Fake News Detector Docs

By Dikshant Sagar

Fetching Data

The Dataset was downloaded from https://github.com/Tariq60/LIAR-PLUS/tree/master/dataset and was in the form of three files namely train2.tsv, test2.tsv, val2.tsv. Out of these only train and test were used for the project.

Loading and Preparing Data

The Data was loaded using pandas module. The Data was in the form of 15 Columns namely-

- Column 1: the ID of the statement ([ID].json).
- Column 2: the label.
- Column 3: the statement.
- Column 4: the subject(s).
- Column 5: the speaker.
- Column 6: the speaker's job title.
- Column 7: the state info.
- Column 8: the party affiliation.
- Columns 9-13: the total credit history count, including the current statement.
    - 9: barely true counts.
    - 10: false counts.
    - 11: half-true counts.
    - 12: mostly true counts.
    - 13: pants on fire counts.
- Column 14: the context (venue / location of the speech or statement).
- Column 15: the extracted justification

For the purpose of training, the features utilized were namely -

- Statement
- Subject
- Speaker
- Speaker’s job title
- Context

These features seemed logically more contributing to the truthfulness of the text overall.

The Data contained 6 labels namely -

- Pants-fire
- False
- mostly-false,
- Half-true
- Mostly-true
- True.

For the Purpose of the 1st Part, these labels were combined into just true or false and vectorized as 0 and 1 and for the 2nd part, the original labels were vectorized to integers from 0 to 5.

Exploring Different Models

With Past experience and some research I shortlisted models that are known to be better performers in cases like this one :

- Naive Bayes
- SVMs
- Logistic Regression
- Neural Nets

Research Articles/Papers: -

- https://towardsdatascience.com/i-built-a-fake-news-detector-using-natural-language-processing-and-classification-models-da180338860e
- https://arxiv.org/pdf/1705.00648.pdf

For NLP we convert text into a number representation via vectorization and two methods came to mind :

- Count Vectorization
- Tfidf Vectorization

So I paired off both kinds of vectorizations with the models above for both cases and compared the statistics
And I noticed Multinomial Naive Bayes and SVM performing better compared to others with the following order -

- Count Vectorizer + Multinomial NB for Part 1
- Tfidf Vectorizer + SVM for Part 2

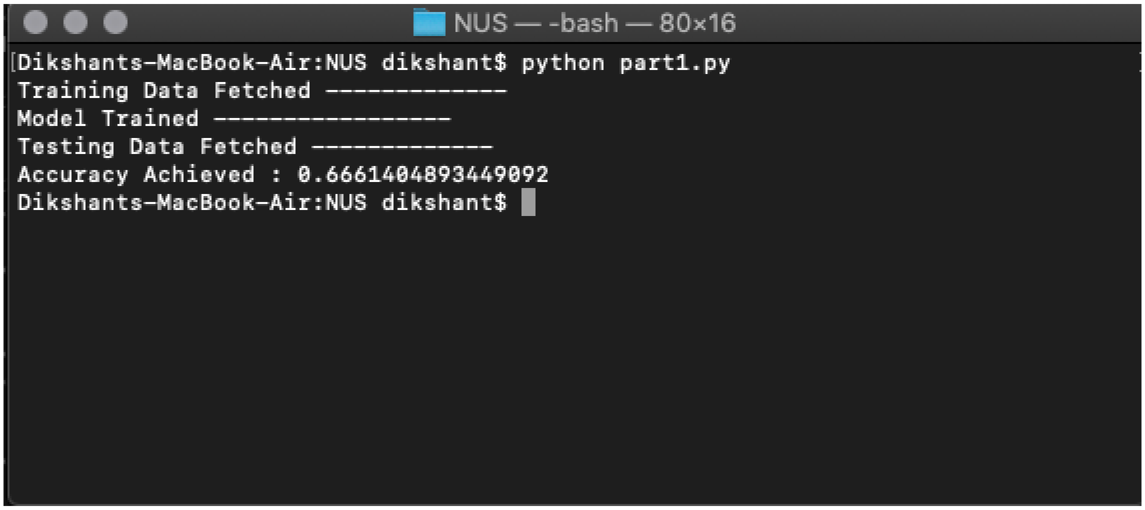
So I created a pipeline and a set up a grid search to maximize the accuracy by finding the best parameters for both

