Sentiment Analysis on Yelp Reviews

A report is presented for the degree of
Master of Data Science at Monash University in Semester 2 2019
Faculty Of Information Technology

Team Members

Pradnya Alchetti - 29595916
Sarokrishna Ekambarakrishnan - 30068029
Suchita Balasubramanian - 29878608

Contents

1 Introduction

Data has become the key resource for businesses. The data can be structured or unstructured such as emails, chats, reviews or articles. Analysis of this data has been receiving increasing interest in recent years (Mäntylä, Graziotin, & Kuutila, 2018). Sentiment analysis is used to analyze the unstructured data with the help of Natural Language Processing (NLP)(Mäntylä et al., 2018). Sentiment analysis consists of various number of methods, techniques, and tools that builds machine learning algorithms to classify the texts as positive or negative which helps to gauge the sentiments, opinions, emotions or attitudes(Pandey, 2019). Analysis of the sentiments is useful for researchers in the field of marketing, advertising or political sciences(Pandey, 2019).

In this data analysis challenge, we analyze the Yelp reviews with the polarity of opinions such as strong negative, weak negative, neutral, weak positive and strong positive. The dataset consists of training data with 650000 reviews and test data with 50000 reviews. The distribution of the given data is as follows:

Dataset	Number of records	Description
labeled_data.csv	50000	It consists of text reviews with their labels of polarity
unlabeled_data.csv	600000	It consists of text reviews with no labels
test_data.csv	50000	It consists of review ids and reviews

Figure 1: Yelp review dataset details

We read, preprocess and analyze these reviews with the help of feature engineering. We analyze the sentiments by building semi-supervised classification models on the training data which consists of `labeled_data` and `unlabeled_data`. We evaluate our models based on their accuracy and predict the sentiments for the `test_data`.

The initial research on text classification and sentiment analysis led to the following steps to be carried out as a part of the analysis:

Steps	Details
Preprocessing	Removal of special characters and new line characters Tokenization Lemmatization Stopwords Removal Frequent word removal Rare tokens removal Word expansion such as dont' -> do not N-grams
Feature Engineering	TFIDF Word2vec
Modelling	Logistic Neural Network Ensemble (Naive Bayes, Logistic, Neural Network)

Figure 2: Preprocessing steps

The above pre-processing steps were altered based on the model accuracy.

1.1 Work Distribution

- Initial preprocessing was equally divided into two group members and the third member worked on setting up an initial multinomial logistic model for training and testing purposes.
- Once a base pre-processing and model was set, all the three worked on different combinations of pre-processing to create best preprocessing steps.
- All the three members explored different feature engineering techniques such as Countvectorizer, TFIDF, Word2vec to achieve the best feature engineering for the analysis.
- Various different models were taken into consideration such as linear SVM, Convolution Neural Networks, XGBoost and AdaBoost before finalizing the three best models.
- All three explored different models with different combinations of preprocessing steps and finalized with Multinomial Logistic, Neural Network and Ensemble of Logistic, Neural Network and Naive bayes.
- We also explored transfer learning and language models such as FastAI.
- Selecting the best feature engineering method and the model was decided after a discussion with the team and brainstorming.

2 Preprocessing and Feature Generation

2.1 Pre-processing

A data frame was created using the labelled dataset. In this, each row corresponds to one review and label (sentiment for that review). Each review was tokenized using a tokenizer and a list of tokens was generated. These tokens were then joined using a "space" and were stored in the dataframe with the corresponding label. We chose to store the data back into the dataframe as we didn't have an ideal review Id or a key that would allow us to store the data in a dictionary. We performed a series of combinations of the following NLP techniques on Logistic Regression model to generate final preprocessing steps.

