CIS4526 Final Project Report

Course: CIS4526
Name: Abu Hasnat Hasib

Paraphrase Identification using Multi Layer Perceptron

General Overview

In this Project, I detect if a particular sentence is paraphrasing another sentence. This is completed in python3 using Jupyter notebook. It is a very simple example of Natural Language Processing. I used MLP (multi layer perceptron) to perform the taskAn MLP uses backpropagation as a supervised learning technique. Since there are multiple layers of neurons, MLP is a deep learning technique.

Features Designed

Difference in wordcount Length: Counts the numbers of words in both sentences and calculated the differences in the number of words.
Fuzzy ratio: Creates a similarity score based on common words.
Fuzzy token ratio: Creates a similarity score based on common words. It ignore rearrangement of words (compared to fuzzy ration).
Levenshtein distance: Finds teh lavenshtein distance between two sentences.
Bleu score: Creates a similarity score based on common words.
NIST score: Finds the NIST score between two sentences.

Data Preprocessing and Feature Preprocessing

For all training data, dev data and the test without label data I did the following:

- Removed unambigiouse values
- Removed all rows that had null values
- Changed all sentences to lowercase and removed commas and any other punctuation

Algorithms and Libraries

sklearn : I have used scikitlearn libraries for many tasks inluding implementing the mlp
pandas : I used pandas to convert the files to dataframe for better processing.
fuzzywuzzy : I used this to find similarities in sentences.
nltk : To find simlarity using bleu score and NIST score

Results

- After implementing Difference in wordcount Length, Fuzzy ratio, Fuzzy token ratio, and Levenshtein distance my results were as follows:

```
In [45]: clfmlp = MLPClassifier(hidden_layer_sizes=(6,5),
                                random_state=5,
                                verbose=False,
                                learning_rate_init=0.01)

clfmlp.fit(X_train, y_train)
y_test_pred = clfmlp.predict(X_test)
f1 = f1_score(y_test, y_test_pred)
acc = accuracy_score(y_test, y_test_pred)
print("F1 score: {:.3f}, Accuracy: {:.3f}".format(f1,acc))
```

```
F1 score: 0.801, Accuracy: 0.894
```

- After implementing the bleu score and NIST score in addition to the previous features my results are as follows. I also changed the hiddent layer sizes here:

```
In [65]: clfmlp = MLPClassifier(hidden_layer_sizes=(8,7),
                                random_state=5,
                                verbose=False,
                                learning_rate_init=0.01)

clfmlp.fit(X_train, y_train)
y_test_pred = clfmlp.predict(X_test)
f1 = f1_score(y_test, y_test_pred)
acc = accuracy_score(y_test, y_test_pred)
print("F1 score: {:.3f}, Accuracy: {:.3f}".format(f1,acc))
```

```
F1 score: 0.798, Accuracy: 0.893
```

Conclusion

This project was really interesting. Although I started the project by implementing MLP using Pytorch, I figured it was much easier and faster to implement the MLP Classifier included in scikit learn library.