

Metrics	Value
Accuracy	0.5
Precision	0.5
Recall	1.0
F_1	0.67

Table 1: Metrics for **Naive Classifier** assuming it randomly selects events

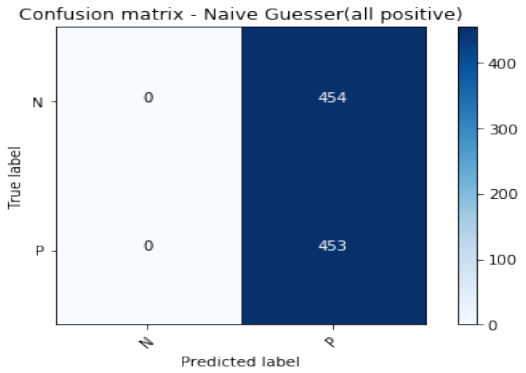


Figure 3: Confusion Matrix for **Naive guesser**

3 Methodology

3.1 Programming Language and Libraries

- **Python 2.7.**
- **Pandas.**
- **Scikit-learn.**
- **NLTK (Natural language toolkit).**
- **Keras.**
- **TensorFlow.**
- **Matplotlib.**

3.2 Data Pre-processing

3.2.1 Basic pre-processing

Some of the basic pre-processing such as, removing punctuation marks and converting characters to lowercase were applied in the beginning of the analysis. Removal of the stop words is done with the help of NLTK (Natural language tool-kit)[20] libraries. Without the stop words, it is clear that, most important feature words like "good", "great" and "bad" have become most occurred words as shown in Figure 4.

[20] http://www.nltk.org/

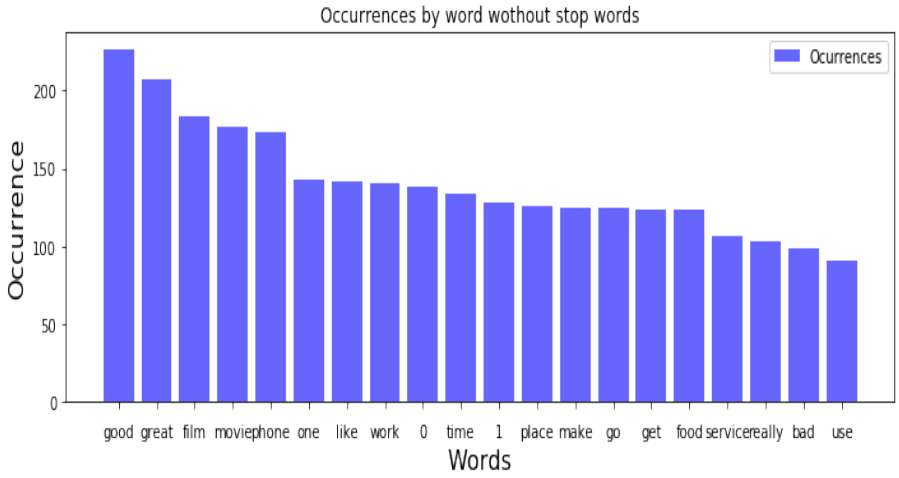


Figure 4: Occurrence of most frequent twenty words, after removal of stop-words.

3.2.2 Lemmatization

To avoid counting words which are directed towards a common meaning, we need to identify the stem of such series of words and put them under one feature name. For an instance; "look", "looking", "looked" and "looks" all give the same meaning - "look". Lemmatizing was used to make this job done with the help of NLTK[21] libraries and decoding of accent characters was needed beforehand. The NLTK vocabulary of words called "wordnet" was downloaded for this purpose.

3.2.3 Vectorization

Bag of words is working based on tokenizing each distinct word and finally counting them only if never occurred before in the text data set. Both tokenizing and counting can be easily done using one of the scikit-learn built-in support called "CountVectorizer". This also provides an option to remove stop-words. Further more *tf-idf* was used in a "pipeline" together with "CountVectorizer" and a classifier.

