Essay Grader Project

Liam Edelman, Corey Habel

The essay grader project was a difficult yet rewarding experience. We started the project the way most groups would, with planning out a course of action for our scoring. We decided that we would weigh sentence length and incorrect spelling more than coherence since coherence is much more subjective category to measure. Despite the rubric advising us to use sentence number for length, our group decided to use word count. We believed that word count would be a better indicator for multiple reasons. For starters, as outlined in the rubric, sentences can be very ambiguous since a full stop does not necessarily indicate the end of a sentence. Another reason was that it is possible to have many sentences while also having an overall short essay, with word count we are able to be much more consistent with the scores given out. In order to achieve an accurate word count we decided to use the nltk library to tokenize the essay and from there we would count the tokens. However, this method would not prove to be as simple as that. A big problem that arose was that the nltk tokenizer would tokenize words based on punctuation, so the word "don't" would count as two words instead of one. We worked on this issue for a few hours and ended up coming up with the idea of removing all punctuation from the document. With this solution the word "don't" would be transformed into "dont" and therefore would count as one word. With the word count working we assigned some arbitrary values to map a score.

The next step in the process was to create a spellchecker. Yet again, we were faced with multiple ways of going about checking essay spelling. The rubric suggested using nltk's built in wordnet spellchecker. We downloaded the necessary packages and tested it out, only to find that wordnet’s dictionary was not nearly large enough. The small dictionary size led to a lot of false positives resulting in most essay’s having below average spelling. We proceeded to come up with a few ideas to counteract the abnormally high spelling errors. One idea would be to weight the spell score to account for the large amount of errors in most essays. While this seemed like a viable solution, we decided against it since this inherently wouldn’t fix the issue at its core. We instead opted for adding an additional larger dictionary to compare words. This way we could be absolutely sure that a misspelled word was misspelled and dock the essay accordingly. While this solution did provide immediate results, there were still a few issues with capitalized words being flagged as misspelled. However, it seemed like this would be a worthy tradeoff for more accurate spelling scores. We decided that in order to truly gauge spelling errors we would need to determine inaccuracies as a percent instead of just total words spelled wrong. We looked through our training data and came to the conclusion that most high scoring essays had around 3% of words misspelled.

For the final part of the grader, we decided to rely more on the number of fragmented sentences than verb and pronoun agreement for calculating coherence. To achieve this requirement we formed parse trees out of each sentence and then found all the nodes labeled FRAG. If a FRAG label was found we incremented a counter, then we decided overall coherence based on the spelling score subtracted from the number of FRAG’s. To account for the occasional anomaly of FRAG’s in good essays we allowed one free FRAG without penalty.

The last part of the project was creating a final score formula. We initially used the given formula in the rubric, but the final accuracy ended up being an unacceptable 33%. We then proceeded to tune the final formula over the course of multiple tests. After an hour or so we came up with a formula that scored a 98% overall accuracy on the test cases. While it may seem like our algorithm overfit the data, we didn’t use any machine learning algorithms so it ultimately didn’t matter.