In the end,
For Part 1 Count Vectorizer + MultinomialNB gave the best result with an accuracy of **0.6661404893449092** on the test dataset with the following hyperparameters:

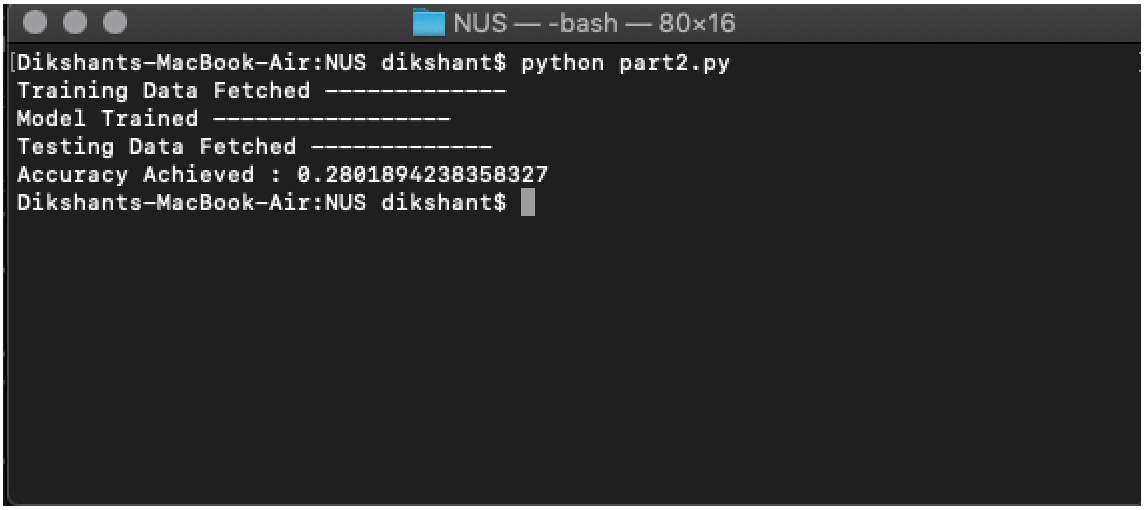
- CountVectorizer → max_features = 10000 , ngram_range = (1,3)
- MultinomialNB → alpha = 4

For Part 2 The Tfidf Vectorizer + SVM Model gave the best result with an accuracy of **0.2801894238358327** on the test dataset with the following hyperparameters:

- TfidfVectorizer → max_features = 10000 , ngram_range = (1,3)
- LinearSVC → C = 0.1



```
[Dikshants-MacBook-Air:NUS dikshant$ python part1.py
 Training Data Fetched -------------
 Model Trained -----------------
 Testing Data Fetched -------------
 Accuracy Achieved : 0.6661404893449092
 Dikshants-MacBook-Air:NUS dikshant$ 
```



```
[Dikshants-MacBook-Air:NUS dikshant$ python part2.py
 Training Data Fetched -------------
 Model Trained -----------------
 Testing Data Fetched -------------
 Accuracy Achieved : 0.2801894238358327
 Dikshants-MacBook-Air:NUS dikshant$ 
```

Running the Scripts

The file `part1.py` contains the code for part1 i.e for binary classification to just true or false and gives the accuracy on the test dataset as an output.

The File `part2.py` contains the code for part2 i.e six-way classification to the default classes in the original dataset and outputs the accuracy on the test dataset as an output.

To run these file :

- On the terminal go to the directory where these files reside i.e the NUS folder.
- And run `python part1.py` for part1 file and `python part2.py` for part2 file.

Dependencies and Libraries Required

For just the Scripts :

- Pandas
- Sklearn

For .ipynb notebooks (where code for all(more or less) trials reside) :

- Pandas
- Numpy
- Sklearn
- Keras
- Tensorflow
- nltk