This document details the way in which my solution to the final project was constructed.

After reading in the training set csv, I used the same function I used in my midterm to separate each of the columns to perform tokenization of the sentences. As in the midterm, I used a regular expression to remove any characters I felt were unnecessary and would negatively impact the performance of the model. At first, I also used the list of stop words included in the library nltk (natural language toolkit) and removed them from the sentences, but after the other preprocessing steps I used, I realized that the model would perform better without the removal of stopwords. I also used the python strip() and lower() functions to convert the sentences to all lowercase and remove any extra whitespace.

After tokenizing the sentences, I used the fuzzywuzzy library in python to compute the similarity scores between each sentence pair. The fuzzywuzzy library uses python-Levenshtein distance to compute this. I first used the simple ‘ratio’ function, but realized that my MLP system actually performs better using token\_set.ratio. Token set ratio takes out the tokens in common (which is why it was not advantageous to remove stop words), and it also tokenizes them more and removes any remaining punctuation, and then sorts the words alphabetically and joins them back together. This was very helpful in conducting any text preprocessing that my previous code did not accomplish.

Since there were far more negative samples than positive, I also used a random oversampler included in the imblearn (imbalanced learn library) in python. I made the proportion of positive and negative samples 0.45, which aided in the classifier not overpredicting negative values. In the dev set, which was used to predict the test set in the final product, this increased the total number of samples from 4000 to 4200.

A list of all of these similarity scores was then passed into the MLP classifier used in the sklearn library. I used the tanh activation function on two hidden layers of size 25 neurons and 15 neurons. The solver used was adam. The batch size was 300, so slightly larger than the default, and I also increased the number of iterations with no change required to terminate the classifier from 20 iterations to 100. I also made the learning rate faster than the default (from 0.001 to 0.008).

All of these previous steps were also conducted on the development set and the test set.

This system achieved a 0.7924 (or 79.24) f1-score on the test set.

Main Features:

- regular expression tokenization

- fuzzy wuzzy token set ratio

- imbalanced learn oversampler

- MLP classifier with 2 hidden layers