- **Case Normalisation and Tokenization:** Some of the words in the reviews were repeated with different cases. In order to maintain consistency of such words at the time of feature engineering without losing any information, case normalisation was implemented on the entire corpus.
- **Special Characters and Punctuation:** The occurrence of special characters such as newline, '(', numbers and punctuation were observed in the reviews. The corpus also contained words with hyphens and apostrophes such as "long-term" or "don't". We retained hyphenated words and words with apostrophe as they can be meaningful while all the other special characters and numbers were removed from the corpus using the custom regex pattern.
- **Stopwords Removal:** When the frequency of the words in the corpus was taken into account. It was observed that the top 100 words were corresponding to the stopwords such as "a", "the" which do not contribute towards the analysis of the sentiment. Thus, the stop words from the corpus were removed using the list of stopwords provided in assignment specification. This did not improve the accuracy of the model and therefore stopwords were retained back.
- **Trailing Spaces:** Trailing spaces were removed from each review in the corpus was removed for data uniformity.
- **Word Expansion:** Words such as "don't", "can't", "h've" have been observed in the corpus which were expanded to "do not", "can not", "have". Expansion didn't help in improving the accuracy of the model and therefore were retained back to their original state.
- **Removal of Least and Highest frequent words:** It was observed that 98% of the entire corpus is 5963968 and the highest frequency of the words observed was 252773 which was far less and therefore, highest frequent words were retained. 1% of the entire corpus is 60856.82 and the least frequency of the words observed in the corpus is 1. Therefore, words with frequency 1 are removed from the corpus.
- **Stemming/Lemmatization:** Stemming and Lemmatization were applied separately on the data in order to convert the words such as "shortly" to "short" or "lively" to "live" so that words with similar meanings are considered in the same way at the time of feature engineering.

The above preprocessing steps were applied on the product reviews and the data was passed to Logistic model for classification. Based on the accuracy of the model the preprocessing steps were changed. After few combinations the finest set of preprocessing steps were identified that gave the highest accuracy of the model.

The final preprocessing steps considered are:

- Normalisation and Tokenization
- Special Characters and Punctuations
- Trailing spaces
- Removal of Least Frequent words

The Corpus Statistics are as follows:

- Total number of product reviews : **50000**
- The size of the vocabulary: **58678**
- Total number of tokens of the product reviews: **6085682**
- Average number of tokens of the product reviews: **121.71364**
- The maximum number of tokens of the product reviews: **1015**
- The minimum number of tokens of the product reviews: **0**
- The standard deviation of the size of the product reviews: **111.575**

2.2 Feature Generation

In order to analyze the sentiments, feature generation plays a vital role. Feature generation in sentiment analysis includes converting the tokens into vectors so that machine learning algorithms can be easily applied on these vectors. N-grams of words such as unigrams or bigrams or trigrams are considered during feature generation. The following feature generation techniques are used for Yelp review analysis:

- **TF-IDF Vectorizer:** It estimates the term frequency (TF) of the word that is the occurrence of the word in the document and also estimates the inverse document frequency (IDF) of the word by calculating the log ratio of the number of documents where the word occurred to the total number of documents (Ramos et al., 2003). It then assigns a weight to the word based on the TF and IDF. We made an attempt to use this vectorizer with bigrams and trigrams. It was observed that the bigrams gave the best results.
- **Word2vec:** It is a method that is used to provide vector representations of the words known as word embeddings. It generates low dimensional vectors for the words of very large datasets. As we have labelled and unlabelled data, we generated a word embedding model using the entire dataset. We made an attempt to use this vectorizer to generate features for analysis but after applying this on the models didn't improve the accuracy of the model (Mikolov, Chen, Corrado, & Dean, 2013).

The above feature generation techniques were applied on the machine learning models.

3 Machine Learning:

The chart represents the architecture followed for the best model generation.

