Natural Language Processing (NLP)
Final Project

IST 644 FALL 2021

Contents

Introduction

The purpose of this final project of Natural Language Processing is to experiment the learnings that are taught during the course and in labs and perform the classification of the given text.

The main objective is to experiment at least with two features and perform analysis using techniques such as Subjectivity Lexicons, POS features, Cross validation and provide the precision, recall and F scores of the chosen text

As per the guidelines and recommendations I've chosen the Kaggle movie review dataset for my final project data. In this document will be providing the steps that are performed during the data processing, data analyzing and the experiments that are taken conducted along with the issues that are encountered.

To ensure that the recommendations provided by professor are followed here with drafting the report, I prefer to mention that I will be leveraging the code used during the lab sessions and the code ideas provided for the final project. Additionally, I prefer to experiment with Subjectivity lexicon to define a new feature and would wish to conduct on experiment using Weka classifiers.

The main goal of this final project (by the chosen dataset) is to predict the sentiments/opinion of reviews posted of users using machine learning (natural language processing) techniques (by coding certain algorithms) and compare the results by experimenting with different parameters.

Dataset Processing and Clean up

As mentioned the topic is movie reviews and the data is that was utilized for the "**Kaggle competition movie review phrase data, labeled for sentiment"** is used from the path https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews .By using the stop words filtering and tokenizing the movie data set up the review(s) text can be cleaned up

