Joseph Shaffer

Shaffer.567

**Summary**

In this program, the similarity between all pairs of sentences is found, such as between sentence 1 and sentence 2, sentence 1 and sentence 3, …, sentence 1 and sentence 3000, sentence 2 and sentence 3, …, sentence 3 and sentence 3000, etc. Thus, the similarity between every sentence and every other sentence is found. The similarity between each pair of sentences is determined with 3 different methods.

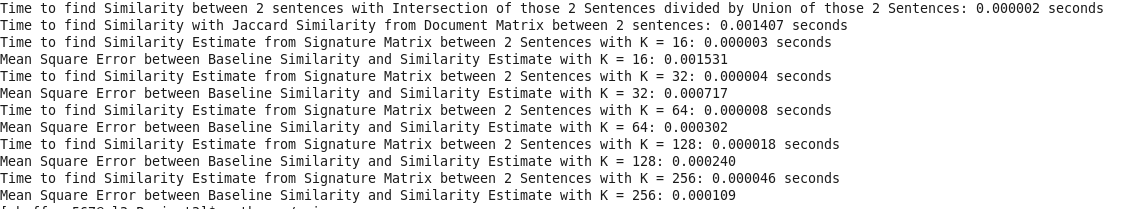
The first method for finding the similarities between every pair of sentences relies on storing each sentence as a set, and taking the intersection of a pair of sets (pair of sentences) and dividing it by the union of the same pair of sets (pair of sentences) which will give the similarity value for that pair of sentences. This is done for every pair of sentences to find the similarity value for every pair of sentences. However, it is important to note for this method that each sentence must be stored as a set to utilize the built in set functions to find the similarity between the two sentences, but gives you the exact similarity value and not an estimate. Thus, to find the similarity for all pairs of sentences, all sentences are stored as a set and the Jaccard similarity is calculated and stored for every pair of sentences.

The last two methods require the creation of document matrix, where every row represents an individual word that appears in at least one sentence, and the columns represent each sentence. The document matrix is filled with 1’s and 0’s by going through all individual words and then determining if that word appears in a sentence or not. If the word does appear in a sentence, a 1 is placed in the corresponding row and column of the document matrix that corresponds to that word and the sentence it appears in. However, if the word does not appear in the sentence, a 0 is placed in the corresponding row and column of the document matrix.

After creating the document matrix, we can find the exact similarity between all pairs of sentences, by iterating through every row of the document matrix for two columns that represent the pair of sentences we are comparing and keeping track of when both columns have a 1 in the same row and also keeping track of when at least one of the columns has a 1 in a row. Thus, using these two values that are found from the document matrix, we can calculate the similarity for the pair of sentences that we are comparing. This is an exact similarity, but requires iterating through every row of the document matrix for two columns at a time, and thus as every sentence is compared to every other sentence for every row in the document matrix (the number of individual words), means a very high number of comparisons from the document matrix and a lot of time to calculate the similarity for every pair of sentences.

The last method relies on k-minhash method, where a certain number of permutation matrices are used with the document matrix to create a signature matrix, that has the same number of rows as the number of permutation matrices, and the same number of columns as the number of sentences. The signature matrix is created by creating one row at a time. Each row is created by iterating through the permutation matrix by indexes and finding the first corresponding index in the document matrix to have a 1 in that row, which is also done for each column in the document matrix. Thus, after going through all columns of the document matrix with each individual permutation matrix, a row in the signature matrix is created with hashed values from the permutation matrix. This process is repeated for all permutation matrices to create a signature matrix that is used to find the similarity between each pair of sentences. To find the similarity for a pair of sentences, you look at the corresponding columns in the signature matrix and iterate through the rows to keep track of how many times both columns have the same value in the same row. Thus, the similarity values from this method are estimates and not exact as the number of rows are equivalent to the number of permutation matrices or the k value that is chosen in creating the signature matrix.

**Results**



From the results, we can see that the fastest method at 0.000002 seconds to find the similarity between a pair of sentences, due to the fact it relies on built in functions and the fact that all sentences are stored as a set. If the data was not preprocessed where each sentence was stored into a set, this method would not be viable and one of the other two methods described for finding the similarity between a pair of sentences.

The second method clearly takes the longest at 0.001407 seconds to find the similarity between a pair of sentences. This is due to the fact that this method calculates the exact similarity between a pair of sentences and requires not only comparing every sentence to every other sentence, but also iterating through the entire dataset. This means for the single comparison of two sentences, the program has to iterate through the a large number of rows that represent the number of individual words in the dataset. This occurs for every comparison of two sentences, and this dataset has just under 5 million comparisons. Thus, it is clear why this method takes so long for a single comparison, and will take an exceedingly long amount of time to find the similarity for every single pair of sentences.

The most reasonable method to use, which does not require a change in preprocessing, such as the first method, is the k-minhash method, that finds an estimate for the similarity for each pairs of sentences. It can be seen from the results above, that at K=16, a single comparison takes about 0.000003 seconds, which is almost as fast as the first method. Furthermore, it can be seen that the estimate has a very small error in comparing the estimated similarity for all pairs of sentences with the exact similarity for all pairs of sentences at 0.001531 or 1.53%. Also, it can be seen from the results that as the value of k is increased and the number of permutation matrices is changed in the creation of the signature matrix (which will have k rows) that the error found in the similarity estimate will decrease every time k is increased. This is because the signature matrix will now encompass more of the data and give a more accurate similarity for each comparison of a pair of sentences. This can be seen in the graph “Mean-Squared Error vs K” below. However, it can also be seen that as k is increased the time to find the similarity for a comparison of a pair of sentences will also increase. This is due to the fact that when finding the similarity for a pair of sentences, the method must iterate through all the rows of the signature matrix, and the number of rows is increased as k increases. Thus as k increases, there are more iterations for finding the similarity between a pair of sentences. This can be seen in graph “Time for Comparison vs K” below.

