Jacob Stovall

Aidan Spies

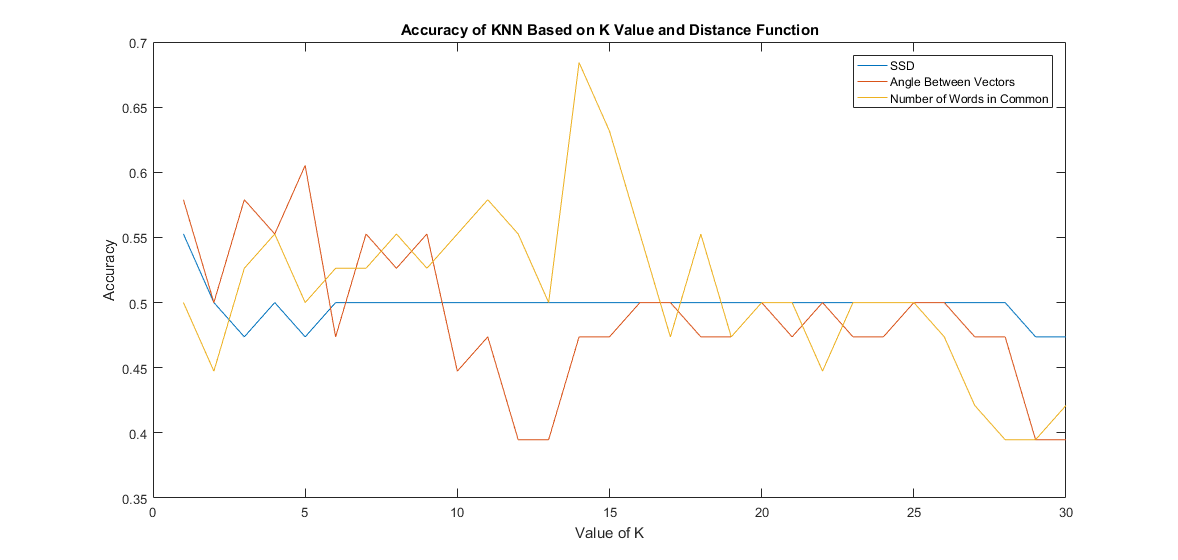
Christopher Mendez

**CSE 408 – Project 1**

**Part 1 - K-Nearest Neighbor**

We started project one by using the K-Nearest Neighbor algorithm (KNN) to try and predict whether a review of a product was positive or negative based on a set of training documents which were reviews labeled either positive or negative. While testing the KNN algorithm, three different distance algorithms were compared. These algorithms were:

* **Number of Words in Common –** This distance algorithm just checked whether two words were in the same document, and then summed the total number of words that both documents had in common. Then the all values in the vector were made negative so that when it was sorted, the documents with the most words in common would be at the front of the list. This is the most intuitive distance algorithm because it is easy to understand. However, the problem lies in the fact that a word only has to show up once for it to be a word in common with another document. So, the longer a document is, the more likely it is to have similar words to differently classified documents (and thus, the higher the chance a document can be misclassified).
* **Sum of Squared Distances –** This distance algorithm takes the difference of the number of times a certain word is used, squares the difference, and then adds all of the differences together (for all of the words) to get the total distance between two documents. This algorithm is useful because it groups together documents that use similar words the same amount. However, because it focuses so much on the number of uses of a word, it may misclassify documents that are written in a different style then the training data. For example, the training data might have a lot of people using the terms “good” or “great” in positive reviews. However, a positive test review might only use the term “stupendous” (which isn’t used in the training data). Thus, while the new term is a good indicator that the review is positive, since it wasn’t in the training data, the new test document may be misclassified.
* **Angle Between Vectors –** Angle between vectors takes two feature vectors and finds the angle between the vectors from the origin (by calculating their cosine similarity). This algorithm is good because it is not affected by the magnitude of each vector (rather it looks at the orientation of the vector). This is good because a document of a certain that has a lot of terms and that uses a certain term a lot can be found similar to a test document that may be a lot shorter and not use that certain term a lot. However, this could be a negative if the frequency of how many times a word is used is important in classifying said document. This algorithm also arguably did the best of all three empirically, but this will be explained in a second.



Above you will see a graph plotting the accuracy of the three distance algorithms based on a given K value. The best accuracy score (0.68421) came from the Number of Words in Common Algorithm when K = 14. However, the fact that the spike was so large and sudden puts into question how accurate it would still be if there was more testing data. In addition, with a K value of 14, that means it is polling a little over a third of the training data to classify the documents. Therefore, even though Words in Common ended up having the best score, the Angle Between Vectors when K = 5 (with an accuracy score of 0.60526) would probably be a more trustworthy combination to go with since it only polls a small portion of the training data and the previous few K values are close to the accuracy (meaning the algorithm is still pretty okay even when polling less of the training docs).

**Part 2 - Text Sentiment Analysis**

For part 2, we looked at classifying documents using sentiment analysis. Our code was very simple in that it read in a list of words with sentiment scores and created a map with key-value pairs, and then it took all of the words from a test document. Finally, we kept a score for the document starting at 0, and if the word in the document was a key in the map, we added (or subtracted if the value was negative) the value for the key from our score. At the end, a positive score meant the document was positive, and a negative score meant the document was negative. While the algorithm is pretty simple and can generally give a rough approximation of what the actual sentiment of the document is (when we tested, our accuracy was around .6944), it is far from perfect. One of the biggest issues with the algorithm is that the actual calculation is affected by how certain words are graded. So, what I may perceive as a strongly negative, the word map might have the word’s value at only slightly negative (or even possibly positive). In addition, the algorithm only takes one word at a time and doesn’t look at the context around it. For example, I might use the phrase “He’s so bad”, and the algorithm would probably classify it as a negative phrase. However, I may be using the phrase “He’s so bad” as slang for rebellious (like how Michael Jackson uses the term bad in the song *Bad*) which would change the meaning of the phrase into a positive one.

To give some examples from the data, “/Data/neg/04.txt” had a sentiment score of 7.9937. It was classified as positive because of words like “well”, “place”, “over”, “catching”, “very”, “visually”, “around”, and “actually” were classified as positive. While there were words like “polish”, “better”, “good”, and “pleasant” that also boosted the positive score, a lot of the words above probably shouldn’t be classified as positive because they can be used in documents with both types of sentiment (for example actually awful and very limited can both be negative phrases but are perceived as less negative because “actually” and “very” are seen as positive).

In addition, “/Data/pos/18.txt” had a sentiment score of -3.4585. It was mainly seen as negative because the review had the phrase “do not have” multiple times in it (all the words in that phrase have a negative score). It was also classified as negative because the review was comparing a good app to a bunch of worse apps, and pointed out the faults of other apps offering the same service (which caused the review to look negative to the algorithm.