Term Project: Automatic Essay Scorer

Group Members – Raj Mehta (UIN: 670579653)

Aarsh Patel (UIN: 659999805)

For our term project we made an automatic essay scorer that evaluates an essay answered in response to a prompt and outputs a score – high or low.

As part of this project, we made a pipeline that evaluates each essay based on the following criteria:

1. Length of the Essay
2. Spelling Mistakes
3. Subject-Verb Agreement
4. Verb Tense Usage
5. Syntactic Structure
6. Semantic Structure and Coherence
7. Relation to the Prompt

For evaluation of each of these aspects, we tried out a variety of methods using the NLTK and SpaCy packages of Python. First, we used a simple approach to compute the length of the essay – we took this as the number of sentences attribute of the document generated by nlp in SpaCy. We compared this approach with a more sophisticated method of counting main clauses and subordinate clauses and found that both methods returned the same value, hence we kept the first approach. However, this wasn't adequate in niche settings such as having multiple sentences (in terms of meaning) in a single sentence.

For computing spelling mistakes, we first tried tokenizing the essay and then extract word stems using a lemmatizer. Then used PySpellChecker to compute the unknown tokens, which we take as the number of spelling mistakes. However, this approach counted contractions like "won't" as a spelling mistake, hence we added a preprocessing step of decontracting such words, for example, "won't" to "will not". This solved the problem.

In order to check for subject-verb disagreements, we analyzed a few texts and listed the POS tag sequences that would occur in a wrong subject-verb sequence. In each sentence, we then checked if there were any such disagreements. This simple approach allowed us to capture most of the mistakes, however, since we cannot list all the possible wrong POS tag sequences, we won't be able to capture rarely occurring disagreements.

To check for verb tense usage mistakes, we initially took a rule-based approach and tried checking each sentence for such patterns. However, this was inadequate as we could only list a few rules. Thus, we took a probabilistic approach where we used the Brown corpus to compute bigram probabilities of the POS tag sequences and stored it. Then for each sentence, we compute the probabilities of each POS tag pair and if it is lower than 5% then we classify that as a mistake. This improved the accuracy significantly.

In order to check if the essay has any syntactical mistakes, we first stored all the various parent-child-grandchild relations that occur in the parse trees of all Brown corpus sentences. We then check in the parse tree of a given sentence, whether all such parent-child-grandchild relations match with the any of the ones obtained from Brown; if not, then classify that as a mistake.

To check if a given essay is coherent or not, we compute sentence-level embeddings and compute pairwise cosine similarities between adjacent sentences. If the cosine similarity of adjacent pairs changes by more than 0.2 then we increment the incoherence counter by 1. This approach worked well for majority cases, however, for some "high" score essays we obtained low coherence and vice-versa. We tried fixing this by changing the threshold values but 0.2 gave us the best results.

To understand whether the essay answers the prompt, we compute the TF-IDF vectors of the relevant part of the prompt and the essay and compute the cosine similarity between them. Based on these values, we mapped them to a score between 1 and 5.

Using these methods, we first stored the values obtained for each essay in their respective lists (sentence length list, number of mistakes list, etc.). Then, we first tried to map this to a score range from min_score to max_score using a linear function. However, the linear assumption may not hold in general as most values were centered closely around the mean. Hence, we made the assumption that the values followed a Gaussian distribution and computed z-scores from the values. We then mapped these z-scores to the final score from min_score to max_score range.

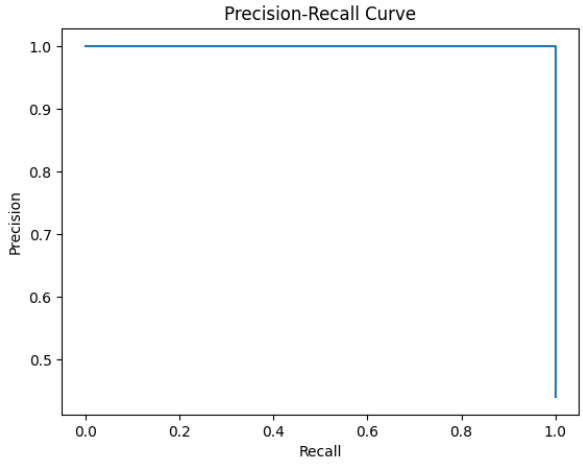
We combined the scores from each of these factors using the formula:

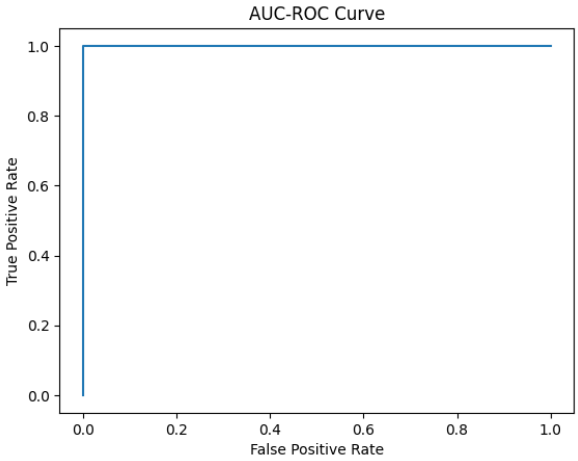
$$\begin{aligned} Final\ Score = {} & 2 * essay\ length\ score - spelling\ score + agreement\ score \\ & + verb\ tense\ score + syntax\ score + 3 * address\ prompt\ score \\ & + coherence\ score \end{aligned}$$

We then tried out various values of threshold to classify the essay with a final score into "high" or "low" and found that a threshold value of 20 works the best, i.e., a essay with a final score of less than 20 is "low", otherwise, it is "high". This gave us an accuracy of 68%.

Graduate Students Part: Training ML Classifiers

Using the raw scores of each of the components, we first constructed a Pandas dataframe with the raw. Then, we split it into a train and test set using random shuffling, with 25% as the test set. Then we trained three ML Classifiers – Naïve bayes, Logistic Regression, and Multi-Layer Perceptron classifier on the train set, and tested the models on the test set. We obtained the following results for all three classifiers:





$$Confusion\ Matrix = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	10
accuracy			1.00	20
macro avg	1.00	1.00	1.00	20
weighted avg	1.00	1.00	1.00	20

With all three classifiers, we obtained a 100% accuracy, which is significantly better than the 68% accuracy obtained using the linear combination. This means that a non-linear combination explains the relationship between various factors and the final “high”/ “low” score assigned by human evaluators. However, since we only have 100 essays the data is not capturing the wide variety of essays that students would write in response to the myriad of prompts that may come in an exam. Hence, high capacity ML models would work really well and give 100% accuracy on such small datasets, however, the accuracy would reduce on more general, comprehensive datasets to realistic values.