MONASH University

FIT5149 – ASSIGNMENT 2 GROUP NO. 19

SENTIMENT CLASSIFICATION ON PRODUCT REVIEWS

ANUP RAJESH - 29933145
RACHANA KALACHETTY - 29562449
MOHAMED HAYATH MOHAAB HASSAN - 29626889

OCTOBER 28, 2019

Contents

1. Introduction

Project description:

Sentiment analysis is a process of extracting and understanding the sentiments defined in the text document. The explosion of data in the various social media channels and product review channels like yelp, Zomato etc has given consumer new way of expressing their opinion on a particular product, person and places. The user opinion is always in the form of textual information. Per day millions of textual message data is sent over social media or online shopping website. Investigating and analysing the sentiment of the opinion is a very critical task to perform. Sentiment analysis is also providing a business intelligence which can be used to make good impactful decision. Sentiment analysis and sentiment classification are the two methodologies used in opinion mining. In this assignment, we stick with sentiment classification. Sentiment classification indicates the sentiment orientation by assigning the class labels to the document or segment. Sentiment orientation is a kind of text classification that classifies text data based on the sentiment orientation of opinion.

The assignment provided is about Fine-grained sentiment classification for product reviews. The dataset given to us is the product reviews provided by yelp. The training dataset is divided into parts: labelled data(50k) and unlabelled data(600k). The test data set consists of 50k rows.

The performance of the model was calculated based on the F-1 score as shown below,

$$\text{Accuracy} = \frac{\textbf{number of correct predictions}}{\textbf{number of all predictions}}$$

Goal of the project: The goal of the project is to develop a semi-supervised machine learning algorithm which can accurately classify sentiments based on the dataset provided to us. Firstly, we train the model on labelled data accurately. This model is used to predict on unlabelled dataset to generate pseudo labels. We then use the labelled data and the unlabelled data together to train our model and predict on test data.

Team work allocation: We had split the tasks between research and implementation. Initially the research with neural nets and different word embeddings for sentiment analysis. However, due to a consistent failure in the implementation of neural nets, we had to pivot to the conventional methodologies which yielded a better result as the assignment progressed.

2. Pre-processing and feature generation

2.1 Pre-processing

The product review data provided was given in comma separated value format, with the review text and the label of each review. We were also provided unlabelled data for which we generated the pseudo labels. Finally, we had to test the model on the test data. This is diagrammatically explained as shown below,

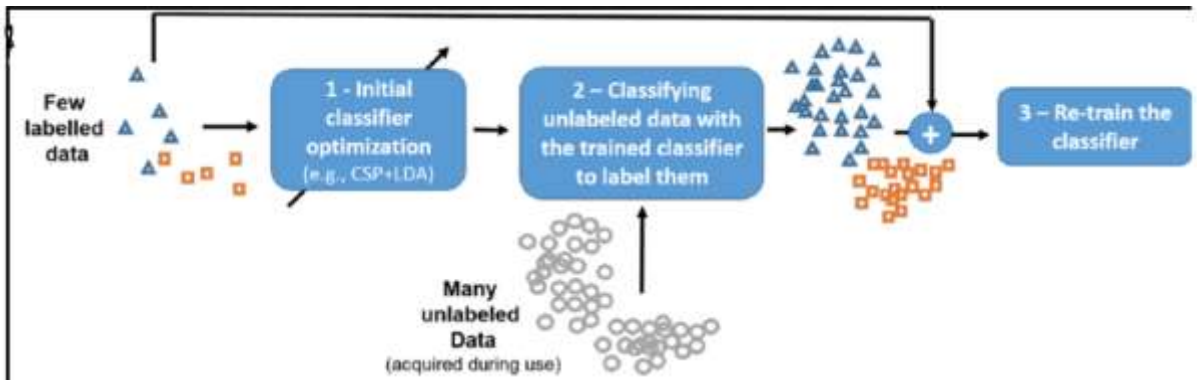

Figure 1 :- Semi supervised learning approach

Since the data given was unclean, some data cleaning and pre-processing was required before preparing the data for the sentiment analysis classifier. The programming language used to achieve the above was Python 3.7, specially the 'Natural Language Tool Kit' package and the TextBlob for Python. The following steps were undertaken while pre-processing the data:

- Capturing Emoticons

Since the reviews were posted by internet users, there was a plethora of emoticons used which help express the feeling of the user through a collection of punctuation characters. It was necessary to separate and identify the 'happy' emoticons from the 'sad' ones as they deliver the user's sentiment. Some of the emoticons we extracted are: -

Happy emoticons: {':D', ';-)', ':o)', 'X)', 'xD', ':O', ';)', 'XD', 'Xo', ':-)', ':)', ':-o', 'xd', ':-D', 'XO', 'xo', ';o'}

Sad emoticons: {':-/', '-_-', ':[', ':-(', ';-(', ':/', ":'(", ';(', ':-[', ';-/', ":'/", ':|', ':(', ':-|', ';/'}

- Extracting Uppercase Words

Sometimes, the words in a corpus are written with a sentiment backing it. One case of that is when someone writes something in all caps. For example, the labelled training data had the following review –

At row 23, “Great food. We got there around 2:40 and paid for the brunch buffet, but ended up being there for more of the dinner portion. The only bad part of it were the LIVE BIRDS flying around eating food off of the floor!!”.

In this review, it is clear to see that the reviewer was extremely dissatisfied by the birds flying around the open area and eating their food. These instances could prove to be the difference between a weak negative review and a strong negative review and hence, need to be preserved.

- Extracting Prices

Since the corpus includes product reviews, there are several mentions of the price that the reviewer paid. Usually, in a review if the price is mentioned, its either mentioned in the context of the price being unbelievably low (positive sentiment) or if its incredibly high (negative sentiment). The examples of price expressing sentiment are given below:

Positive – *$5 for lunch, where a part of it goes for charity, well that made my day good.*

Negative - *…. So, we paid $1000 for nothing but that shit, fucking crazy.*

As seen in the examples above, the price shows the sentiment behind the review and could be a powerful feature for sentiment analysis.

- Removing Links

Since the data is extracted from Yelp, we may have several URL links. These characters are junk for the sentiment analyser and hence needed to be removed.

- Case Normalisation

Since the data will be converted to a TF-IDF vector, to gain the true term and document frequency, case normalisation was implied to the data. The data was subjected to case normalisation where the entire corpus was converted to lowercase, except the all-uppercase words extracted earlier.

- Removing Control Characters

The data contained certain control character such as the newline character and the tabspace character. These characters are junk for the dataset and produce noise inside the dataset. These were removed as a part of the pre-tokenisation process and replaced with a whitespace.

- Removing Numbers

Digits were removed from the corpus except for the ones with the prices. The standalone digits provide no sentiment and hence are junk characters and tokens for the text. Since the purpose of the pre-processing is to extract the best features, it is necessary to eliminate all unnecessary features from the corpus and keep the important ones.

- Removing Whitespaces

Any instance where there were 2 or more whitespace, was corrected to just one whitespace to remove blank tokens from appearing in the text. A regular expression was used to achieve this objective.

- Removing Stop words

Stop-words are the most common words in a language used in almost every instance of text. Such words hold little meaning and have high frequency in the texts which alter the values of the TF-IDF vectors. In this project, certain stopwords provided by the nltk library in Python were of significance. Words such as ‘not’, ’no, ‘almost’, ‘always’, etc were of value in the terms of sentiment analysis and hence, a modified list of stop words was created to preserve these special stop words.

- Removing Punctuations

The punctuations provide little to no value and hence, they must be removed. The punctuation was chosen from the string library in Python and removed from all text. The only exception to this process was the contractions extracted earlier.

2.2 Feature generation

Feature generation is an extremely important step towards building a successful sentiment analyser and in the identification of the features to be used as predictors.

