FIT5149 S1 2019 Assignment 2 REPORT VERSION 1.0 GROUP No. 69

SENTIMENT CLASSIFICATION OF YELP PRODUCT REVIEWS

SUBHASISH SARKAR
AJAY GANAPATHY
RISHABH KOCHHAR

27TH OCTOBER, 2019

Abstract

We attempt the identification of sentiments expressed in text documents, obtained from customer reviews on the popular review website Yelp. The sentiments are categorized into one of five categories. Using these review texts as training data we train machine learning models which perform automatic sentiment classification on unseen review data, with close to 61% accuracy. We provide a detailed analysis of the features we have included to train the models, and have made comparisons of 3 models.

CONTENTS

1. Introduction and Problem Setup

NLP and Sentiment Analysis

Natural Language Processing, or as it is commonly referred to in academia, NLP, is a field which deals with processing, and analysing of language as spoken or written by humans at the computational level. It is the intersection of Computer Science, Machine Intelligence, and Linguistics. Broadly speaking, its main motivation is to develop systems, or algorithms which "understand" natural language in order to perform various human like tasks like translation, question answering, or understanding context-based sentiment.

Sentiment-analysis is a sub-field within NLP, which deals with the contextual mining of text in order to identify and extract subjective information regarding a specific domain in the given material. Its main aim is to help businesses understand the social sentiment of their brand, product or service.

A very important task in sentiment analysis is the classification of the polarity of a given text at the document, sentence, or a feature/aspect level - whether the expressed opinion is positive, negative, or neutral. There are some advanced sentiment classification tasks which also looks at emotional states such as *happy*, or *sad*.

In this data analysis challenge however, we are interested in developing an automatic sentiment classification system that relies on machine learning techniques to learn from a large set of product reviews provided by Yelp. The levels of polarity of opinion we consider here are: *strong negative, weak negative, neutral, weak positive, and strong positive*. Thus, it is a multiclass sentiment classification challenge.

In this report we shall go through an overview of all the steps which we have followed prior to the analysis task. This includes:

1. The exploratory data analysis that we have performed on the text data, to extract features which we have later used in the model for the classification task,
2. The process of cleaning of data which included removing stop-words and punctuations among others
3. The development and comparisons of models
4. Further scope of improvement

2. Pre-processing and Feature Generation

2. 1 Data Pre-processing

The data was given to us in a CSV file, where the ***text*** column contained the documents which held the product reviews. These were unclean and contained characters for newlines, tab-lines, numbers, punctuations, special characters, emojis, among other such trivialities which would make it difficult to input directly into a machine learning model. As such we have performed certain pre-processing tasks to make it easier for the model to interpret the data. They are listed below:

2.1.1 Case Normalisation

We have ensured that all the words have been converted to lower case, in order to reduce feature redundancy. We do not want “Man”, and “man” to be two separate features.

2.1.2 Removing certain stop words

Stop words occur commonly in natural language as a means to connect sentences, however in our context they do not convey any useful information about the sentiment of the sentence itself. As such we have removed them from the text. There are certain stop words like ‘not’, ‘wouldn’t’, ‘couldn’t’, etc., which hold semantic value, and as such we have kept them in the text.

2.1.3 Handling Contractions

Contractions are a part of everyday colloquial language; however, they make it difficult of a model to interpret the semantic meaning behind the text. As such we have handled contractions by expanding them to their base forms. For example, “can’t” was replaced with “cannot”, and so on.

2.1.4 Removing Punctuations

Punctuations also do not add any semantic meaning to text in terms of sentiments. As such they are removed.

2.1.5 Removing Newline and Tab characters

When people write, certain linguistic tendencies like thematic unity, make people separate their text into paragraphs, and use tabs and line spaces. These are read in the machine in the form of special characters like \r, \n, and \t. These are also removed.

2.1.6 Replacing numbers with a "NUM"

Digits are also redundant when we want to make an analysis of the sentiment of a text block. As such they are not entirely removed, but have been replace by a custom string called "Num" to ensure that occurrences of words like "five stars" etc, are still preserved.

2.1.7 Handling multiple consecutive spaces

Once we remove special characters, punctuations, etc., there are some occurrences of multiple consecutive spaces. These have been replaced with a single space.

2.1.8 Tokenization

Once the raw text has been cleaned, it needs to be tokenized. This is a process where words are extracted from a sentence, and occur singularly in the form of a list of words. This will allow us to process each word individually if we want to perform n-gram analysis or do POS tagging.

