Final Report

Project 1 for CS421- Natural Language Processing

Shobhit Lamba - slamba4@uic.edu Mohit Adwani - madwan2@uic.edu

---

For Essay Autograder, we had to dive into various techniques learnt in class and implement most, if not all, of them to grade different aspects of the an essay. Here are the different techniques used for each of the four tasks:

1. **Counting number of sentences**
   We took a multi-level approach to count the length of an essay:
   - First, we get the sentences by applying nltk sentence tokenizer on the essay. Then we split all sentences which have a newline character (\n).
   - We saw a pattern in the essay in which there wasn't a space after period (which denotes the end of a sentence) and hence the sentence tokenizer didn't split them into 2 sentences. Hence, we split the sentences into two if the character after a period is alpha and before the period isn't period(.) since some essays had two or more consecutive periods to denote continuation. Also, the sentence after the period should at least have 3 characters.
   - While looking at multiple finite verbs in the sentence(hint given in project_part1.pdf), if the sentence didn't have coordinate or subordinate clause then we saw a pattern in the parse tree, denoting the finite verb phrase as '(SBAR (S' – where SBAR denotes 'clause introduced by a (possibly empty) subordinating conjunction'. If the sentence has a subordinating conjunction then it is denoted by '(SBAR (IN that) (S'.

2. **Spelling mistakes**
   Again, we took a multilevel approach for counting number of spelling mistakes:
   - First, we simply tried to find the spelling mistakes using wordnet. But wordnet falsely reported simple words like "how", "you" etc as wrong spellings. So we used it as our first pass of spellcheck and its output was used as an input for the next step.
   - After that, I employed Peter Norvig's spell correction code to further filter out the words. Now to a great accuracy, we can say that the words left in the list are probably the only words that are incorrect in the essay.
3. **Grammar**

Grammar is the hardest task we had to tackle in part 1 of the project. Here, we had to handle the following:

- ➔ Subject-Verb Agreement
- ➔ Missing Verbs
- ➔ Incorrect Verbs
- ➔ Verb Tense

For this, we had to employ POS tagging. After every word of the essay was tagged, the task was subdivided as c.i and c.ii

For c.i (Subject-Verb Agreement), we defined certain rules that a subject and verb combination shouldn't follow, which were basically the singular and plural agreements. If the POS-tag sequences were found in the list of rules, error count is incremented.

For c.ii (Rest of the conditions), we again defined rules, but this time the list included rules that are correct. So, if the sequence does not match any of the defined rules, error count is incremented.

For c.iii (Sentence formation), we postulated the correct form of the parse tree and converted those rules in if else blocks.

1. (ROOT (S/(SINV is the correct form, if not, then increment the errors.
2. (ROOT (SINV (VP is found, then increment the errors.
3. If FRAG is found, then increment the error
4. If (NP/(VP is not found before (SBAR (S then increment the error

4. Essay Coherence (Topic coherence, in particular)

For D.i, pronoun coherence, we extract all the pronoun tags (PRP and PRP$) after pos tagging of the sentence. Then we look at the current sentence (till the word) and previous two sentences to search for the co reference. We extract all the common nouns and proper nouns. We search the common nouns in the wordnet and get its lexname. noun.group, noun.person, noun.body are taken into consideration and matched with the pronoun cardinality. If it doesn't match, then increase the number of errors. If the POS tag returns a Proper noun, we check the Named Entity of the word using Stanford core NLP Named Entity Recognition. If for singular NER returns anything other than Person we increase the number of errors. Similarly if for plural NER returns anything other than Group we increase error count.

For D.ii, topic coherence, we extracted the unique topics provided in the csv file and saved an identifying marker in the form of a unique word present in each sentence, in the form of a list. Now, everytime the function is called, it checks if the topic has any word corresponding to the unique marker in the list. If such a word exists, a list is called up which contains most probable words, synonyms and antonyms that can occur in a coherent essay and every occurrence of such word is counted. At the end of the run, the function returns a count of occurrences of these words in the essay. It is assumed that higher the word count, more coherent the essay will be.

After the list of scores were generated for a, b, c.i and c.ii, we normalized it using the following equation-

$x = x - \min(x) \;/\; \max(x) - \min(x)$

This gave us values normalized to a range of 0-1.
To get the spelling errors in range of 0-4, we multiplied every value with 4.
All the other scores were multiplied by 4 and subtracted from 5 to get the final scores in the range of 1-5. This ensured that lower the number of errors, higher the score obtained by the essay.

All values finally were rounded off to nearest integer value.

Final score was calculated for each essay using the given equation which is:

Final score = 2 * a - b + c.i + c.ii + 2 * c.iii + 2* d.i + 3 * d.ii

The results are finally written into a text file.

- ★ **Problems faced**
  Every part of the project had some issues that we faced.

  While calculating length of an essay, we found that our function wrongly separated sentences when a period was encountered with an abbreviation (like in “e.x.”). But since the number of such cases were minimal, we decided to ignore it. We also found that using capitalization as a characteristic for new line did not help. So we excluded that.

  For part b, we found that wordnet had a very weird way of finding out existence of word in the corpus. It uses a function called “synset” which finds synonyms of words in the corpus. Since many words do not have synonyms, they were falsely marked as wrong spellings. That is why we implemented a second layer of scrutiny which employed probabilistic methods to find out if a word is misspelled. This really boosted our performance in part b.

  For part c, the main problem was generalizing the code. Right now it is not that general, since english grammar has a lot of rules to define grammatical errors pertaining to verbs.. So, we had to separately jot down rules for both c.i and c.ii. Additionally, c.i does not have rules pertaining to adverbs so it might

still have a certain degree of inaccuracy on the higher end. Since the nltk POS tagger sometimes doesn't tag the verbs correctly, some verb formations, even though correct, are tagged as incorrect.

★ **Variations in Final Equation**

Final score:

2 * a - b + c.i + c.ii + 2 * c.iii + 2 * d.i + 3*d.ii

Intuitively, we observed the following things:

1. Number of sentences should not get such a high weight in final score. An essay with 50 sentences might not be better than an essay with 25 sentences. It may so happen that someone just writes one sentence, "Pollution is bad." over and over 50 times while someone else wrote an essay that was coherent and flowing but only had 25 sentences. Our autograder might wrongly grade the first essay higher with higher weight given to the number of sentences.
2. Spelling mistakes should have more weight proportionately. More number of spellings mean the writer is not well versed in writing in english and consequently writes poorly.
3. While grammar holds huge importance, our model for subject verb agreement only detected a maximum of 2 errors while most of the scores were zero. Scaling them to a scale of 1-5, zero errors got a 5, 1 error got a 3 while 2 got 1. But 2 errors is not that bad compared to 1 or zero, yet score difference is an anomaly. What if someone wrote an essay with an error in each line. He scores a 1 too, but in contrast, his essay was way more poorly written than the one with 2 errors.
4. Our scores for c.ii had a wide array of results and we believe c.ii should have more weight in the final score as well.

Running a linear regression analysis on the coefficients, we found that the best results are obtained for the following equation:
0.44412049 * part_a - 0.18825032 * part_b + 0.12061126 * part_c_i - 0.08335527 * part_c_ii + 0.05898674 * part_c_iii + 0.03865994 * part_d_i + 0.05246802 * part_d_ii

---