3.2.4 Embedding and Padding

Embedding was used in order to pre-process text data before using in the convolutional neural network (CNN). This enables us to go to a lower dimension in word space. The similarity between vectors are evaluated using techniques like in the calculation of "cosine similarity" between two vectors. In embedding, each word in a sentence get vectorized with arbitrary size. Therefore padding was applied in order to have fixed sized vectors.

3.3 Implementation and Refinement

For the selection of best model, all the following implementations and refinements are done by training and testing on train (67%) and test (33%) samples from all three data sets (Amazone, Yelp and IMDb) combined together.

3.3.1 Multinomial Naive Bayes Classifier (MultinomialNB)

This implementation was done using a "pipeline" consist of "CountVectorizer", "TfidfTransformer" and finally the "MultinomialNB" classifier, all with default settings. Table 2 shows the performance of the model

[21] http://textminingonline.com/dive-into-nltk-part-iv-stemming-and-lemmatization

without optimization.

Metrics	Value
Accuracy	0.79
Precision	0.80
Recall	0.80
F_1	0.80

Table 2: Metrics for **MultinomialNB** - not optimized.

Then the optimized parameters were found by using "GridSearchCV" and main contributor to optimized results was the parameter "alpha" which controls the form of the model itself. "alpha" was changed in the range [0.1, 0.4, 0.6, 0.7, 0.8, 1.0] where 1.0 is the default. Grid search gave 0.6 as the best value for the model with accuracy greater than 0.80. Performance in the optimal mode is shown in Table 3 and confusion matrix is shown in Figure 5.

Metrics	Value
Accuracy	0.801
Precision	0.81
Recall	0.80
F_1	0.80

Table 3: Metrics for **MultinomialNB** - optimized.

In all the implementations I believe low statistics of data samples made it difficult for the models to go for high accuracy.

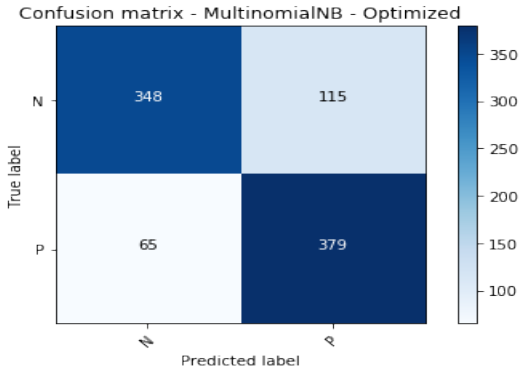


Figure 5: Confusion Matrix for **MultinomialNB** - optimized.

3.3.2 Stochastic Gradient Decent (SGD) Classifier

This classification was also applied together with "CountVectorizer" and "TfidfTransformer" in a "pipeline". First, the model was trained with arbitrary parameters and the GridSearchCV was used to find the optimal parameters. SGD classifier preformed with slightly lower accuracy with respect to MultinomialNB classifier. Performance are listed in Table 4 and confusion metric is shown in Figure 6.

Metrics	Value
Accuracy	0.778
Precision	0.78
Recall	0.78
F_1	0.78

Table 4: Metrics for **SGDClassifier** - optimized.

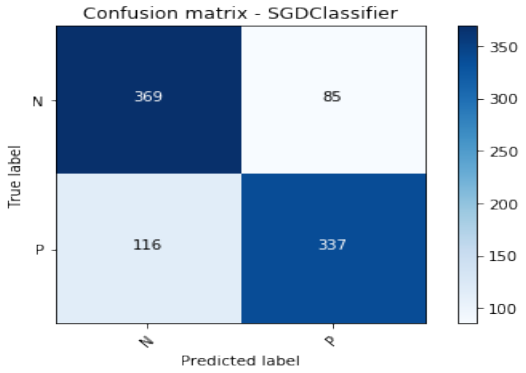


Figure 6: Confusion Matrix for **SGDClassifier** - optimized.