2. 2 Feature Extraction and Engineering

Once the data has been tokenized, it can now be input to a model to run predictions. However, to increase accuracy, we find features from within the text which allows the model to make better predictions. Features which have high correlation with the target variable will be essential in making the model have higher prediction accuracies, since these are latent features not quite evident from the text itself. As such we have extracted the following features:

2.2.1 Bi-grams

Once the data has been converted into tokens, we can consider co-occurring words as a feature for our model. Co-occurring words may sometimes have semantic meaning, for example, the term, 'not bad' has a positive meaning, whereas both 'not' and 'bad' have negative connotations. We have thus preserved such co-occurring word.

2.2.2 POS Tagging

Parts-of-Speech (POS) tags denote the grammatical tag of a word with respect to other words in the text. They represent whether a word in the sentence is a verb, noun, adverb, adjective, etc. The NLTK package in python has an inbuilt POS tagger which tags words for us. Relevant POS tagged words to identify semantic meaning for us are – Adjectives, Superlatives, Adverbs, which allow the model to make a judgement on the opinion expressed in the text. As such we have only kept these POS tagged words to be input to the model.

2.2.3 Other Quantitative Features

The other features which we have included are:

1. *Word counts*, which indicate how many words a certain review has
2. *Number of Capital words*: which could indicate the sentiment expressed in the text, since negative reviews tend to be longer than positive or neutral ones
3. *Number of Exclamation and Question Marks*: which could also indicate extreme sentiments like anger, or excitement.
4. *Number of Unique Words*: Positive reviews and negative reviews tend to have opposite number of unique words. People in general tend to use a larger vocabulary to express anger than they do otherwise
5. *Number of Emojis*: Emojis are generally used in everyday text to express emotions. Emojis are a good indicator whether the sentiment expressed is positive or negative. As such we have made a count of the number of emojis in the raw text.
6. *Word Vectors*: This allows all words to be represented in the form of numerical vectors stored as a dictionary pairing of words as keys and their index as values. This allows us to make a vocabulary from which the model may extract each word w.r.t to its value in the vocabulary dictionary. This allows the text to be transformed into a machine-readable format.

2.2.4 TF-IDF Features

TF-IDF, which stands for term frequency—inverse document frequency, is a scoring method widely used in information retrieval (IR) or summarization. TF-IDF is intended to reflect how relevant a term is in a given document.

The idea behind it is that terms which occur frequently in a document should boost its relevance and give us an idea as to what the document is about, more so than those words which occur less frequently. Simultaneously, if the word also occurs frequently across all documents, the word might not be relevant to a specific document, but rather is just a common word in general. This is called as the Document Frequency.

As such those words which occur frequently inside a document but are rare in the entire corpus, are words which have the highest semantic meaning in the sense of a given document. Thus, the term-frequency multiplied by the inverse of the document frequency is used as a metric to identify and score the most relevant words.

This process can be summarized from the following:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 1: TF-IDF Formula

The approach we are taking here is a Bag-of-Words approach, where we are unable to garner the sequence based contextual meaning of words, but rather, are looking at a what collection of words each document has, and making a prediction of its sentiment based on the frequencies of these words in the document, and in the corpus as a whole. This is a very rudimentary approach to sentiment analysis and we lose a lot of semantic value which comes with approaching the problem in a broader manner of contextual learning.

3. Model Building and Validation

Following the extraction of features from the text, and making sure that it is in a machine-readable format, we build the different classifiers and check their accuracies with the available text. For this classification task, we have considered 3 different models – Logistic Regression, Multinomial Naïve Bayes, and an Ensembled and Stacked Model. They are explained below.

3.1 Logistic Regression

Logistic Regression is the most commonly used classification technique. It uses a sigmoid activation function to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events. These probabilities depend on the feature values of each observation in a set of observations, and are validated based on the cross-entropy between predicted and true values. In our case since we have a multiclass classification problem, we use the categorical cross-entropy loss and aim to reduce it.

The parameters used in our logistic regression model are as follows:

1. Penalty: It is used to determine the normalisation used in penalization. We have used the default L2 normalisation as our penalty in the model.
2. C: it is the inverse of the penalisation strength. It must be input as a positive float. We have experimented and found that the optimal value of C is 1.01.
3. The rest of the parameters were set to their default values.