Note : Stemming and Lemmatization are not experimented here with experiment

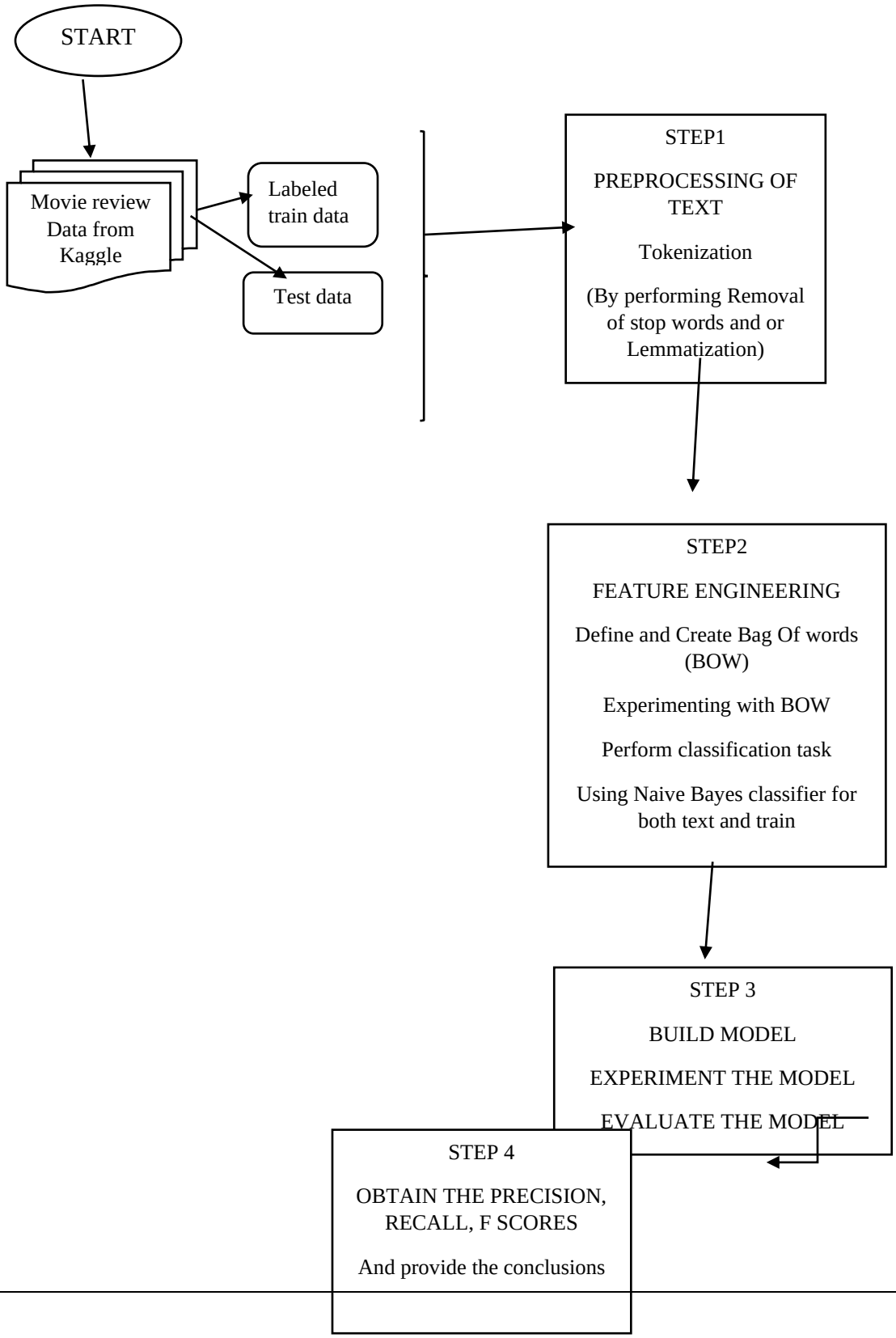
Approach

Based on the class learnings and certain study of NLP papers at this duration of my understanding "Bag of words" feature is something seems to be more effective and has a good impact to provide the results with better accuracy. Yet times this Bag of Words (BOW) is termed as Bag of Words model is a state-of-the-art method model to analyze the text.(most favored by the text mining professionals in the current trend)

The **Bag of Words** model is a method that takes as input text in the form of a sentence and turns it into a feature vector by considering the extracted vocabulary words and the frequency of their

occurrence. Named such because each word frequency count is like a "bag," with each occurrence of a word as an item in that bag, for the current movie review example it takes the review(s) of a movie text and convert it into a feature vector, which you need to classify its sentiment. This model concerns about whether given words occurred or not in the document.

Flow of the approach



The main theme of the project is to ensure all the natural language processing techniques are understood and applied appropriately.

For any type of text (document, tab separated, csv,,etc.) the very first step is to remove the words that doesn't add any value for analyses, like stop words (predefined and custom defined based on the type of document that is getting analyzed)

Perform the tokenize split the text into phrases or list of words (this needs to be list) which are readily usable to perform the grammar techniques and apply the parts of speech tagging.

There are few more smart techniques can be applied such as stemming and lemmatization. However, for the current project work the only techniques followed are tokenization and removing stop words

The more focus has been given to understand the concept of having bag of words here with the current assignment.

This has been defined using word features and the vocabulary keywords by specifying the same as (V stands as Vocabulary in the below code statement)

```
features['V_{}'.format(word)] = (word in document words)
```

Here the system can execute the lines of code provided if we define a feature such as positive, negative, highly positive, highly negative, neutral, objective, subjective. These are few terms that machine learning can understand the algorithms that are created to execute and provide the sentiment, polarity and or aspect, emotion and opinion of a text, novel, review (product, movie, service)

Having said the above, we defined a negation feature as the first of our feature.

Now, we shall create a function which are not in features, the purpose is that to identify whether the word exists in the review text or not and indicates True or False.

Note: Leveraging the lab code

```
# Creating the NOT features function

def NOT_features(document, word_features, negationwords):

    features = {}

    for word in word_features:

        features['V_{}'.format(word)] = False

        features['V_NOT{}'.format(word)] = False
```

Next is to define the POS (parts of speech) tagging this essentially helps to identify the nouns, verbs adverbs and adjectives, during our labs we did the exercises how the grammar can be utilized to the tokens and apply the tagging according to type of text that can be analyzed

For example

Noun followed by a noun

Noun followed by adverb or adjective

..etc.,

Since the current assignment is on the collected data of public opinion there will be vast different way of analyzing the statements / sentences to a novel or news article.

Main purpose of tagging here to instead of analyzing the raw text, we can analyze the tokenized (tokens) text to extract more information(emotion, opinion)

Defining Bigrams

For any type of corpus handling (engineering) one of the best techniques that are highly recommended are extracting unigrams, bigrams, trigrams, and quad grams and obtain their frequency distributions across the corpora.

In the current assignment we have the bigrams as the major language modelling technique, one can understand the meaning of bi gram as it suggests itself "probability of occurring a preceding word given word in connection with the previous word that would have been there in the sentence.

It has both negative and positive advantages for the current corpora.

For example : The movie was pretty bad as one of the user's comment

However, some one can mention "The DOP is pretty good in the movie"

In the above context we really can't guesstimate how the word "pretty" word can be understood by machines as this was preceded by a positive word "good" and a negative word "bad"

To avoid such confusions, one must rely on the Gold standard data which is mostly highly quality comparison of such things to get the accurate results when compared.