The following steps were performed during feature generation:

- Unigrams
- Bigrams
- Tf-idf

3. Models

We performed a two-method approach to building our models,

- The first approach is illustrated in the figure below,

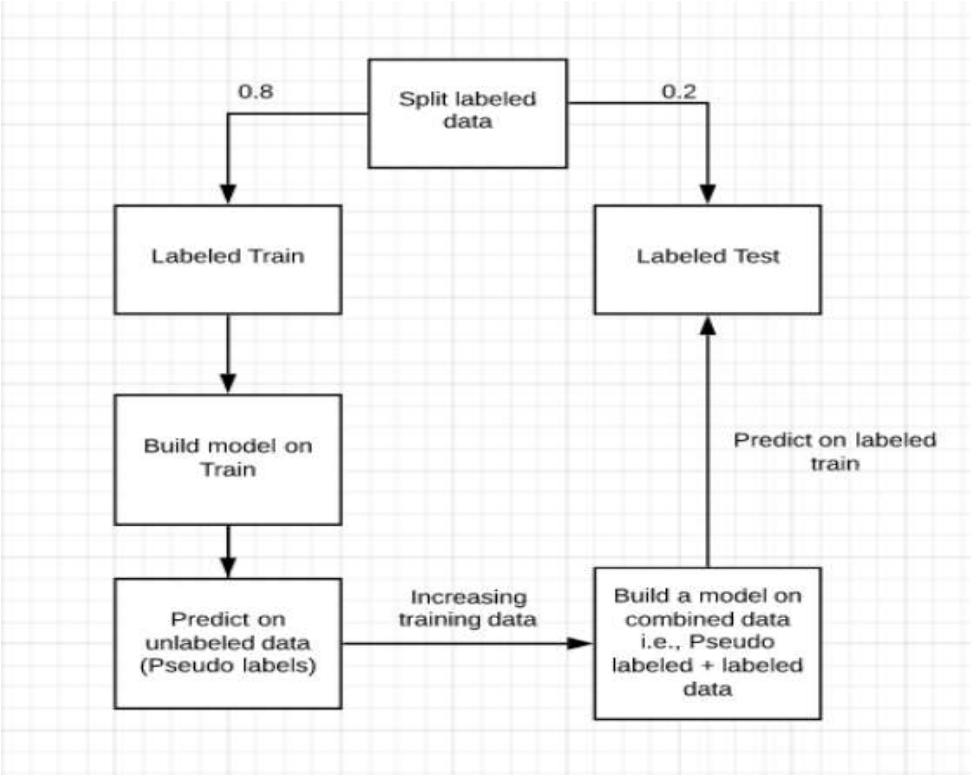

Figure 2: - Building the model without considering the probability threshold

In this approach, we do the following

- Split the labelled data to train and test.
- Build the model on training labelled data.
- Generate pseudo labels on unlabelled data.
- Predict and evaluate the model on test labelled data.
- Predict on test data for final evaluation.

