CIS 4526 Paraphrase Detection Project

Will Schenk

Summary: Classification model to determine whether two sentences express the same meaning (without neural networks).

<u>Three datasets were provided:</u>

train_with_label.txt dev_with_label.txt test_without_label.txt

<u>Each contains two sentences.</u> 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

The syntax for each sentence is very similar or nearly identical, making the problem difficult.

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:

- <u>Subset features:</u> Count the number of times each sequential subset of words of length "difference" appear in both sentences. This count is divided by the mean length of the two sentences.
    - Data Preprocessing: Dividing by the mean of that sample length prevents differing length samples from skewing the result.
- <u>Bleu Score</u> (BiLingual Evaluation Understudy Score): Similar to my subset feature, this subset-word-counter computes a score, with a greater number of subset lengths. <u>Levenshtein Score:</u> Measures the number of edits needed to transform sentence1 into sentence2.
- <u>Uncommon Words in Common:</u> Import a list of the 1,000 most common English words and counts the number of words in each sentence that do not appear in this list. Then find the magnitude of the intersection between sentence1 and sentene2.
    - Data Preprocessing: Divide by mean to prevent skew.

Libraries Used:

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

Results: My program uses a radial basis function kernel in SVM to classify sentences with an accuracy of 75%.

Experience and Lessons Learned: Learned how to engineer a machine learning task from start to finish with organization that allowed for model selection and fine-tuning of parameters. Over-preprocessing sentences can convert sentences that previously had different meanings into sentences with the same meaning. I learned that a fine balance exists in the processing of data.