SENTIMENT CLASSFICATION ON PRODUCT REVIEWS

Jaimon Thyparambil Thomas
Roopak Thiyathuparambil Jayachandran
Vidhu Krishnan Unnikrishnan

OCTOBER 28, 2019

Table of Contents

1 Introduction

Opinion mining or Sentiment analysis refers to the process of identifying the emotional state from a piece of text by using Natural Language Processing (NLP) techniques.
This is achieved by building machine learning algorithms to determine the opinion polarity contained in the text. Implementation of sentiment analysis has gained traction in the recent years in fields of both industry and education. Training chat bots, adaptive customer service, tracking the overall customer satisfaction of customers are few of the benefits of sentiment analysis.

The aim of this data analysis challenge is to build a sentiment classification system for product reviews that bank on machine learning techniques. The different type of machine learning techniques is broadly classified as supervised, semi supervised and unsupervised methods.

The system must be capable of classifying the large amount of reviews into five opinion polarities using the given limited set of labelled reviews and larger set of unlabelled reviews. The sentiment labels are: strong negative, weak negative, neutral, weak positive, and strong positive.

2 Pre-processing and feature generation

The first hurdle in text analysis is handing of semi-structured and unstructured text. The conversion of such texts to structured texts are performed prior to running any analysis tasks.
All the text analysis tasks share the same set of basic text pre-processing steps, which include the following:

- Tokenization:
  - Representation of texts as a set of characters
- Case Normalization
  - Conversion of words into either upper or lower case.
- Removal of stop words
  - Words that are too common or carry very little meaning are categorised as stop words.
- Stemming and Lemmatization
  - Conversion of words into their base or simplest form
- Segmentation of sentences
  - Breaking of sentences to decide where a sentence begins and ends

The key aspect here is to select which all among the above to be selected for the given task.

2.1 Pre-processing

In our system during model selection most or all of the basic steps of pre-processing were implemented to see the impact it had on the model accuracy.

2.2 Feature generation

Once the pre-processing is completed the next step is developing a classification system to perform feature generation. Most text mining and NLP algorithms requires that the text be converted to some form of numeric representation (e.g.: SVM). The most common type of structured representation of text is the vector-space model – where the text is represented as vectors with its elements indicates frequency of the words within the text. The order of the words in the text document are assumed to have no importance and this assumption is known as Bag-of-Words.

Some algorithms prefer the to represent the documents as weighted vectors instead of using multiple vector representations. For example, the term frequency can be replaced with a weighted term frequency to denote the importance of the word in the document. The most popular such scheme is TF-IDF. Term Frequency-Inverse Document Frequency weighting approach gives importance to the number of times a word occurs in a document and the number of documents that contains the words.

3. Models

3.1 Model 1 – Logistic Regression

Logistic Regression Classifier is a type of Supervised Machine Learning algorithm that uses a decision/activation function to transform the predictions. Due to which it is not possible to identify the prediction as a linear combination of the inputs like in case of linear regression.

3.2 Model 2 – Support Vector Machine

SVM is an algorithm applied on vectors that belong to a given category and those that do not to find the best decision boundary between them. Vectors are number representation of coordinates in some space. Hence, to implement SVM we need the text to be transformed to vectors. This also defines the caveat for SVM. As the performance of the machine depends upon the vector representation and how much of the text information can be encoded in it.

Linear kernel is used for making the model as most of the text classification models are linearly separable. The linear kernel is good when there is **a lot of features**. That's because mapping the data to a higher dimensional space does not really improve the performance. In text classification, both the numbers of instances (document) and features (words) are large. The model was generated with initial set of labeled dataset having 50K records. Then this model was used to predict almost 1L records of unlabeled data and predictions were recorded. After that, both initial labelled and predicted data was combined to form a new training record with 150K records. This approach is called Pseudo labelling and is employed to improve accuracy when we have semi-supervised scenario.

Though accuracy of test increased from 56.6 to 73.2, but this model performed badly with the new set of training data provided in the moodle. This was because of overfitting of data.

Another approach we tried with SVM is using cross validation of the model with 5 folds. K-fold cross validation is an expensive approach where a loop is created and average value is calculated. After performing cross validation, the average accuracy was found to be 57%

3.3 Model 3 – FastAI

FastAI is a library that is used to simplify the implementation of neural networks. The training process for fastai library is structured around a Learner class that binds together a dataset, optimiser, loss function and a PyTorch model.
FastAI takes as input a DataBunch i.e., bind train, validation and test data to single data object. The databunch is then used to create language model which will then be used to build our classifier. We use the data to fine tune the pretrained language model available for fastai : AWD_LSTM (Long Short Term Memory) which then aids to build our classifier.

3.4 Discussion of model difference(s)

Accuracy Comparison:

Model	Accuracy% (with given test data)
Logistic Regression	**46.78**
SVM With Cross Validation	**56.6**
FAST AI	**65.8**

Complexity Comparison:

Model	Complexity
Logistic Regression	**Very less complex**
SVM with Cross Validation	**Less Complex**
Fast AI	**Highly Complex**

Pre – processing techniques used:

Model	Complexity
Logistic Regression	• **Tokenization** • **Case Normalization** • **Stop Words Removal** • **Stemming and Lematization** • **Segmentation** • **TF-IDF**
SVM with Cross Validation	• **Tokenization** • **Case Normalization** • **Stop words removal** • **Segmentation** • **TF-IDF**
FastAI	**Actual data fed to the model**

Any Tuning used:

Model	Complexity
Logistic Regression	**No tuning used**
SVM with Cross Validation	**Cross validation with k=5 used**
Fast AI	**Pseudo labelling technique used**

4 Experiment setups

Note: Environment used for all the experiments is Python 3.6

Library used:

- Logistic Regression
  - Pandas for reading csv
  - Pyspark - for parallel computing
  - Nltk - for data preprocessing like tokenizing, removing stop words etc
  - Sklearn - for generating models using logistic regression and also for pre processing like finding TF IDF values etc.
- SVM
  - Pandas for reading csv
  - Numpy for setting random seed
  - Nltk - for data preprocessing like tokenizing, removing stop words etc
  - Sklearn - for generating models using svm and also for pre processing like finding TF IDF values etc.
- FastAI
  - Fastai - For generating models using fastai
  - Pandas - for reading cs files
  - joblib from sklearn.externals for saving the model for future use

5 Experimental results

Initially we tried using Logistic regression to check the accuracy of the model. Using that we got an accuracy of 46.78. Then we tried SVM with cross validation which gave us a result of approximately 56%. Then we tried using fast ai after pre-processing and before pre-processing. For us the accuracy score was better in case when the raw data was passed as it is to fastAI without any pre-processing. Which gave us an accuracy of approx. 64%. Then we thought of doing pseudo labelling on first 50K rows in the unlabelled data, using the model that we initially created using the fastAI. Then we combined this pseudo labelled data with original labelled data. We used this new data set for training using fast ai. This semi supervised training model using the fastai algorithm resulted in an accuracy of a little over 65%.

6. Conclusion

Out of all the models that we used we were able to see that sentiment analysis prediction performed by fast AI gave us maximum accuracy. We can also see that when we did pseudo labelling for 50K unlabelled data and used that along with the labelled data for training there was an increase in accuracy. Which indicates that as the amount of data increases for training, the accuracy of prediction also increases. So in cases where the available labelled data is less, like in our case pseudo labelling is a good alternative to increase the amount of training data. As we can see in our model, when we did semi supervised learning using the pseudo labelling we could see an increase in overall accuracy.

References

https://confusedcoders.com/data-science/deep-learning/sentiment-analysis-with-deep-learning-and-fastai