Eric Lewantowicz
CSE 5243 Data Mining
Homework 3: Programming Assignment 2

For CL-1, I developed a K-Nearest Neighbor classifier from scratch using Python 2. For each test sentence, its words are compared against each of the training set sentences. If a training sentence contains a matching word from the test sentence, the similarity value for the training sentence is incremented in a simple accumulator. The similarity scores are then sorted, and the highest K similarity scores are chosen. The sentiments from the K most similar training sentences are used in majority voting to predict the sentiment of the test sentence. The prediction is compared to the actual test sentiment to calculate the accuracy of the classifier.

Additional features of this K-NN classifier: the 3,000 dataset sentences are divided into three sets—60% training, 20% validation, and 20% testing. The program first runs the classifier using the validation sentences for k = 1, 3, 5, 7... The k that yields the highest accuracy is used to classify the test set sentences. Also in the first project phase, three separate feature vectors were created: manually parsed, NLTK Stemmed, and NLTK Lemmatized. During the validation phase, for each k, the three vectors are run independently in CL-1 to create similarity scores and predictions, and the vector that produces the highest accuracy is used to run on the test set sentences. The idea in this approach is that Stemming and Lemmatization create word roots, reducing the overall number of unique words in the dataset and creating more opportunity for increased similarity between sentences. It took 8 seconds to parse the raw data and create all three dictionaries. (Tests were conducted remotely on stdlinux, and there was variability of execution time depending on stdlinux's workload.)

Finally, negative-word correction was used to improve accuracy. For example, if a training sentence matches a test sentence with the word "good", but the training sentence also contains the word "not", "isn't", etc., but the test sentence does not contain negative descriptors, then there is some chance that the sentiment of the training sentence should be opposite of the sentiment of the test sentence, even though they appear to have a high similarity due to other matching words. So the classifier also predicts the sentiment both with and without negative word correction. Whichever parameter combination of k, feature vector, and negative word correction produces the highest accuracy in the validation set is used on the test set and training sets. Optimal k was found to vary between program execution instances due to the randomization of datasets.

CL-1 on FV-1 on the validation set produced best accuracy 67.17% when k = 3 using the NLTK Stemmed vector. The time to classify a tuple before pruning ranges from 0.1 – 0.2 seconds depending on the value of k and the vector used. The optimal parameters were then fed into the test set. CL-1 on FV-1 on the test set produced best accuracy of 66.67% (400 correct/200 wrong), time to classify 0.10 seconds per tuple., and the K-NN classifier was executed again. Finally, the training set produced 67.72% accuracy in another program execution. The training set uses a leave-one-out approach, where the matching training sentence is ignored, and its similarity is not considered, so as not to positively bias the training accuracy. Time to classify was 0.02 seconds per tuple (significant variance during this execution due to server load).

The next step was feature reduction of FV-1 to create FV-2. First, any single-occurrence words were deleted from the frequency vectors since a word occurring one time throughout the dataset has no reference and is likely not useful. This step vector unique word counts by approximately 2,200 – 3,100 words from each of the three vectors, making the vectors smaller by over 50% compared to the original

unpruned vectors. Next, high-entropy words were deleted when the sentiment was evenly split between occurrences, i.e. the same number of occurrences in positive sentences as in negative sentences. Finally, using a measure similar to TF-IDF, words that had both relatively high entropy and frequent dataset occurrence were removed, using two threshold variables to adjust entropy and frequency limits, to prune common words that do not strongly classify a similar test sentence. After the last step, the vectors are 35% of the size of the original. Pruning time for all three word vectors took 81 seconds.

After pruning, CL-1 was run on FV-2 validation set determine the new optimal k, vector, and negative word correction parameters. Again, k = 3 gave the best accuracy, but it was found to vary throughout testing. The low optimal k is likely due to the sparseness of useful common word occurrences among sentences in dataset, such that there is minimal similarity among a test sentence and the training sentences beyond the top few similar sentences in the set. For larger datasets with more words per sentence, we may expect optimal k to be larger. The optimal was again the NLTK Stemmed, with best validation accuracy of 77.17%. This is an improvement of 10% over the performance on FV-1 validation set. Time to classify each tuple was also reduced, 0.026 seconds per tuple. Using optimal parameters, the test set was classified with accuracy of 74.33% (446 correct/154 wrong). Training accuracy during a typical run was 76.61%.