Not to deviate from the topic of bigrams and frequencies and scores, as taught in the labs

This feature uses the "lambda" function and measuring with one of the best statistical tool of Chi Square measure

```
def bag_of_words_biagram(wordlist,bigramcount):
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(wordlist,window_size=3)
finder.apply_ngram_filter(lambda w1, w2: len(w1) < 2)
finder.apply_freq_filter(3)
#(this function removes the data that are occurred only with a frequency < 3)
bigram_features = finder.nbest(bigram_measures.chi_sq, 3000)
#(for measuring informative features), using 3000 bigrams
```

```
return bigram_features[:bigramcount]
```

Note : I could run the code with “2000” bigrams only for “3000” and with “10000” words I got memory exceptions.

Cross Validation

Cross validation is one of the popular method that can be used to analyze the text of this type, here main aim is to predict the movie reviews that are given by large population.

The Cross validation will let us know the stability of the model (statistical analysis) on the trained data set and test data set. This helps the quality to evaluate the quality of our model chosen for machine learning.

There are multiple types of cross validation models are available (K-nn, multi fold) train and test with splitting the data either 50/ 50 or 80/20 and evaluate the results by computing the precision, accuracy, and F scores of the data with the features for both test and train data sets. Current one used here is “k fold”.

As I’ve leveraged the code from lab 8,9 and final project code ideas. Tried to attain the scores and accuracy with by having certain negation words more and the comparing the results with bigram sizes.

Test1

With the feature count 1000, 10 fold and 500 bigrams and 100 most common words and for 100 documents.

I’ve received the following scores for the POS feature set

For original set at 1000

```
Average Precision       Recall          F1      Per Label
0               0.029      0.033      0.031
1               0.142      0.254      0.174
2               0.860      0.579      0.687
3               0.129      0.273      0.170
4               0.075      0.095      0.080

Macro Average Precision Recall          F1      Over All Labels
               0.247      0.247      0.228

Label Counts {0: 54, 1: 169, 2: 506, 3: 207, 4: 64}
Micro Average Precision Recall          F1      Over All Labels
               0.492      0.400      0.419
```

For Bigram

```
Average Precision       Recall          F1      Per Label
0               0.029      0.033      0.031
1               0.142      0.254      0.174
2               0.860      0.579      0.687
3               0.129      0.273      0.170
4               0.075      0.095      0.080

Macro Average Precision Recall          F1      Over All Labels
                0.247      0.247      0.228

Label Counts {0: 54, 1: 169, 2: 506, 3: 207, 4: 64}
Micro Average Precision Recall          F1      Over All Labels
                0.492      0.400      0.419
```

For negation

```
Average Precision       Recall          F1      Per Label
0               0.045      0.067      0.053
1               0.137      0.254      0.171
2               0.864      0.581      0.690
3               0.136      0.274      0.176
4               0.075      0.090      0.077

Macro Average Precision Recall          F1      Over All Labels
                0.251      0.253      0.233

Label Counts {0: 54, 1: 169, 2: 506, 3: 207, 4: 64}
Micro Average Precision Recall          F1      Over All Labels
                0.496      0.403      0.422
```

For Positive

```
Average Precision       Recall          F1      Per Label
0               0.029      0.040      0.033
1               0.170      0.276      0.199
2               0.871      0.588      0.698
3               0.171      0.310      0.214
4               0.051      0.067      0.053

Macro Average Precision Recall          F1      Over All Labels
               0.258      0.256      0.240

Label Counts {0: 54, 1: 169, 2: 506, 3: 207, 4: 64}
Micro Average Precision Recall          F1      Over All Labels
               0.510      0.414      0.436
```

With the above process the third time run was giving the scores that are not related to with the rest of the evaluations

Which means the corpora with this technique of bigram having multiple times checking with occurrence of the wordings are better than the rest because when 4th and 5th attempt while checking the data either the data must be underwent with multiple conditions and could not get the stability.

