NUS Internship Selection Test

Aniket Pradhan
Fake News Detection

August 5, 2019

Introduction

My task was to implement a model to detect/predict fake-news from the given data-set. For this purpose, I was given the LIAR_PLUS[1] data-set. The LIAR_PLUS data-set is used for fact-checking and fake news detection released in the original author's paper[2].

LIAR_PLUS

The LIAR_PLUS data-set contains information from the full-text reports of journalists in Politifact. The data-set is a tab separated (tsv) file, ordered in the following format:

- the ID of the statement ([ID].json).
- the label.
- the statement.
- the subject(s).
- the speaker.
- the speaker's job title.
- the state info.
- the party affiliation.
- the total credit history count, including the current statement.
    1. pants on fire counts.
    2. barely true counts.

[1] https://github.com/Tariq60/LIAR-PLUS
[2] http://aclweb.org/anthology/W18-5513

3. false counts.
4. half true counts.
5. mostly true counts.

- the context (venue / location of the speech or statement).
- the extracted justification

Although the data-set does not contain the number of "true" counts, some "true" counts are present in the data. For that, I have set the "true" counts as:

$$true\_counts = min(half\_true\_count, mostly\_true\_count) \tag{1}$$

Methods and Results

The original author had implemented a long short term memory (LSTM) neural network. Since, I don't have enough experience and knowledge in deep learning or neural networks and due to the time constraint, I stuck with the methods pertaining to simple machine learning problems.
The statements could be classified into six classes (pants-fire, barely true, false, half true, true) or two classes (true, false).
Instead of analyzing the statement or the justification, I made use of the credit history count of the data (number of times a statement has been annotated as true/false). I used a simple (Gaussian) naive-Bayes classifier for this purpose. First, I sorted the data, taking out relevant information and organizing them into proper classes. The next step was to train the model. The next step is the final step, that is to test the model by making predictions, and then calculate the efficiency of the same. I used a Multinomial and Bernoulli naive-Bayes classifier as well, and the results are shown in Table 1. I also used Logistic Regression for classification. The results for the same are shown in Table 1.

Type of Classification	Number of Classes	Efficiency
Gaussian Naive-Bayes	6	0.186
Gaussian Naive-Bayes	2	0.566
Bernoulli Naive-Bayes	6	0.353
Bernoulli Naive-Bayes	2	0.604
Multinomial Naive-Bayes	6	0.341
Multinomial Naive-Bayes	2	0.583
Logistic Regression	6	0.311
Logistic Regression	2	0.586

Table 1: Efficiency Comparison of Different Classification Models

Observations

From Table 1 it can be observed that Bernoulli Naive-Bayes model gives the highest accuracy as compared to all other models, and Gaussian Naive-Bayes model gives the least.
This can be explained by the fact that Bernoulli NB model assumes data to be distributed according to multivariate Bernoulli distributions, where there is at least one feature which is binary. In our data-set, there were multivariate classes, however features were not binary. According to me, Multinomial NB should give a better accuracy as compared to Bernoulli NB, since it just accounts for the multinomial features.
Logistic Regression Curve Fitting gives a better accuracy than Gaussian NB because the data-set is not Gaussian in itself, and using a Gaussian NB, hence, decreases the efficiency of the model. Logistic Regression is an efficient model in itself, and at times performs even better than Multinomial NB as well.

1 Other Work

I was trying to implement a LSTM in-order to process word-to-word embeddings. Due to time constraints, I was only able to implement the tf-idf model of the data-set. The same can be seen in the source code.

Libraries Used

- numpy
- scikit-learn
- progressbar2

How to run?

Please refer to README.md[3] present in the root of the folder. :D

[3]If you have a markdown parser, else refer to README.txt