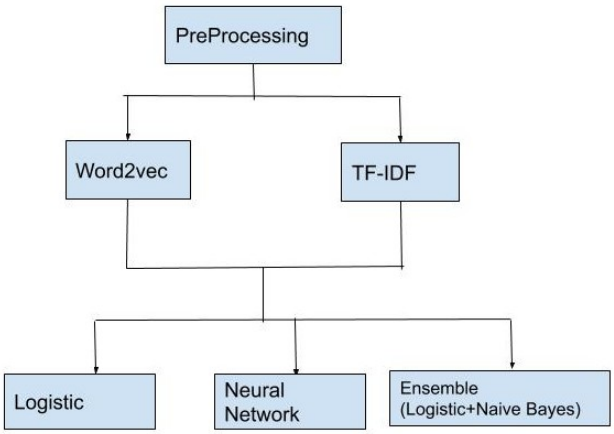

Figure 3: Architecture for model generation

Figure 3 represents the architecture followed in order to identify the best model for analysis.

The preprocessed data without special characters, numbers, punctuation and less frequent words is feature engineered using Word2vec and TF-IDF. These generated features are then passed to the three models for classification.

As we have labeled as well as unlabeled data as a part of the training data, we use the Pseudo labeling method of semi-supervised machine learning. In this method the model is first trained on the labeled data and then the labels of the unlabeled data are predicted. The instances predicted with more than 95% probability are considered correct. These instances are then concatenated with the labeled data and the model is trained with new concatenated data. This process is repeated until we see there is no significant change in the accuracy of the model. This returns the final trained model for predictions.

We have applied the semi-supervised method on the following models.

3.1 Model 1: Logistic Regression

The general use case of logistic regression is for binary classification. The introduction of multi-class parameter makes it compatible to work with multi-class labels. This model classifies each review with their respective polarities.

Logistic Regression was applied on the two feature generation methods. The labeled data was split into 80% train data and 20% test data. It was observed that using TF-IDF feature generation method the accuracy of the logistic model was around 61% on the labeled data while word embedding provided only 58.34%. The TF-IDF logistic model was then used for predicting the unlabeled data and the pseudo label method of semi-supervised machine learning was applied on it.

The above figure represents the classification report of the multinomial logistic regression. It can be observed that the model classified 73% of the instances correctly. The recall represents the

	precision	recall	f1-score	support
1	0.73	0.80	0.76	2281
2	0.53	0.50	0.52	2002
3	0.53	0.48	0.51	1982
4	0.55	0.55	0.55	1971
5	0.74	0.77	0.75	2290
accuracy			0.63	10526
macro avg	0.62	0.62	0.62	10526
weighted avg	0.62	0.63	0.63	10526

Figure 4: Classification report Logistic

percentage of positive instances. In order to improve the precision and recall further parameter tuning was done.

3.1.1 Parameter Tuning

The hyperparameter tuning was done with the help of GridSearchCV where the parameters considered were L1 and L2 for penalty, ovr or multinomial for multi-class, C values ranging from 0-4 and the GridSearchCV creates k fold cross validations and identifies the best model. It returned the best model with parameters as multi-class=multinomial and C=1.

3.2 Neural Network

The Neural Network (NN) consists of input layer (take the input data), hidden layer(processes the data using multiple neurons) and an output layer (provides the output of the model). The multinomial classification can be easily computed using NN.

The labeled data was split into 80% train data and 20% test data. It was observed that using TF-IDF feature generation method the accuracy of the model was around 73% on the labelled data while word embedding provided only 59.46%. The TF-IDF NN model was then used for predicting the unlabelled data and the pseudo label method of semi-supervised machine learning was applied on it.

	precision	recall	f1-score	support
1	0.84	0.90	0.87	4157
2	0.55	0.51	0.53	2043
3	0.53	0.53	0.53	1983
4	0.57	0.53	0.55	2042
5	0.86	0.87	0.86	4134
accuracy			0.73	14359
macro avg	0.67	0.67	0.67	14359
weighted avg	0.72	0.73	0.73	14359

Figure 5: Classification report Neural Network

The above figure represents the classification report for the Neural networks. It can be observed that class 1 and class 5 are classified with 80% and above precision. The average recall of the model is 67% which indicates the percentage of true positives.

3.2.1 Parameter tuning

The hyperparameters such as "activation" was tuned from sigmoid to softmax as the sigmoid is suitable only for binary classification and softmax provides good results for multiclass classification as it generates only one hot encoding vector as the output which sums up to 1 (Malik, 2018).