3.3.3 Convolutional Neural Network (CNN)

Convolutional neural network architecture was implemented as shown bellow which gave better performance after testing different architectures. But still the accuracy score was bellow the accuracy values of MultinomialNB classifier and SGD Classifier. Shown bellow is the network architecture with layers.

- CNN model

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 200, 200)          800000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 200, 64)           38464
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 200, 32)           6176
_________________________________________________________________
flatten_4 (Flatten)          (None, 6400)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6400)              0
_________________________________________________________________
dense_4 (Dense)              (None, 180)               1152180
_________________________________________________________________
dropout_2 (Dropout)          (None, 180)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 181
=================================================================
```

```
Total params: 1,997,001.0
Trainable params: 1,997,001.0
Non-trainable params: 0.0
_________________________________________________________________

The maximum number of words considered was 200
Embedding dimension: 200
Drop-out rate : 0.2
The activation function used was a "sigmoid"
```

In this architecture, I have selected the embedding layer as the input layer followed by two "conv1d" layers. Then the output was flattened and dropout was applied with a rate of 0.2. Then a dense layer of 180 units was added followed by another dropout layer with same rate. Finally another dense layer was introduced at the output. I tried varying different architectures in order to obtain better accuracy, however it was not performed well like in the case of "MultinomialNB". Performance and the confusion matrix are shown in Table 5 and Figure 7.

Metrics	Value
Accuracy	0.73
Precision	0.73
Recall	0.73
F_1	0.73

Table 5: Metrics for **CNN model.**

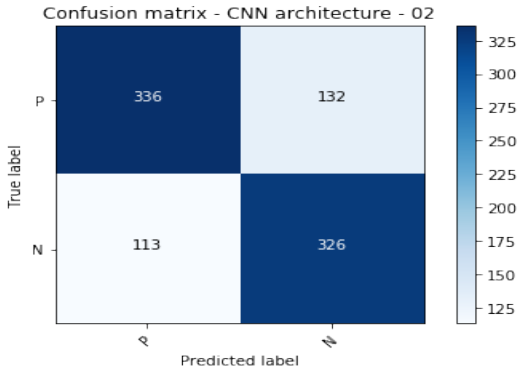


Figure 7: Confusion Matrix for **CNN model.**

3.3.4 Implementation Challenges