Test 2

With the following

Updated the negation words with few more

negationwords = ['poor', 'dull','ugly', 'worst', 'waste','brutal', 'blood', 'bloody', 'murder', 'revenge','horror', 'sad', 'cry','unhappy', 'noise', 'sick','repeatedly', 'repeat', 'predictable', 'nonsense','underrated']

top ‘200’ bigrams

for “5000” phrases

for “5” fold

for “100” most common words

For the original set

```
  Average Precision       Recall          F1      Per Label
  0             0.232      0.165      0.192
  1             0.159      0.313      0.210
  2             0.846      0.610      0.709
  3             0.195      0.377      0.256
  4             0.104      0.221      0.140

  Macro Average Precision Recall          F1      Over All Labels
                0.307      0.337      0.301

  Label Counts {0: 228, 1: 827, 2: 2580, 3: 1084, 4: 281}
  Micro Average Precision Recall          F1      Over All Labels
                0.521      0.468      0.473
```

For Bigram

```
  Average Precision       Recall          F1      Per Label
  0             0.232      0.165      0.192
  1             0.159      0.312      0.210
  2             0.845      0.610      0.708
  3             0.195      0.377      0.256
  4             0.104      0.223      0.140

  Macro Average Precision Recall          F1      Over All Labels
                0.307      0.338      0.301

  Label Counts {0: 228, 1: 827, 2: 2580, 3: 1084, 4: 281}
  Micro Average Precision Recall          F1      Over All Labels
                0.521      0.468      0.472
```

For negation

```
   Average Precision       Recall          F1      Per Label
   0              0.550      0.135      0.216
   1              0.173      0.263      0.209
   2              0.597      0.696      0.642
   3              0.215      0.368      0.272
   4              0.369      0.150      0.213

   Macro Average Precision Recall          F1      Over All Labels
                  0.381      0.322      0.310

   Label Counts {0: 228, 1: 827, 2: 2580, 3: 1084, 4: 281}
   Micro Average Precision Recall          F1      Over All Labels
                  0.429      0.497      0.447
```

For positive

```
Average Precision       Recall          F1      Per Label
0              0.275      0.176      0.214
1              0.169      0.336      0.225
2              0.836      0.618      0.710
3              0.192      0.353      0.248
4              0.115      0.217      0.150

Macro Average Precision Recall          F1      Over All Labels
               0.318      0.340      0.309

Label Counts {0: 228, 1: 827, 2: 2580, 3: 1084, 4: 281}
Micro Average Precision Recall          F1      Over All Labels
               0.520      0.471      0.476
```

Understanding from Test 2 is that by observing the results

At the 3rd attempt (2-fold) with the positive features have the better score in comparison with original data set.

Note:

I've tried with "10000" records, it ran for nearly 1.5hours and gave me a memory exception.

Lessons Learnt

The overall techniques, idea behind the natural language processing of various forms of documents are understood with this course here with.

However currently these techniques are widely used for the consumer or customer reviews

This Final project topic is on movie reviews which is again can provide the positive and negative or and neutral comments provided by the viewers.

I was intended to extract the results like

1. No of positive words
    a. Highly repetitive words in the positive context
2. No of negative words
    a. Most used negative words
    b. High intensity words (this is to capture aspect and emotion of the viewers) how disappointed were they
3. Graphs with ranging words (features) for 1000, 3000, 5000, 10000 to provide better conclusions of the data and models used
4. Use the word to vector techniques and all

However, I could not conduct successfully any of the thoughtful techniques.

Also dusted with subjectivity lexicon ran into errors with my limited knowledge of python scripting.

Though the concepts and logic behind the processing of corpora are understood I was not there upto the level with coding to transform my intend.

However, with this tests I can conclude that this Kaggle movie review data sets are already cleaned up, processed and less chances of preprocessing

The accuracy received shows towards a positive mode of viewers.

The similar tests we perform at work for customer interest towards launching a new product line across top 10 and top 50 and top 100 flagship stores since we work a group of team members and the data provided us would go different filtration process it was effective to provide a stable and quality output.

Other Information

My future learning shall concentrate more on the python techniques with real world scenarios and ensure the logic thought through shall translate properly with the right features or functions.

My learnings should focus on

Sentiment Lexicons combined approach like Subjectivity and

Sci-Kit Learner Classifiers- with Random Forest and SVC other techniques

I’ve used this in one my R for data analytics for covid related project how preexisting conditions patients are impacted

However, lack of proper preparation from my end with python did not able to explore those techniques.

References

https://www.kaggle.com/c/word2vec-nlp-tutorial

http://keenformatics.blogspot.com/2015/07/sentiment-analysis-lexicons-and-datasets.html

https://nlp.stanford.edu/IR-book/
https://nlp.stanford.edu/sentiment/treebank.html?we=OR
https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184

https://realpython.com/python-nltk-sentiment-analysis/

https://www.nltk.org/book/ch07.html