Project 1 for CS421 - University of Illinois at Chicago

Jordan Williams jwill49@uic.edu and Jason Deutsch jdeuts4@uic.edu

April 27, 2015

SETUP

For our project we are using Python 2.7, NLTK and Pyenchant. We are also using the Stanford Parser which requires JRE 1.8.

Please update src/stanfordParser.py with the correct java path.

Additionally, please add stanford-parser-3.5.2-models.jar to lib/

The input requires tokenized text.

Run make to install the necessary packages and to test the data. NOTE: The NLTK POS tagger requires a couple of corpora to be installed as well. You will get an error if they are not.

TECHNIQUE

For spelling, we counted the number of words that did not appear in the pyenchant dictionary divided by the wordcount.

For subject-verb agreement, we made a list of violating rules and counted the occurrences of such rules in the essay divided by the number of tags.

For verb tense agreement, we used a similar process for subject-verb agreement.

For length, we noticed a strong correlation between the number of unique verbs and the overall essay score so added that as a factor for the essay length along with the number of unique POS tags and the number of sentences.

For complete rule definitions see: - Spelling: —> spelling.py —> pyenchant - Subject-Verb: —> subjectVerbAgreement.py - Verb Tense: —> verb.py - Topic —> topicCoherence.py - Length: —> sentence.py and verb.py

Score Calculation

```
    We have learned the cutoff points for each category based on our training data. The model is loaded

    - Spelling:     ---> [0.09375, 0.05649717514124294, 0.04184100418410042, 0.029850746268656716]
    - Subject-Verb: ---> [0.009852216748768473, 0.007874015748031496, 0.005141388174807198, 0.002832861
    - Verb Tense:   ---> [0.18181818181818182, 0.125, 0.1, 0.058823529411764705]
- Topic         ---> [0.006493506493506494, 0.01141552511415525, 0.014577259475218658, 0.019480519480519
    - Length:       ---> [52.2, 64.9, 71.8, 81.8]

    - Overall Grade ---> [16.333333333333332, 24.333333333333332]
```

TODO

For the second part we will exploit the parsers to find errors in the sentences. We can see for instance how many fragments were recorded in the parser divided by the total number of sentences. We will also analyze parsed sentences that have SBAR in them and check whether the sentence is correct or not, possibly by using some rules defined with the POS tagger.

We will evaluate the strenghts and weakness of different parsers (py stat parser, NLTK, pattern.en etc.) We will also look at a POS tagger using conditional random fields.

For text coherence we will some sort of refrence expression (refrence resolution). We may also use a chunker for additional aid to differentiate between coherent and non-coherenet sentences.

To see if the essay addresses the topic, we can look at how many expressions include words that are related to cars. Examples could be, fuel, gasoline, pollution, parking, cars, etc.