Biggest implementation challenge I faced was working with low statistics. I have seen that the accuracy is sensitive to the change in the number of reviews used in the training. Also, I had a hard time working with accent characters and finally found that NLTK libraries have the needed tools for the refinement.

3.3.5 Using the best performed model and comparing the effect of different training and testing samples.

Objective of this study was to investigate how the selection of training set effects the accuracy of predictions on reviews of a certain company. In the previous implementations, it was found that "Multinomial Naive Bayes classifier performs better than the other chosen models. It showed an accuracy of above 80% when tested on the test data sample obtained by combining reviews of all three companies (Amazon, Yelp and IMDb) together. This selected optimized model will be trained and tested in following ways and tested on reviews of individual companies. Each approach is named with an abbreviation for convenience.

- Comb - Amaz : Trained on Combined reviews and tested on Amazon.
- Comb - Yelp : Trained on Combined reviews and tested on Yelp.
- Comb - IMDb : Trained on Combined reviews and tested on IMDb.
- Amaz - Amaz : Trained on Amazon reviews and tested on Amazon.
- Yelp - Yelp : Trained on Yelp reviews and tested on Yelp.
- IMDb - IMDb : Trained on IMDb reviews and tested on IMDb.

4 Results

4.1 performance of the Multinomial Naive Bayes classifier on different combinations of training and testing data.

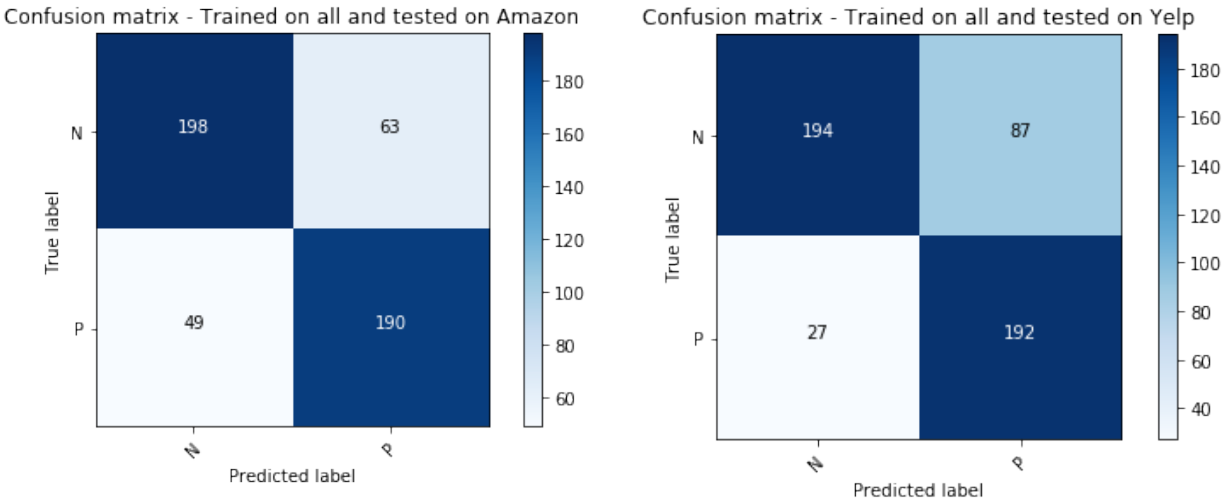
Table 6 shows the evaluation matrices for different combinations of training and testing combinations. All these combinations were trained by using Multinomial Naive Bayes classifier with optimized parameters as shown in the implementation section.

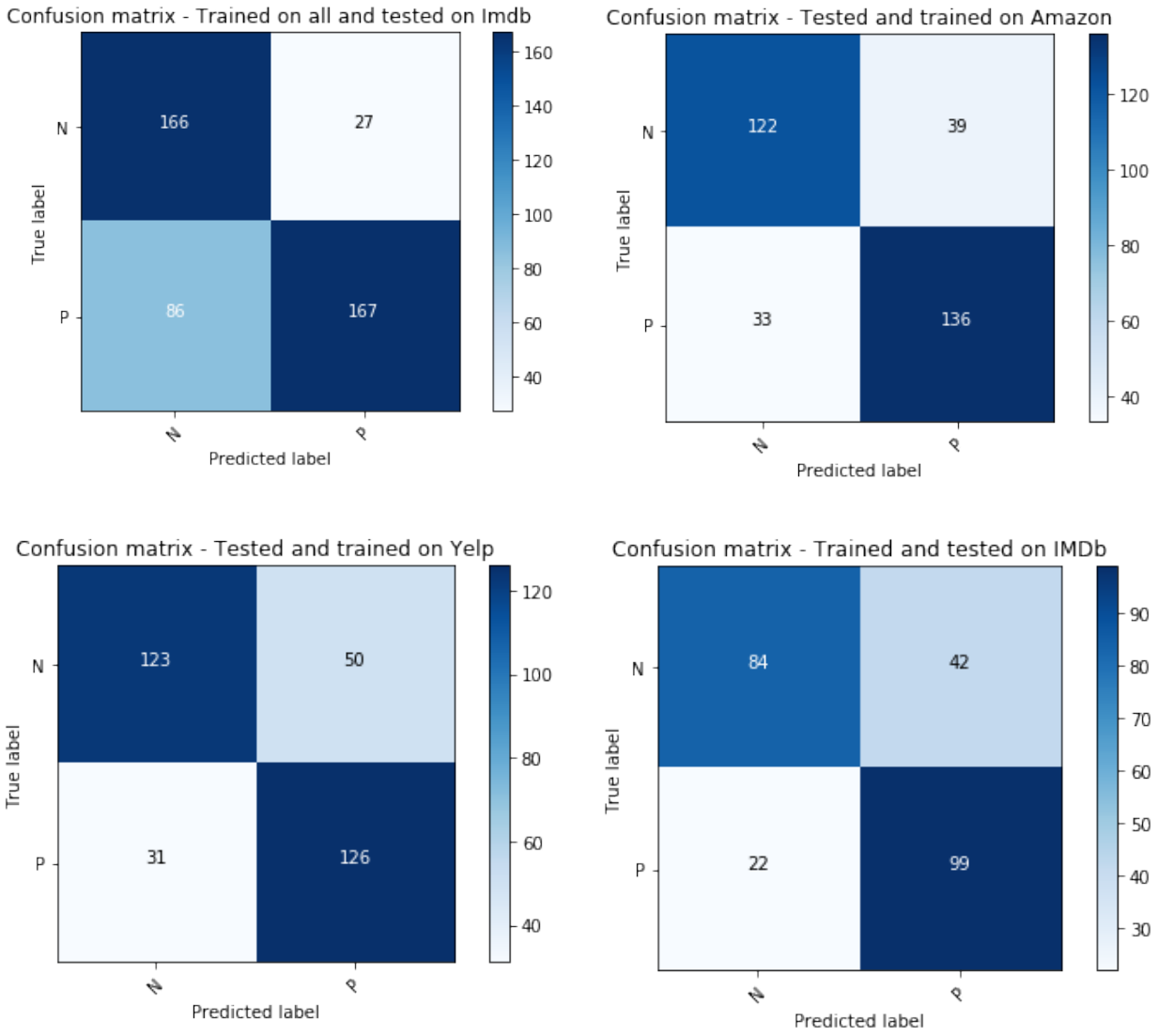
Additional care was taken in the case of training from combined reviews and testing on one company. That is, we have to exclude the set of test data for a particular company form the combined reviews where we intend to train the model. For this purpose, first an individual data set was divided in to two equal portions (eg: Amazon.txt in to AmazonTrain.txt and AmazonTest.txt) and only training set (AmazonTrain.txt) was added to other data sets (Yelp.txt and IMDb.txt) to make the combined data set. Once the training is done, model is tested on test sample (AmazonTest.txt).

