Machine Learning Engineer Nanodegree

Capstone Proposal

Aditya Tandon
December 24th, 2017

Sentiment Analysis of Movie Reviews Using NLP

Domain Background

Wikipedia defines sentiment analysis as the process that "aims to determine the attitude of a speaker or a writer with respect to some topic". Automated Sentiment Analysis is the process of training a computer to identify sentiment within content through Natural Language Processing (NLP). According to Mechanical Turk, automated analysis can be almost as good as human analysis (or, "as good as it gets"). Analysing 20 documents can be done easily by reading it and not much of human effort would be required. But analysing 50,000 documents will require a lot of effort and labour charges. And by using Automated Sentiment analysis, it can be completed within few minutes. Here is the paper on which machine learning has been applied for the sentiment analysis of the tweets on twitter.

In this project, I have created a web application, which analyses the sentiments from the Rotten Tomatoes movie dataset by differentiating sarcasm, terseness, language ambiguity and many others using NLP and classification algorithm.

Problem Statement

The goal is to implement and train a classifier that can predict the likelihood of the type of sentiment (multi-class classification) by using Natural Language Processing and Machine Learning Methods. The tasks involved are the following:

1. Split the given training data into train and validation data.
2. Pre-process the data by converting the words as tokens and creating a dictionary.
3. Train a classifier with the pre-processed training data that can predict whether that review is a negative review or a positive one.
4. Compute accuracy of the model.

The final application is expected to be useful for determining the sentiments of the movie reviews in just a few seconds. Therefore, this project can be useful building later projects like building an automation sentiment analysis for Food Reviews or Analysis of Twitter tweets.

Datasets and Inputs

In this project, I will be using the unbalanced dataset from the Sentiment Analysis on Movie Reviews competition on Kaggle. In this dataset, I am given with the text data with phrases from the Rotten Tomatoes Dataset, along with the type of sentiment (negative, somewhat negative, neutral, somewhat positive, and positive)

The dataset consists of tab separated, i.e., data.tsv file containing the phrases and their associated sentiment labels. The data description for sentiment labels are as follows:

0 – negative

1 – somewhat negative

2 – neutral

3 – somewhat positive

4 – positive

Total data for training and validation: 156,060

Total Files of data: 1

Total training data: 117045

Total testing data: 39015

Count of Negative Phrases: 7,072

Count of Somewhat Negative Phrases: 27,273

Count of Neutral Phrases: 79,582

Count of Somewhat Positive Phrases: 32,927

Count of Positive Phrases: 9,206

Solution Statement

We have data with complicated expression like sarcasm, terseness, language ambiguity and many others. So, making the dictionary of all the words using CountVectorizer is the first step and after that using a Neural Network by optimizing the parameters using Grid Search, so as to prevent overfitting and increasing the accuracy. I will use python as the software language, Sklearn and Numpy libraries to train the model.

Benchmark Model

I will use the RandomForest model as my benchmark model and train and test the model on the given data. And after that compare the results with my final model, i.e., the Multilayer Perceptron Neural Network model using the logloss evaluation metrics.

Evaluation Metrics

I will use the logloss function evaluation metric for both the models. Each Phrase has been labelled with one true class. For each phrase I will calculate the predicted probabilities. The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{ij}\, log\,(p_{ij})$$

Project Design

I have summarised some of the tasks briefly at the problem statement section. Here are the expansions and extension of those task lists:

1. Since I don't need the PhraseId and SentenceId, I will use only the Phrase from the dataframe and split the data into training and testing data using train_test_split.
2. After splitting, I have to change the sentences into different tokens ,i.e., vectorizing and form the whole dictionary in order to train the model
3. Removing the stop words, i.e., the words that are used most commonly in a sentence. For eg. "*a, the, an, that*".
4. Vectorizing the data into unigrams and bigrams will ensure the model to understand the words as a meaningful sentence, eg., "It is no good" is a negative review, whereas, "No, it is good" is a positive sentence.
5. Transform the vectorized data using the TfidfTransformer.

6. Train the Mutilayer Perceptron, with some hidden layers in order to train the model as the data is multiclass.
7. Adding a pipeline in order to execute 2,3,4,5 steps and optimizing the parameters using GridSearch.
8. Getting the best parameters from GridSearch for optimizing the model.
9. Using the parameters fit the MLP classifier with the transformed vectorized data.
10. Transforming the testing data with TfidfTransformer.
11. Predicting the values using the trained model.
12. Checking the accuracy using the logloss function from the metrics module of Sklearn Library.