The second approach is illustrated below,

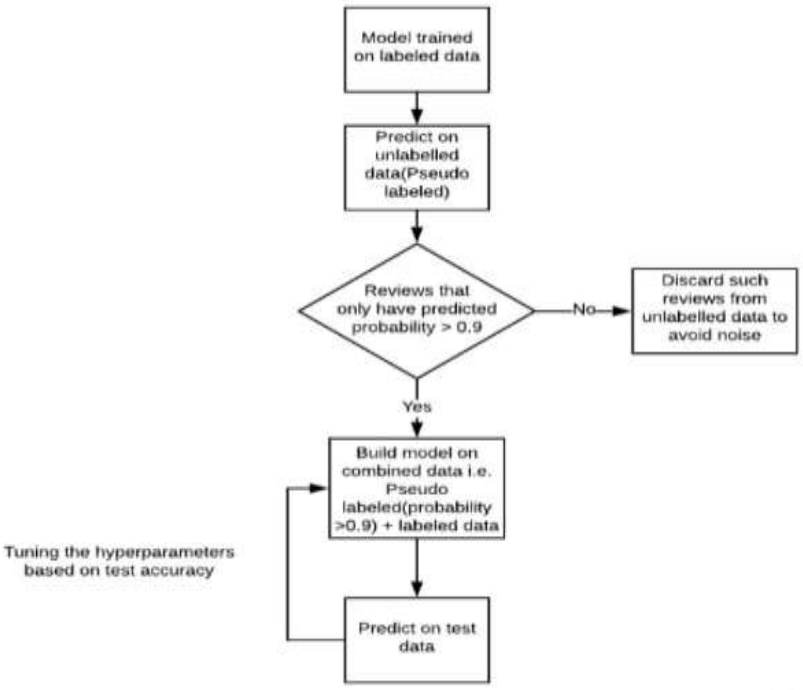

Figure 3: - Building the model considering probability threshold

In this approach, we did the following

- Split the labelled data to train and test.
- Train the model on labelled train and generate pseudo labels for unlabelled data.
- Retrieve only those data points in the unlabelled data with a probability confidence greater than 0.9. We discard the rest of the unlabelled data.
- Combine the labelled train data with retrieved pseudo-labelled data and fit a model on this data.
- Test it on the labelled train data to evaluate the accuracy.
- Predict on the final test data to finalize the accuracy.

We found that the first method yielded poor results compared to the second method.

This was followed by selecting a feature set to run the models. Below were the selected feature sets,

Feature set 1:

1. Lowercase
2. Strings but remove numbers
3. Remove punctuation
4. Remove stopwords
5. Lemmatization
6. Extracting emoticons

Feature set 2:

1. Lower case
2. Strings but remove numbers
3. Remove punctuation

4. Remove stopwords
5. SnowballStemmer
6. Unigrams

Feature set 3:

1. Remove digits
2. Remove single and double quotes
3. Normalizing
4. Removing white spaces
5. Removing control characters

The following table summarizes our results,

Feature Set 1	LogisticRegression	0.58170
	MultinomialNB	0.51744
	SGDclassifier	0.39458
	Ensemble	0.52444
	SVC	0.55586
Feature Set 2	LogisticRegression	0.59885
	MultinomialNB	0.42512
	SGDclassifier	0.49482
	Ensemble	0.57582
	SVC	0.56551
Feature Set 3	LogisticRegression	0.60806
	MultinomialNB	0.52186
	SGDclassifier	0.60900
	Ensemble	0.60933
	SVC	0.60033

Table 1: - Model performance on different feature sets.

4. Experiment setups

Model	Tuned parameter	Accuracy
Logistic Regression	C = 4.1	0.60806
Multinomial Naïve Bayes	Alpha = 0.04	0.51733
Linear SVC	C = 4	0.60033
Ensemble(Logistic+NaiveBayes+SGD)	C = 2.6, alpha = 0.07, weights = [0.75,0.15,0.1]	0.60933

Table 2: - Model performance after fine tuning on third feature set

As majority of the models performed well on the third feature set(Table 3.1), we will use that as a basis to run our model.

As seen from the table above, the models are ranked as shown below,

1. Accuracy by Logistic regression: 0.60806

2. Accuracy by Multinomial Naive-Bayes: 0.51733

3. Accuracy by SVC: 0.60033

4. Accuracy by using Ensemble: 0.60933

5. Experimental results

Thus, after using an Ensemble of Logistic regression, Naïve Bayes and SGD, and fine tuning Logistic Regression's cost function to 2.6, Naïve Bayes alpha to 0.07 and setting weights of 0.75 to Logistic regression, 0.15 to Naïve Bayes and 0.1 to SGD, we achieved an accuracy of 60.933 ~ 61 on the test data.

6. Conclusion

In conclusion, we can prove that after fine tuning our parameters, Ensemble performs the best as it has a really good test accuracy in comparison to other models used.

References

https://ieeexplore.ieee.org/document/7975207
https://scikit-learn.org/

https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/

—