CS 421 Term Project Part I

George Dill, Hassan Murtaza,

University of Illinois at Chicago

April 18, 2018

# Summary

The project is to demonstrate understanding of calculation techniques in Natural Language Processing by developing an essay grader similar to one used on the TOEFL exam. The grader will rank a writer’s English language proficiency from High to Low based on the following criteria [ and ranking ]:

* Spelling [ 4 – 0 ]
* Sentence Count [ 1 - 5 ]
* Subject – Verb Agreement [ 1 – 5 ]
* Verb Usage Error [ 1 – 5 ]

This document provides instructions for installing and running the essay grader and explanation of the algorithms used and scoring methodology.

# Installation and Execution

The program is written in Python 3.6.5 on MacOS High Sierra and relies on the NLTK package, NLTK words corpus, and Stanford Core NLP Parser.

1. Startup Stanford Core NLP Parser Local Server
   1. [Download](https://stanfordnlp.github.io/CoreNLP/) Stanford Core NLP and extract the .zip file
   2. From terminal, change directory to the extracted file
   3. Startup the server with the command:

java -mx4g -cp “\*” edu.stanford.nlp.pipeline.StanfordCoreNLPServer -port 9000 -timeout – 15000

1. Startup AutoGrader Package
   1. Download autograder package from blackboard and extract
   2. Place essays to be graded in: ./autoGrader/input/testing/essays/
      1. Place the index file for testing essays in ./autoGrader/input/testing/
   3. Run the essay by executing ./autoGrader/executable/run.sh
2. Grading Essays:
   1. The grader will prompt for a filename. Enter the filename to grade the essay.
   2. Scores will be logged to ./autoGrader/output/results.txt

# Approach

## Spelling

Spell check is accomplished by comparing words from the essay to nltk words corpus. The words corpus is the spell check dictionary used by open office project.

Algorithm:

Split the essay into word tokens

For each token

Strip tailing punctuation from the token

Remove internal punctuation and split into separate tokens

Test if token appears in the dictionary

If the token appears in the dictionary: Word is spelled correctly

Else If the token does not appear in the dictionary:

For each inflective ending

Modify word to remove ending and test

If the word is in the dictionary: Word is spelled correctly

End For

Word is spelled incorrectly

End If

Return Count of Misspelled Words / Total Words

Scoring:

## Sentence Count

Using the Stanford Core NLP Tools, the parse tree is found and the number of trees marked “S” is counted, approximating the count of complete clauses in a sentence.

Scoring:

## Subject Verb Agreement / Verb Usage Errors

The algorithm utilizes feature structures to compare properties of parse tree constituents and count violations of common grammar rules.

A parse tree is returned from Stanford Core NLP. Our algorithm recursively traverses the parse tree until a leaf node is found. At the leaf level a feature structure is constructed with attributes based on the word’s POS tag.