Siddarth Srinivasan

Professor Wenpeng Yin

CIS 4526

6 December 2022

Machine Learning Final

1. For my Multi-layer Perceptron machine learning model, I had maximized my accuracy using a total of four features: Here is the breakdown of what features I used: ['wordcount difference', 'sequence matcher', 'Overlapping ratio', 'synonym ratio']
    - ‘wordcount difference’

        Wordcount difference is the absolute value of the difference of the number of words between sentence 1 and sentence 2.
    - 'sequence matcher'

        Sequence matcher is a ratio that I had created by using the difflib package, which calculates the similarities of 2 phrases using gestalt pattern matching
    - ‘Overlapping ratio’

        To get the overlapping ratio, I got the total number of overlapping words between sentence 1 and sentence 2, and then divided this number by the number of words in the shorter sentence.
    - 'synonym ratio’

        To get the number of synonyms between the two sentences, I used the nltk package and imported wordnet to do this. With this, I created a function called getSynonyms and found the number of synonyms between sentence

1 and sentence 2. After getting this number, I divided it by the sentence with the lesser number of words. This gave me the 'synonym ratio'.

2. Data preprocessing and feature preprocessing
    - For data preprocessing, all I did was import the training, development, and test data and separated the data as specified with a '\t+', and then dropped any null values in the datasets. I also set the 'gold label' column in both the training set and the development set to an integer. Here is a screenshot on what I had done:

```
import os
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import string

columns = ['id', 'sentence 1', 'sentence 2', 'gold label']
training_data = '/kaggle/input/mlfinal/MLFinalProject-main/train_with_label.txt'
development_data = '/kaggle/input/mlfinal/MLFinalProject-main/dev_with_label.txt'
test_data = '/kaggle/input/mlfinal/MLFinalProject-main/test_without_label.txt'

df_test = pd.read_csv(test_data, sep = '\t+', names = ['id','sentence 1', 'sentence 2'])#

df_dev = pd.read_csv(development_data, sep = '\t+', names = columns)#.apply(lambda x: x.a

df_dev['gold label'] = pd.to_numeric(df_dev['gold label'], errors='coerce')
df_dev = df_dev.dropna().reset_index(drop = True)
df_dev['gold label'] = df_dev['gold label'].astype(int)

df = pd.read_csv(training_data, sep = '\t+', names = columns)#.apply(lambda x: x.astype(s

df['gold label'] = pd.to_numeric(df['gold label'], errors='coerce')
df = df.dropna()
df['gold label'] = df['gold label'].astype(int)

#Lists information of columns and respective data types
df.info()
df_dev.info()
```

3. Algorithms and Libraries used
    - Numpy, Pandas, String, fuzzywuzzy (fuzz.ratio), difflib (SequenceMatcher), nltk (nltk.corpus, wordnet), matplotlib (pyplot), sklearn (svm, make_pipeline, StandardScaler, LogisticRegression)
4. Experiences and Lessons learned
    - Doing this project was a very good learning experience and helped me strengthen my machine learning skills. I learned how to implement a proper machine learning model using a singular vector machine model and a multi-layer perceptron model. Doing research on what features I can add to the model helped me expand my skill set on what packages and libraries I can use. In addition, I learned a lot about the svm and mlp models specifically through tweaking arguments to maximize the efficiency and f1 score.
5. Results
    - Here is the result of the SVM model with the above features mentioned on the development data and the accuracy:

```
#development
import sklearn
from sklearn import svm
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

#features = ['wordcount difference', 'fuzz ratio', 'overlapping words', 'char difference', 'sequence matcher', 'wo
features = ['wordcount difference', 'sequence matcher', 'Overlapping ratio', 'synonym ratio']
X_train = df[features]
X_dev = df_dev[features]
y_train = df['gold label']
y_dev = df_dev['gold label']

classifier = make_pipeline(StandardScaler(), svm.SVC(kernel = 'rbf', gamma = 1, C = 1, class_weight = 'balanced'))
#classifier = make_pipeline(StandardScaler(), svm.SVC())
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_dev)
print(classifier.score(X_dev, y_dev))

X_test = df_test[features]
y_test_pred = classifier.predict(X_test)
```

```
0.879
```

        - The accuracy as shown above for the SVM model is at: 0.879
    - Here is an image of the Multi-layer Perceptron model classifier I created:

```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(hidden_layer_sizes=(6,5),
                    random_state=5,
                    verbose=True,
                    learning_rate_init=0.01,
                    max_iter = 200)

clf.fit(X_train, y_train)
```

      The model went through 54 iterations until it stopped, delivering an accuracy score of 0.897

```
#Calculate classifier score
ypred=clf.predict(X_dev)
print(clf.score(X_dev, y_dev))
```

```
0.897
```

    - To calculate the F1 score, I add the following code and got the result of 0.8991:

```
#Calculate f1 Score
f1 = sklearn.metrics.f1_score(y_dev, ypred, average='weighted')
print(f1)
```

```
0.8991117498537455
```

6. Precautionary measures
    - Concerned by potential overfitting, I found the accuracy for the training data and the development data and compared them. The accuracy for the training data was 0.90565 and the accuracy for the development data is 0.897:

```
# Test for Overfitting by using model on trai

# Evaluate the model on the training set
train_acc = clf.score(X_train, y_train)
print("train_acc = " + str(train_acc))

# Evaluate the model on the development set
dev_acc = clf.score(X_dev, y_dev)
print("dev_acc = " + str(dev_acc))

#Seeing how both numbers are very similar, thi
```

```
train_acc = 0.905653121394693
dev_acc = 0.897
```

- Comparing these accuracies, it demonstrates that there is very little to no overfitting

7. Output predictions
    - Here is the code for outputting the predictions into a .txt file

```
#Output results into a file
file = open('SiddarthSrinivasan_test_result.txt',"w")
df_test.reindex()
for i in range(df_test.shape[0]):
    print(df_test.loc[i,'id'],"\t",y_test_pred[i],file = file)
file.close()
```