Overall time spent developing the K-NN classifier was 8+ hours, including developing the classifier, pruning the vectors, thresholds, vector handling, testing, and threshold optimization. Areas for improvement include optimization of pruning techniques and pruning threshold variables, and negative word correction. The final implementation of the negative word correction did not improve the performance, but it may improve predictor accuracy if implemented appropriately.

---

CL-2 is a binary Decision Tree using information gain, designed and implemented it from scratch. A BinaryTree class was implemented, with each inner node containing the split string value of a word from the dataset, and each leaf node containing a sentiment determined by majority voting of the sentiment of subset of sentences from the training set at that node. The dataset used the same division into 60% training, 20% validation, and 20% testing. The algorithm is as follows: using the training set, for the root node, the program iterates through each unique word occurring in the set at the node. The word that produces the best info gain based on sentiments for the sentences where the word occurs or does not occur is chosen as the binary split point for the node. All sentences in the training set are then assigned to the left subtree if they do not contain the word, and to the right subtree if they do contain the word, and the word is removed from the sets of unique words passed to the subtrees. The algorithm is recursively repeated on the left and right subtrees to create the tree. If all the sentences at a node are of the sentiment, the node becomes a leaf node and is assigned that sentiment. If the sentences are mixed sentiment, but the entropy (information needed) falls below a threshold parameter, the node also becomes a leaf node and the majority sentiment value is assigned to the node. This also helps to minimize overfitting the data and avoid creating an overly complex tree that may reduce accuracy.

Due to the relatively large size of the dataset, building a complete tree becomes time-intensive and memory-intensive. Therefore, the validation set is used to help identify the optimal depth of the decision tree and the optimal info-gain threshold parameter to stop subtree construction and assign a leaf node below a threshold information gain. Using the validation set, multiple trees are built with varying depths and info-gain threshold parameters, and the parameters that produce the best validation set prediction accuracy are then used in the classifier for the test and training sets.

For CL-2 on FV-1, the large size of the unpruned dataset made creating large decision trees time consuming. Therefore, the info gain threshold parameter was fixed to 0.20, and using the validation set, trees starting at 6 levels deep up to 21 levels were constructed at 5-level increments. The time to construct a 21-level tree with 155 nodes was 393 seconds, with a prediction accuracy of 66.17% on the validation set (compared to 59.0% on a 6-level tree with 33 nodes). The accuracy on FV-1 could have been increased with larger trees, but the time required to construct them would be unreasonably long. Using the 21-level tree with a 0.20 info-gain threshold parameter, the prediction accuracy achieved classifying the testing set was 68.83%, with a time to classify tuples of 0.0001 seconds per tuple. Compared to the CL-1 K-NN classifier, the majority of the time is spent constructing the decision tree, and once constructed, classifying tuples is very fast. Training accuracy was 74.22%. The relatively low accuracy of the training set is likely due to the small depth of the tree and small number of nodes.

The same feature reduction process was used as with CL-1. Due to the longer model construction time, only the manually parsed word vector was used for CL-2, reducing the vector from 5,383 to 1,806 words. The decreased size of FV-2 significantly reduced the time required to build larger decision trees, allowing for the construction of larger trees and more info-gain threshold parameters, resulting in greater accuracy on FV-2. The validation set was used to determine the optimal decision tree size, now between 26 and 61 levels, and for info-gain threshold parameter from 0.1 to 0.7 at increments of 0.2. Constructing a 26-level tree with 141 nodes took 132 seconds, and a 61-level tree with 354 nodes took 243 seconds. The best accuracy was generated by a tree with 61 levels, 354 nodes, and an info-gain threshold parameter of 0.1, producing an accuracy of 73.67% on the validation set. Using this optimal tree, the test set was classified, at a time of 0.0002 seconds per tuple, and a test accuracy of 76.33%. An additional test constructing trees up to 91 levels did not produce significantly better accuracy. Training accuracy was 81.00% on the optimized decision tree, Cl-2 on FV-2.

Time spent to build and test the Decision Tree classifier was 6+ hours, including test iterations under different threshold parameter conditions. Areas for improvement include continuing to construct larger trees with more depth, further adjusting the info-gain threshold parameter, and introducing different pruning techniques and/or weighting techniques to compensate for overfitting and noise within the training set and resulting tree.