**NAME:** Yashvardhan Udia **NETID**: yudia2 **UIN**: 656090513

**Project Files and Functions:**

1. **run\_project.py:** This is the main entry point of the project. In this file, all other functional results are aggregated. It contains function:

* **main:** That reads the input\_essay and prompt from a text file in the same directory, pre-processes the corpora to find mean lengths/errors/similarity score and calls get\_grades.
* **get\_grades:** It takes the input\_essay, prompt and the corpora averages, calculates the scores for the input essay using score = mean error(implicitly takes c1+c2+2\*c3) + 2\*(avg of high- and low-grade essay lengths) + 3\* cosine sim score and returns a grade comparing the corpora averages with essay scores. If essay score lesser-equals to mean scores or length is less than 11 or the essay is the same as the prompt then it’s a low essay otherwise high. Here all scores are scaled only when the final score is being calculated.
* **show\_processing\_animation:** Fun little function to display a loading animation while the essay is being processed. Threads are used to parallelly run this task.

1. **essay\_pre\_processing.py:** This file pre-processes the essays corpora to find average scores of number of sentences and c1/c2 type errors. The functions:

* **essay\_pre\_processing:** It takes the paths of the index.csv and essays folder and returns the averages, max/min sentences of high & low grade essays, and max/min number of error in the corpora. Classifiers are called from here.
* **Get\_essays:** this function fetches the essays in a dictionary with key as file\_number and value as the essay.
* **Get\_essay\_details:** this function reads the index.csv into a dictionary of dictionary with key as file\_number and values as dictionaries labeled as (prompt, grade, number of sentences, total number of errors)
* **Pre\_process:** this function reads all the essays one by one and stores its sentence count and errors in the dictionary essay\_details. It segregates number of sentences based on the grade of the essay(high/low). It calculates the parameters used for scaling the scores and grading, namely, max\_length, max\_error, min\_length, min\_error, high\_mean\_length, low\_mean\_length, mean\_error
* **pre\_process\_prompts:** Computes prompts and essay similarity for the corpora to have thresholds for mapping individual essays. Gives the csim high and low scores on 1-5 scale.

1. **Scoring.py:** Used for scaling. Maps thescore of number of sentences/number of errors from a random min to max range of corpora to a score of 1 to 5. Formula for:

* **Number of sentence:**  scaled value = 1 + ( (raw\_value-min\_length)/(max\_length-min\_length) ) \*4, rounded to 2 decimal places
* **Number of errors:** scaled value = 5 - ( (raw\_value-min\_length)/(max\_length-min\_length) ) \*4, rounded to 2 digits

1. **a\_length:** Checks each sentence for complex/multiple sentence and assigns score for well-constructed sentences:

* If multiple finite verbs (VERB or AUX POS) are present check for either (Mid-sentence capitalization that’s not a Noun) or (a conjunction that’s joining 2 finite verbs) then uses formula min(F. verbs -1, #Conjungations + #mid-sentence capitals found)

1. **C\_syntax\_grammar:** Checks grammatical errors using below functions:

* **Sv\_agreement errors:** If a subject is found, check if its plural [NNS,NNPS] or [‘we’, ‘they’, ‘you’]. Or else check plurality if ‘And’ is encountered after a subject indicating compound subject. Another check is of Verb/Aux and its subject’s plurality, as they should agree on plurality.
* **Verb\_form\_errors:** Auxiliary verbs/Modals should be in agreement with VB. Checks for multiple AUX-VB pairs. Checks if have/we is not followed by a VBN.
* **Expl\_check (UNUSED for now):** If an expletive (there etc) is used it should agree in plurality with its following [Noun, Pronoun]. Checks for plurality agreement of expletive and the verb it points to.
* **Evaluate\_syntax\_errors:** Goes through the essay sentence wise and collects all the C1 and C2 type errors sing these functions.

1. **D\_i:** Checks grammatical errors using below functions:

* **get\_semantic\_analyses:** This function finds the embedding of the essay and its prompt using spacys vector. Then it calculates the cosine similarity between this essay and its prompt based on its grade and then calculates cosine sim between this high/low essay and a prompt from the corpora that has the least similarity to the prompt. Then averages of the high/low essay values are taken and scaled to find an average high/low cosine sim for the corpora.
* **Get\_essay\_emb:** Takes the essay/prompt then finds the average of the word embedding of the sentences and then averages for all the sentences in the essay.

1. **C\_3:** Checks grammatical errors using below functions:

* **Evaluate\_syn\_well\_form:** First checks for question sentences if they have Wh-terms or start with Aux. Else checks sentence with verb/aux in the beginning that is not aux/supporting another verb. Then checks for singular nouns that have unnecessary determiners with them and considers exceptions. Then Checks for Subordinate conjunctions for their respective clauses. This analysis uses dependency parse trees and spacy tokenizer.

1. **classifiers:** Sklearn and numpy are used here to train *multilayer perceptron* and *logistic regression* models on the corpora. High/low essays are separated. Test and training sets are developed by using these separated essays in a 20:80 split and the 80% training set has proportional high/low essays. Binary mapping in high for 1 and low for 0 is done. F1 scores are then calculated and accuracy is reported.

**Sample OUTPUT:** The output will display various raw scores (not 1-5) and then the evaluated Grade for the input. Then the F1 accuracies of the models are displayed.

**Raw scores:**

**Number of sentences: 17,**

**C1 errors (SV agreement errors): 0,**

**C2 errors (Verb form errors): 3**

**C3 errors(Syntax not well formed errors):15**

**Semantic similarity with prompt:0.648819088935852**

**Grade: HIGH**

**F1 Accuracy using different classifiers:**

**Multilayer Perceptron accuracy:0.8571428571428571**

**Logistic Regression accuracy:0.9090909090909091**

**Packages Used:**

* threading
* itertools
* sys
* time
* spacy ("en\_core\_web\_sm")
* os
* numpy
* csv