The results of the improved POS tagger, as saved in table.txt, are the following. Note that the values for the Base column show the baseline performance, and the values for columns a1-a7 show the difference in performance between the base performance values and the performance values when that feature was dropped.

Difference in performance	Base	A1	A2	A3	A5	A6	A7
DummyPrecision	0.1296	0	0	0	0	0	0
DummyRecall	0.1296	0	0	0	0	0	0
DummyAccuracy	0.1296	0	0	0	0	0	0
DummyF1	0.1296	0	0	0	0	0	0
NaiveBayesPrecision	0.3668	-0.0052	-0.0034	0.0188	0.0102	0.0042	0.0016
NaiveBayesRecall	0.3668	-0.0052	-0.0034	0.0188	0.0102	0.0042	0.0016
NaiveBayesAccuracy	0.3668	-0.0052	-0.0034	0.0188	0.0102	0.0042	0.0016
NaiveBayesF1	0.3668	-0.0052	-0.0034	0.0188	0.0102	0.0042	0.0016
DecTreePrecision	0.6032	-0.0102	-0.0126	0.0170	0.0100	-0.0116	-0.0094
DecTreeRecall	0.6032	-0.0102	-0.0126	0.0170	0.0100	-0.0116	-0.0094
DecTreeAccuracy	0.6032	-0.0102	-0.0126	0.0170	0.0100	-0.0116	-0.0094
DecTreeF1	0.6032	-0.0102	-0.0126	0.0170	0.0100	-0.0116	-0.0094
RandForestPrecision	0.6716	-0.0042	-0.0054	0.0250	0.0176	-0.0100	-0.0054
RandomForestRecall	0.6716	-0.0042	-0.0054	0.0250	0.0176	-0.0100	-0.0054
RandForestAccuracy	0.6716	-0.0042	-0.0054	0.0250	0.0176	-0.0100	-0.0054
RandomForestF1	0.6716	-0.0042	-0.0054	0.0250	0.0176	-0.0100	-0.0054

1. We can see that the Random Forest classifier had the best base performance of all four classifiers across all four metrics of precision, recall, accuracy, and f1 (since all four values are the same). The base performance for Random Forest was 0.6716 for all metrics, putting it approximately 7% above the performance of Decision Tree, the second best classifier.

2. For the Dummy Classifier, none of the features had any impact on any of the four performance metrics. This is because the Dummy Classifier does not use any advanced classification method, but essentially reports the accuracy that would be obtained by guessing.

For the Naive Bayes classifier, dropping the a3 feature had the strongest impact on the performance. However, the performance actually decreased when this feature was dropped (as indicated by a *positive* value in the table). The feature that contributed the most to an *increase* in performance was the a1 feature, indicated by the smallest *negative* value in the table. The feature that led to the smallest amount of change in performance (either positive or negative) was a7.

For both the Decision Tree and Random Forest classifiers, the feature that contributed most to performance was the same as the Naive Bayes - a3, which lowered performance when dropped. The feature that had the largest positive performance contribution was a2 for Decision Tree and a6 for Random Forest. The feature that led to the smallest change in performance when dropped was a7 for Decision Tree and a1 for Random Forest.

3. It seems like this feature set was not very good for the Dummy and Naive Bayes classifiers, since their
performance measures were generally poor across the board. They consistently had performance measured well under 50 percent, even when features were dropped or text preprocessing was implemented. The feature set did seem to be better-suited to the Decision Tree and Random Forest classifiers, but I suspect that the increase in performance those two classifiers showed is more so due to them being overall better models and less due to the characteristics of the feature set.

4. If I had more time to work on this assignment, I think it might be interesting to try using different
tagsets (I believe Penn-Treebank was used for this text data) to see if a different, potentially larger number of POS tags would improve performance at all. I remember that in class we talked about how sometimes tagsets can have hundreds of different POS tags, but that more specific tags does not necessarily lead to an improvement in performance if the number of tags is cumbersomely large. I expect that a larger tagset would not improve the performance of the classifiers, but I think it would make an interesting point of comparison.

Something else I wish I could have implemented was better text preprocessing. For whatever reason, methods such as converting the data to all lowercase or removing certain punctuation actually had a negative affect on the classifiers' performances. I suspect this is because certain parts of speech such as proper nouns largely depend on these features to distinguish them from other parts of speech. I think implementing a better text preprocessing model could have been beneficial for the classifiers' performance, since for example you would still want the classifiers to recognize "The" and "the" (with different capitalizations) as the same word. However, implementing this in such a way that the it actually makes it easier for the classifiers to understand the data would be a more time-consuming challenge.