Project Report

Teammate 1 NAME: Mehrab Mustafy Rahman

netid: mrahm

Teammate 2 NAME: Mohammad Anas Jawad

netid: mjawad4

<u>Addressing the Scoring Components:</u>

a) Number of Sentences:

To count the number of sentences in a paragraph, the first task we performed was to split the paragraph into sentences using periods at the end. But some of the sentences may not have ended with periods. To deal with those we identified the POS tags of the individual sentences obtained earlier. We then checked the number of words starting with an uppercase but not being any proper noun, common noun or the letter I itself. There could be cases where the user might have wanted to create new sentences but did not use proper capitalization. So for cases where there were no capitalization we checked for the count of finite verbs, non-finite verbs, conjunctions and linkers such as ‘because’, ‘that’, ‘which’, ‘whose’. If the sentence had multiple finite verbs with no conjunctions/linkers in between we increased the total count of sentences. If there are conjunctions/linkers between them, such sentences are compound/complex sentences, hence we do not increase their count. In the exceptional case where there might not be any finite verbs in a sentence (by mistake) we consider the count of non-finite verbs.

Scoring:

- **If the number of sentences is less or equal to 5 we scored it as 1**
- **If the number of sentences is more than 5 but less or equal to 15 we scored it as 2**
- **If the number of sentences is more than 15 but less or equal to 20 we scored it as 3**
- **If the number of sentences is more than 20 but less or equal to 30 we scored it as 4**
- **If the number of sentences is more than 30 we scored it as 5**

b) Spelling Mistakes:

We used the Spellchecker to identify the spelling mistakes.

Scoring:

- **If the number of mistakes is less or equal to 5 we returned 0**
- **If the number of mistakes is more than 5 but less or equal to 15 we returned 1**
- **If the number of mistakes is more than 15 but less or equal to 25 we returned 2**
- **If the number of mistakes is more than 25 but less or equal to 40 we returned 3**
- **If the number of sentences is more than 40 we scored it as 4**

c1) Subject-Verb Agreement:

We completed this part by creating a tag rule set for NN, NNS, NNP, NNPS, and PRP tags. We used the following functions:

subject_verb_agreement_noun: defines rule set when subject is NN, NNS, NNP, or NNS and returns boolean value representing whether a subject-verb mismatch has been found.

subject_verb_agreement_pronoun: defines rule set when subject is PRP and returns boolean value representing whether a subject-verb mismatch has been found.

agreement: checks sequence of tags and uses subject-verb agreement functions above to check for mismatch and returns a score from 1-5 depending on number of errors found.

Scoring:

- **If the number of mistakes is greater than or equal to 9 we returned 1**
- **If the number of mistakes is more than 5 but less or equal to 15 we returned 1**
- **If the number of mistakes is between 6 and 8 we returned 2**
- **If the number of mistakes is between 4 and 5 we returned 3**
- **If the number of sentences is is between 2 and 3 we returned 4**
- **Else return 5**

c2) Verb-tense Agreement:

We completed this part by creating a tag rule set for known verbs and auxiliaries. We used the following functions:

verb_tense_agreement: defines rule set for known verbs and auxiliaries and returns boolean value representing whether a subject-verb mismatch has been found

verb: checks sequence of tags and uses verb-tense agreement function above to check for mismatch and returns a score from 1-5 depending on number of errors found

Scoring:

· **If the number of mistakes is greater than or equal to 5 we returned 1**

· **If the number of mistakes is more than 4 but less than 5 we returned 2**

· **If the number of mistakes is between 2-3 we returned 3**

· **If the number of mistakes is between 1-2 we returned 4**

· **Else return 5**

c3) Syntactic Wellformedness:

Implementation details and future improvement directions:

- Currently, our logic scores this criteria in two ways:
  - Whether a sentence falls under the categories [imperative, assertive, yes-no, wh-question]
  - Whether the bigram tag-pair for each tag pair in the sentence is correct based on a preset rule dictionary.
- To improve upon this, we can check for further categories when there are subordinate clauses
- We can fine-tune our tag-pair rule dictionary

Scoring:

We used the following mapping scheme based on our raw score calculation.

```
if raw_score>=1.64:

      mapped_score = 5

   elif raw_score>=1.48 and raw_score<1.64:

      mapped_score = 4

   elif raw_score>=1.32 and raw_score<1.48:

      mapped_score = 3

   elif raw_score>=1.14 and raw_score<1.32:

      mapped_score = 2

   elif raw_score<1.14:

      mapped_score = 1

   return mapped_score
```

d1) Topic Coherence:

Implementation details and future improvement directions:

- Currently, we are only using cosine similarity values between prompt and essay (after removing stop words from the essay) to check whether the essay is coherent with the topic of the prompt. Since this highly depends on the embedding used, we can experiment with embeddings other than word2vec to check for performance improvement.

Scoring:

We used the following mapping scheme based on our raw score calculation.

```
if raw_score<0.557:
    mapped_score = 1
elif raw_score>=0.557 and raw_score<0.637:
    mapped_score = 2
elif raw_score>=0.637 and raw_score<0.737:
    mapped_score = 3
elif raw_score>=0.737 and raw_score<0.82:
    mapped_score = 4
elif raw_score>=0.82:
    mapped_score = 5
return mapped_score
```

d2) Essay Coherence:

Future improvement directions:

- Currently, we are only using cosine similarity values between sentences to check for coherence. In the future, adding a function for implementing centering function that we learnt in class to get entity coreference coherence may improve our score.

Scoring:

We used the following mapping scheme based on our raw score calculation.

```
if pairwise_similarity<=(-2)*std:
   mapped_score = 1
elif pairwise_similarity>(-2)*std and pairwise_similarity<=(-1)*std:
   mapped_score = 2
elif pairwise_similarity>(-1)*std and pairwise_similarity<=(1)*std:
   mapped_score = 3
elif pairwise_similarity>(1)*std and pairwise_similarity<=(2)*std:
   mapped_score = 4
elif pairwise_similarity>(2)*std:
   mapped_score = 5
```

Here, std = standard deviation

Final Score Mapping:

We tested the scores of different essays and found the best fit for the following scenario:

```
if (Final_Score >= 38):
    print("This essay is classified as: high")
else:
    print("This essay is classified as: low")
```

Additional Report for Machine Learning models

Model	Accuracy	Precision	Recall	F1-Score
MLP	95%	1	0.89	0.94
Logistic Regression	100%	1	1	1
Naive Bayes	95%	1	0.89	0.94
Linear Combination	84.85%	0.85	1	0.92

Here we can see that the Machine Learning models performed better than the linear combination, the reason being, that the machine learning models can detect complex relations and interactions between features much better than the linear combination.