Combination	Accuracy	Precision	Recall	F_β score
Comb - Amaz	0.786	0.780	0.780	0.780
Comb - Yelp	0.772	0.790	0.770	0.770
Comb - IMDb	0.747	0.770	0.750	0.750
Amaz - Amaz	0.780	0.780	0.780	0.780
Yelp - Yelp	0.754	0.760	0.750	0.750
IMDb - IMDb	0.741	0.750	0.740	0.740

Table 6: Metrics Evaluation for different combinations of training and test sets.

Confusion matrices for each combination of training and testing data sets are shown bellow.





4.2 Justification

Firstly, the optimized classification model was selected by testing several classifiers and choosing the one with the highest accuracy. Once a model was selected (in this case the MultinomialNB classifier) further tests like "GridSearchCV" were carried out to find the optimal parameters. Even though the accuracy was not perfect, with this small number of training samples, this model can be justified as a reasonable estimator of labels.

Secondly, as shown in the results in Table 6, we can justify that the model was able to reflect the effects of choosing different training and testing combinations. By small margin, the accuracy of classifier, when trained on combined-data showed greater score in all three instances.

Also the pre-processing done before training the models was crucial in obtaining precise results. However all the models performed far more better than the naive guesser and for the future implementations one could use MultinomialNB as a benchmark and go for even better accuracy with different approaches.

5 Conclusion

- Among the classifiers used in predicting the labels of reviews data, Multinomial Naive Bayes classifier showed the best accuracy of 0.801 when trained on combined-data training sample and tested on combined-data test sample.
- Since the testing accuracy shows slightly better score when trained on combined-data, we can conclude that model selection to predict labels of reviews in a certain company can be done using common reviews, even without necessarily using reviews of that particular company. This gives some flexibility to data selection but this matter/conclusion should be further thoroughly investigated.

5.1 Free form visualization

I have selected the following Figure 8 as a free form of visualizing the performance of the project. This is how the accuracy and loss changes when the CNN was trained. You can see that it approaches a desired accuracy and very small loss which indicates the proper learning process.