3.2 Multinomial Naïve Bayes

One of the other popular text classification algorithms used, is the Multinomial Naïve Bayes model, which is the multiclass implementation of the Binomial Naïve Bayes model; both of which were built upon the core concept of Bayes Theorem.

It predicts membership probabilities for each class; such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as Maximum A Posteriori (MAP).

The 'Naïve' part of the model refers to its naïve assumption that every word in a sentence is independent of the other ones. This means that we're no longer looking at entire sentences, but rather at individual words. This is not a good model for sequence-based contextual learning, but it generally does a good job of classifying simple sentiments using a bag-of-words approach.

We have set the Alpha parameter to 1.01 which denotes the Laplace smoothing, which increases the probabilities of all bigrams in the non-maximum likelihood equation by 1.01 to make everything non-zero.

3.3 Model Ensemble and Stacking

A good way to boost prediction accuracies of our models is to use model ensembling, and model stacking.

3.3.1 Model Ensembling

The main idea behing building a model ensemble is to combine the predictions of two different models which run on the same data, and use the weighted average of the two models to make our final predictions. Since Logistic makes better predictions than the Naïve Bayes model, we have given the predictions from the Logistic model more weight than that of the

Naïve Bayes one. Doing this, we are ensuring that we reduce the bias and variance of each model.

3.3.2 Model Stacking

Model Stacking, or Meta stacking, is a technique of ensembling which uses the predictions from a set of base learners and the true labels to make predictions using another classification algorithm usually called the Meta classifier.

Base Learners and Meta Learners are the normal machine learning algorithms like Random Forests, SVM, Perceptron etc. Base Learners try to fit the normal data sets whereas the Meta learner fit on the predictions of the base learners.

In our case, we have used the Ensembled model, the Logistic model, and the Naïve Bayes model as the base learners and another logistic model as the meta classifier.

3.4 Discussion of Model Differences

While both the Logistic regression model and the Naïve Bayes model are used for classification, their core functionalities are different. The learning mechanism is a bit different between the two models, where Naive Bayes is a generative model and Logistic regression is a discriminative model.
A Generative model models the joint distribution of the feature X and target Y, and then predicts the posterior probability given as P(y|x).
While a discriminative model directly models the posterior probability of P(y|x) by learning the input to output mapping by minimising the error.

4. Experimental Setup

The semi-supervised experimentation steps included the following:

1. Pre-process the labelled and unlabelled datasets
2. Removing common rows from the unlabelled dataset, as around 45,000 observations from the labelled dataset are included in the unlabelled data
3. Develop and train models on the labelled dataset first, and checking the accuracy score
4. Run the most accurate model developed in the last step on the unlabelled dataset, to generate pseudo-labels based on the highest-class probabilities.
5. We then take those observations which have class probabilities over 0.8 and add them, along with their labels to the labelled data
6. We then develop and train the models on this larger labelled set, and check the accuracy scores.
7. We use the most accurate model to generate labels from the test file.

5. Experimental Resuts

We have summarised the results of our experimentation here:

Model Name	Labelled	Labelled+Unlabelled	Test
Logistic Regression	0.6102	0.713	0.609
Multinomial NB	0.577	0.686	0.573
Ensembled Model	0.60	0.709	0.602
Stacked Model	0.60	0.710	0.604

Figure 2: Table with models and their accuracies on different sets of data

As is evident, we have used the Logistic Regression model trained on the larger training data to make the final predictions on the test set on Kaggle.

6. Conclusion and Scope of Improvement

We have conducted sentiment analysis of a set of Yelp Reviews, by pre-processing the data and converting it to machine-readable format, following which we have experimented with 3 classification models, and obtained their accuracy scores. The most accurate model was used on the test data and obtained a score of around 61% accuracy.

Further scope of improvement for this would be to conduct the sentiment analysis from a contextual perspective, rather than a BOW approach. This would include the usage of word embeddings like Word2Vec, GloVe.

Pre-trained models like BERT, Elmo, XLNet, can be used to fine tune Deep Learning models like LSTM and Recurrent Neural Networks for downstream tasks like sentiment analysis, thus allowing us to streamline the context-based learning approach, and give higher accuracies.

However, as it can be seen, using simple models also give us accuracies close to these state-of-the-art techniques. Thus, a call can be made based on the requirements of the project as to which techniques need to be used.

References

Please refer to the Jupyter Notebook file for a full list of the references used.