CS 421: Essay Grader - Report

What worked: We generated parse trees, utilized pos tags, word lemmatizers and much more, and it was challenging to make it work. For parse trees we utilized the stanfordcorenlp which took some bit of time to figure out how to incorporate into the project.

- Sentence count: The number of sentences counted using sentence tokenizer. Some discrepancies identified in punctuation locations and sentence count adjusted accordingly.

- Spelling correction: Spelling mistakes identified and a count of spelling errors maintained. Pyenchant python package used to check tokenized words against dictionaries for spelling mistakes.

- Subject-Verb agreement: The subject and verb of a sentence must agree with one another in number whether they are singular or plural. If the subject of the sentence is singular, its verb must also be singular; and if the subject is plural, the verb must also be plural.

- Verb tense: Tense mistakes were identified by finding various tags which if found together do not work well.

- Sentence formation: Sentence formation mistakes are evaluated using the parse tree

- Essay coherence: Essay coherence is checked by looking at the individual sentenes and checking whether the sentence references are done correctly.

- Topic coherence: For topic coherence we have found out the synonyms of the essay topic and stored them in a dictionary. After this I check for all the common noun occurrences in the essay. I used Wordnet to find the important words in the topic which are used later to find similar words in essay. Now using the synonyms of the words in the topics I find the match in the essay.

What did not work: We realised that a few techniques could not be implemented due to time or memory consumption boundaries. Some functions that included heavy use of Spacy library using some pipelining techniques were very time consuming and we had to find alternative solutions sticking to nltk or CoreNLP. While trying to find ‘SBAR’ discrepancies, we tried to use the Stanford CoreNLP module directly by calling the parser functions having ‘path to jar, path to module’ as arguments.

Even though the method was correct, it consumed 70-80 minutes for generating parse trees which is why we had to find a effective way to utilize corenlp. We had to use the techniques which involved calling the java server for the same which surprisingly took only 10 minutes for the same procedure. Generalisation of the POS bigram pairs for finding grammatical mistakes was very tricky. Even though a pair seemed very likely to work for every POS pair in the sentence, it would generate some erroneous pairs which weren’t supposed to be tagged as grammatical mistakes.

Learnings from the assignment: From the assignment we learned how to make use of various methodologies in python. We learned about how to make use of various modules as well how to make them run in an effective manner. Utilizing parse tree to find out the mistakes in the essay was very challenging and making it efficient was important as well.

We were able to get a hands-on experience with the techniques studied in class. We were also able to work on some external modules like Spacy, neuralcoref, integrating CoreNLP with python. Incidentally, some of the techniques proved useful for one of us in a project for a different subject dealing with Data Mining which increased the efficiency of that project as well.

For the essay grader to work better: In order to get better performance from the essay grader we would have to make use of probabilistic model as well as a word similarity metrics. A probabilistic model would help to figure out the sentence formation and it would be helpful to get better accuracy. Also for the purpose of topic coherence currently we have just utilized the common nouns as it was mentioned but we could utilized verbs and other useful criteria as well.

Use of deep leaning modules and an extensive training data would be useful to weed out some mistakes based on a more efficient pattern recognition. This would optimize the project further for data format which need not be structured like an essay, but also for some informal text structures.