AbuHasnatHasib_system_description

Course: CIS4526
Name: Abu Hasnat Hasib

Paraphrase identification

General Overview

In this Project, we 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. However, no deep learning model was used for developing this model.

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).
Bleu score: Creates a similarity score based on common words.

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
- Chnaged all sentences to lowercase and removed commas

Algorithms and libraries

sklearn : I have used scikitlearn libraries for many tasks.
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

The following images shows all the libraries I used for this project:

```
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from random import shuffle
```

```
from fuzzywuzzy import fuzz
import string
```

```
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import SmoothingFunction
```

For training, I dropped incompatiple features and then normalized the features I would use using Standard Scalar.

Results

SVM: For svm I have had the best accuracy of 61.2%.

SVM Model

In [473]:
```
clf = svm.SVC()
clf.fit(X_train, y_train)

# evaluate the model on the testing set
y_test_pred = clf.predict(X_test)


acc = accuracy_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred)


print("accuracy: {:.3f}, recall: {:.3f}, precision: {:.3f}, f1: {:.3f},".format(acc, recall, precision, f1))
```

```
accuracy: 0.612, recall: 0.952, precision: 0.568, f1: 0.711,
```

Logistic Regression: For Logistic Regression the accuracy was much lower at 51.6%.

Logistic Regression model

In [472]:
```
# retrain the model
clf = LogisticRegression(penalty='l2', solver='lbfgs')
clf.fit(X_train_val, y_train_val)

# evaluate the model on the testing set
y_test_pred = clf.predict(X_test)


acc = accuracy_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred)


print("accuracy: {:.3f}, recall: {:.3f}, precision: {:.3f}, f1: {:.3f},".format(acc, recall, precision, f1))
```

```
accuracy: 0.516, recall: 0.994, precision: 0.509, f1: 0.674,
```

Thus, for the final prediction I used svm.

Conclusion

This project was a great learning curve for me. Since, we could not use deep learning, I have learn a lot of basics on how to approach to build a model from ground up. I have never done a project like this, so it was really helpful to get help from the Internet and learn as I coded.