CIS 4526 Paraphrase Detection by Multi-layer Perceptron

Will Schenk

Summary: Classification model to determine whether two sentences express the same meaning with a multi-layer perceptron.

Three datasets were provided:

train_with_label.txt dev_with_label.txt test_without_label.txt

Each contains two sentences. The training and dev sets contain ground truth values of 0 or 1.

- 0 For the sentences with different meanings
- 1 For the sentences with the same meaning

Data Preprocessing:

- Removed non-alphanumeric characters, lower-cased letters, and vectorized each word.
- Normalized all training data with a MinMaxScaler() since features do not follow a normal distribution.
- Upsampled the training set to increase the number of samples with ground truth of zero.
- Lemmatized each word using a dictionary provided by the NLTK library. This lemmatization converts a set of words to common synonyms.

Features:

-
- Bleu Score (BiLingual Evaluation Understudy Score): Similar to my subset feature, this subset-word-counter computes a score, with a greater number of subset lengths. Levenshtein Score: Measures the number of edits needed to transform sentence1 into sentence2.
- Common Synonym Count: For each word, find all of its synonyms, and then count the number of times a word in the second sentence appears in this list of synonyms.
- Difference In Length: Difference in the length of the sentences
- Words in Common: Number of words that appear in both sentences

Libraries Used:

- Pandas: Dataframes
- Numpy: Array and numerical operations
- Sklearn: Preprocessing, Grid Search for hyperparameter tuning, KFold cross-validation, metrics, resampling, and feature importance
- NLTK: Lemmatizing words, Bleu score algorithm
- Levenshtein: Levenshtein score algorithm

Results: My program uses a mutli-layer perceptron by Sklearn and obtained an F1-score of 0.82.

Experience and Lessons Learned: I originally created a bag of words vectorization and applied it to both sets of sentences. I then found the difference between these vectors in each sample and passed this as the feature vector. Unfortunately, this had a lower score than using my previous features. Still, to obtain an even higher score, I believe this BoW (Bag of Words) vectorization, formed differently, would be the correct path.