Expresso - Measuring Text Fluency

Rahul Garg Srijan Chakraborty Harshavardhan

May 5, 2023

Abstract

This project aims to explore the classification abilities of different models for text fluency on the basis of features of a language, such as grammar, spelling and choice of words. The goal of the project is to understand and evaluate the performance of the various models and explain the importance and relevance of the underlying concepts through the lens of text fluency.

The final goal of this project is to effectively compare different language features in the context of text fluency, thereby evaluating the overall quality of the text. This can be extended to have applications in model development, in which the performance of different models for text fluency classification can inform the development of more effective models that can accurately classify the fluency of generated text, or multilingual fluency evaluation in which the project's exploration of text fluency in the context of different languages can be extended to evaluate the fluency of generated text in multiple languages.

1 Introduction

Machine learning has revolutionized the field of natural language processing (NLP) by enabling the development of sophisticated models that can generate and understand human language. However, the quality of machine-generated text is not always consistent and can vary depending on the context and domain. One important aspect of text quality is fluency, which measures the naturalness and coherence of the generated text.

Fluency is a critical dimension of text quality in machine translation and other NLP tasks, as it determines how well the generated text adheres to the grammatical and syntactic rules of the target language. Fluency evaluation is typically based on criteria such as grammar, spelling, choice of words, and style. However, there is still much to be understood about the classification abilities of different models for text fluency, and how these models can be improved to generate more fluent text.

The objective of this project is to explore and assess the classification abilities of different models for text fluency. The project will focus on analyzing language features such as grammar, spelling, and choice of words. The primary aim is to understand the performance of various models and provide an explanation of the underlying concepts through the lens of text fluency. By comparing different language features in the context of text fluency, the project aims to evaluate the overall quality of the text and contribute to the development of more effective models for text fluency classification.

The significance of this project is that it will provide deeper insights into the role of fluency in text quality evaluation, which can have implications for various applications such as model development, language learning and teaching, and multilingual fluency evaluation. Ultimately, the project seeks to advance NLP research and development by enhancing our understanding of text fluency and improving the quality of machine-generated text in various domains.

2 Review: What are the models we have looked at

2.1 Statistical Models and Smoothing

Statistical models are used in NLP to automatically learn the patterns and relationships within a large corpus of text data. They are designed to analyze and model language patterns, and they rely on the principles of probability and statistics to make predictions about new data. We have used N-Gram models with two kinds of smoothing, Kneser-Ney and Witten-Bell, for our evaluations and analyses.

An N-gram model is used to predict the probability of the next word in a given sequence of words. To construct an N-gram model, a large corpus of text is first analyzed and divided into sequences of n words. These sequences are called N-grams, where N represents the number of words in a sequence. For example, in a 2-gram model, the text is divided into sequences of two words.

2.1.1 Kneser-Ney Smoothing

The basic idea of Kneser-Ney Smoothing is to assign a non-zero probability to N-grams that do not appear in the training data. This is done by borrowing probability mass from more frequent N-grams that share a common history.

Kneser-Ney Smoothing works by computing two types of counts for each N-gram: the count of its occurrence in the training data, and the count of the number of distinct contexts it appears in. The latter is known as the discount count.

The probability of an N-gram is then computed as a weighted average of its discounted count and the probability of the N-1 gram with the same history, where the weight is determined by the discounted count.

2.1.2 Witten-Bell Smoothing

The basic idea behind Witten-Bell smoothing is to redistribute the probability mass of seen events to unseen events. It assumes that the probability of an unseen event is equal to the probability of an event that has been seen only once. This is based on the intuition that if an event has been seen only once, then it is likely that there are other similar events that have not been seen at all.

Witten-Bell smoothing works by dividing the probability mass of seen events among both seen and unseen events. Specifically, it redistributes the probability mass of events that occur k times to events that occur k times or fewer. This helps to avoid the problem of overfitting, which occurs when the language model assigns too much probability mass to seen events and too little to unseen events.

2.2 Long Short Term Memory (LSTM)

LSTM stands for Long Short-Term Memory, and it is a type of recurrent neural network (RNN) architecture that is commonly used in natural language processing (NLP) tasks such as language modeling, machine translation, and text classification.