3.3 Ensemble Classifier

In Ensemble classifier, multiple machine learning algorithms can be combined, and improved classification results can be obtained. One of the simple and effective type of ensemble learning is Voting classifier. Here, two sub models such as Logistic Regression and Naive Bayes are built, fit and predicted. The Naïve Bayes provided the second highest accuracy amongst Adaboost Classifier, XGBoost Classifier, RandomForest. The sub-models are then combined and each model votes on what the output label might be. The Voting Classifier takes two parameters such as the dictionary of built models and the voting parameter being soft as the it would produce the probabilities of the predicted values from the Voting Classifier model. The predicted probabilities can be later used to validate its own prediction so that the unlabelled data can have appropriate labels.

4 Discussion of model difference

The data was divided into 80% training data and 20% test data and all the three models were applied. The models were evaluated based on their accuracy. The bar chart represents the best accuracy obtained for all the models with TF-IDF vectorizer and Word2vec.

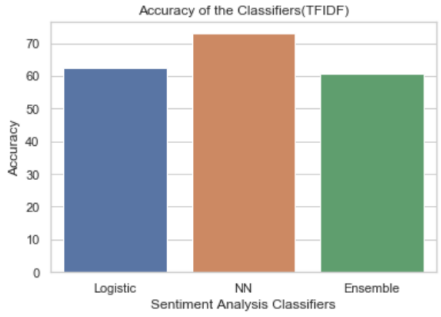

Figure 6: Accuracy of the models with TF-IDF features

From the figure, it can be observed that Neural Network model provides the highest accuracy of around 72% on the training data (labelled + unlabelled). It was observed that the accuracy of the models for word embedding was relatively low as compared to TF-IDF vectorizer. The logistic model can be considered as one layer of the NN. The Logistic model doesn't provide any loss function penalization parameter while the NN provides binary cross entropy and categorical cross entropy that penalizes the model for overfitting. The NN is also compared to ensemble model that consists of Logistic and Naive Bayes where the NN models outperforms based on the accuracy. Thus, we chose Neural Networks with TF-IDF and bigrams as the best model for the Sentiment Analysis classification.

5 Experimental Setup

- For the Logistic regression parameters C, solver, and multi-class are set to 1, sag and multinomial.
- For the Neural Networks the parameters activation, loss, metrics and optimizer are to be initialized to softmax, categorical crossentropy, accuracy and adam.
- The input dimension for the input layer should be the number of maximum features required.
- For the ensemble model the parameters are model dictionary and voting where model dictionary consists of all the models to be assembled and voting is set to soft.
- For all the above models the training and test data split is 80% and 20% respectively.
- The accuracy is evaluated based on the predicted values and the test values.

6 Experiment Results

Feature Generation Method	Model	Accuracy
Word2Vec	Logistic	0.5834
	NN	0.5947
Tf-IDF	Logistic	0.625
	NN	0.73
	Ensemble	0.6078

ccuracyyy.PNG ccuracyyy.PNG

Figure 7: Accuracy of the models with TF-IDF and word2vec features

7 Conclusion

- For the text analysis, as the number of tokens and the features increases the accuracy of model also increases.
- Different combinations of pre-processing steps should be applied on the model to get the best optimized model.
- The text classification model performs the best when the vectors generated by feature engineering considers the importance of the tokens in the entire corpus.
- Neural Network best suited for the Yelp reviews classification with an accuracy of 62.23% on the test data provided.

8 Future Scope

Due to the time limit ensemble with Neural networks was not implemented though we thought it would have boosted our accuracy of the model.

References

Malik, U. (2018, Oct). *Creating a neural network from scratch in python: Multi-class classification.* Stack Abuse. Retrieved from https://stackabuse.com/creating-a-neural-network-from-scratch-in-python-multi-class-cl

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pandey, P. (2019, Aug). *Simplifying sentiment analysis using vader in python (on social media text).* Analytics Vidhya. Retrieved from https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-

Ramos, J., et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133–142).