Midterm Project: Paraphrase Identification
Sean Britt

I used five features, BLEU scores 1 through 4 comparing the first sentence to the second sentence of each pair, and a length score. This decision came after trying many different ideas. I designed my own version of BLEU scoring, but after speaking to Professor Yin via email, he suggested that I try using the BLEU scores from a library. The library I chose was NLTK, mostly because it was also the library I used to tokenize the sentences. When comparing my BLEU score implementation to that of the NLTK library, I found the library versions had a much higher accuracy on the dev set. My length score is calculated as exp(abs(len(sentence 1) – len(sentence2)). I tried many iterations of a length function, including squaring the difference of the length of the two sentences and a ratio of the two sentences. I was originally going to try the inverse of the formula I described above, but after testing with this formulation, I found my best results, possibly because the larger the gap between two sentence lengths, the less likely the two sentences are a paraphrase, which this formulation greatly emphasizes.

Preprocessing consisted of:

1. Setting all words to lower case
2. Removing stop words from a list via NLTK
3. Stemming the tokens via NLTK
4. Removing any words of length 1

This is a tokenizing pipeline I saw repeatedly in literature. I noticed that this method was powerless against typos in the data. For example, in one pairing, “problem” was in sentence one and “prob lem” was in sentence two. This confuses the BLEU scoring later on, and is hard to fix. I looked into methods of accounting for this via an algorithm that finds the shortest path to convert one sentence into another via inserts, swaps, and deletions, but I believe it was beyond the scope of this project.

The model I used was sklearn’s SGDClassifier with a loss of “hinge”, which is equivalent to SVM. I got a top accuracy for scaled feature vectors of about 0.58 when using plain SVM, so I looked into the SGD version and topped out at 0.7 accuracy. Unlike the always-same accuracy I got using standard SVM, every run of the SGD SVM gave a different score, which ranged from 0.5 to 0.7. To take advantage of this, I fit the scaled training feature matrix to the model repeatedly until the accuracy is greater than 0.69. This brute force method provides good results on the dev set in 5-10 seconds. I tried repeating this until the model provided a score above 0.7 on the dev set, but after a long wait I decided 0.69 was the upper limit.

I learned about the options available in sklearn and NLTK, and the general pipeline of producing a machine learning model. Much of my time was spent looking for and reading material on the subject of feature extraction. One feature I saw repeatedly was the tf-idf. This produces a vectorized matrix of the words in a corpus, where each value in the matrix is a measure of the relevance of the word to multiple documents. I decided not to implement this because the documents in question are single sentences, but also because I never thought of a clever way to fit the td-idf matrix into a single feature value. What’s more, intuitively, BLEU scoring already holds some information on the relevance of a term or consecutive terms to the set of documents in question.