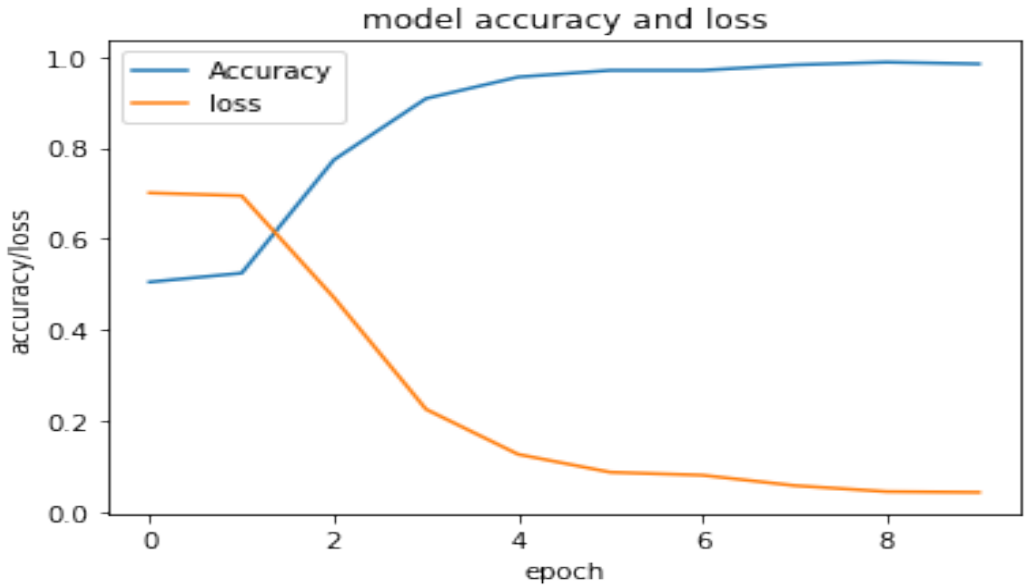


Figure 8: Accuracy and loss vs epochs in CNN training.

5.2 Reflection

This analysis is based on customer reviews from three companies which have distinct business scopes. So this gave an opportunity to compare between reviews of different companies and how that effects a classification problem. Data sets were not the cleanest and had to perform several processes in order to achieve a reasonable accuracy.

This also reflect the fact that more complex models are not always the best. For an instance, CNN model was tested with different architectures and adding more hidden layers, but still Mailinomial Naive Bayes classifier was the one which showed the best accuracy.

As in the Table 6, this may be a good analysis to continue with a deep study and show that reviews are not company specific. So, identifying key features for label prediction is independent of the company where the review was written on.

End-to-End Reflection:

- Obtained the data sets with customer reviews labeled "good"(labeled 1) or "bad"(labeled 0) for amazon.com, yelp.com and imdb.com
- Pre-processing data using techniques; removing stop words, lemmatizing, etc..
- Selected the best classifier for this problem as Multinomaial Naive Bayes classifier.
- Optimized the MaultinomialNB classifier with best parameters.
- Used different combinations of training and testing data sets to train and test above selected classifier and analyzed the accuracy scores.
- It is concluded that classification of reviews for a certain company does not necessarily need a classifier trained on the same type(company) reviews. That is, general human reviews have their own features make them positive or negative reviews regardless of the type of business/work addressed.

6 Improvements and Additional Algorithms

I certainly suggest to do the analysis on a larger data set (at least with 10,000 total reviews). It would help in obtaining stable answers and evaluations. Long and Short Term Memory networks (LSTM) would be a better approach for this classification problem, but it may be time consuming at the same time. I also believe, more attention should be payed to choose a better model architecture for CNN.