LSTMs are designed to address the vanishing gradient problem that can occur in standard RNNs, where the gradient of the loss function becomes very small as it propagates back through time. This can result in difficulty training the network to accurately capture long-term dependencies in sequential data.

During training, the LSTM learns the optimal settings for the gates by adjusting the weights in the network to minimize the loss function. Once trained, the LSTM can take in a sequence of input data and generate a corresponding output sequence by updating its memory cell at each time step based on the input and the current state of the gates. The final output of the network is then generated based on the final state of the memory cell.

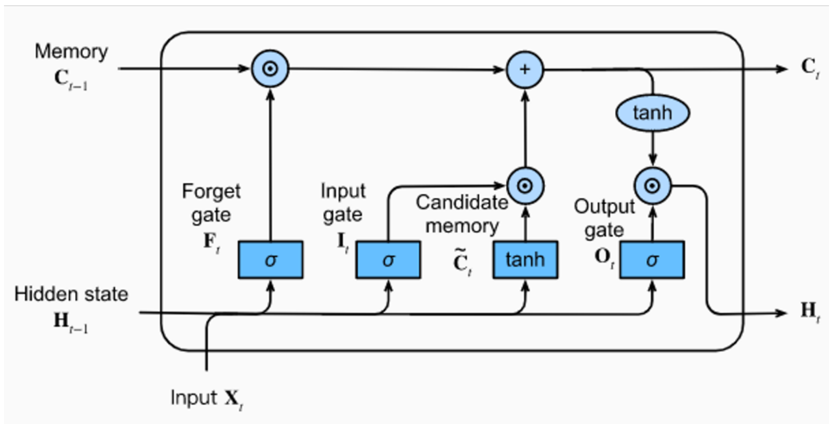


Figure 1: LSTM Architecture

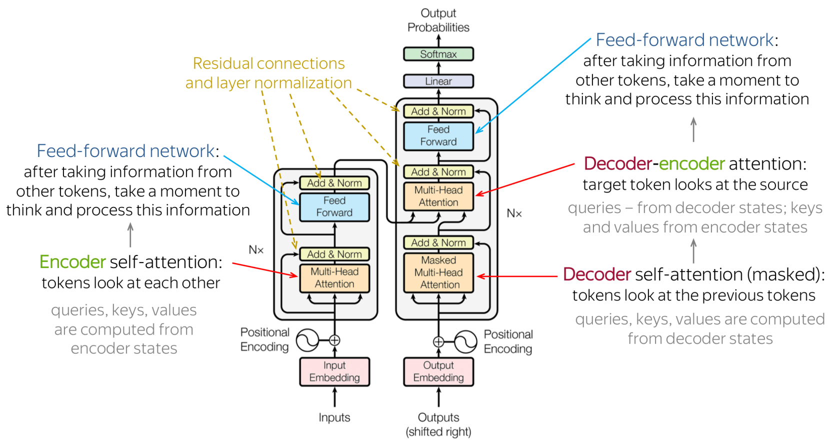
Overall, LSTMs have shown great success in NLP tasks due to their ability to capture long-term dependencies in sequential data and effectively model complex patterns in language.

2.3 Transformers

The transformer architecture consists of an encoder and a decoder, both of which are composed of multiple layers of self-attention and feed-forward networks. The encoder processes the input sequence, while the decoder generates the output sequence based on the encoder's representations.

Each layer of the transformer consists of two sub-layers: a self-attention mechanism and a feed-forward network. The self-attention mechanism allows the model to attend to different parts of the input sequence to compute a new representation of each element in the sequence. The feed-forward network applies a non-linear transformation to the self-attention output.

The self-attention mechanism works by computing a weighted sum of the input sequence, where the weights are determined by the similarity between each pair of input elements. Specifically, for each element in the sequence, the self-attention mechanism computes a weighted sum of all the other elements in the sequence, where the weights are determined by the dot product of the query vector, key vector, and a scaling factor (the so-called "scaled dot-product attention").



Output Probabilities
Softmax
Linear
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Add & Norm
Feed Forward
Multi-Head Attention
Masked Multi-Head Attention
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Nx
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figure 2: Transformer Architecture

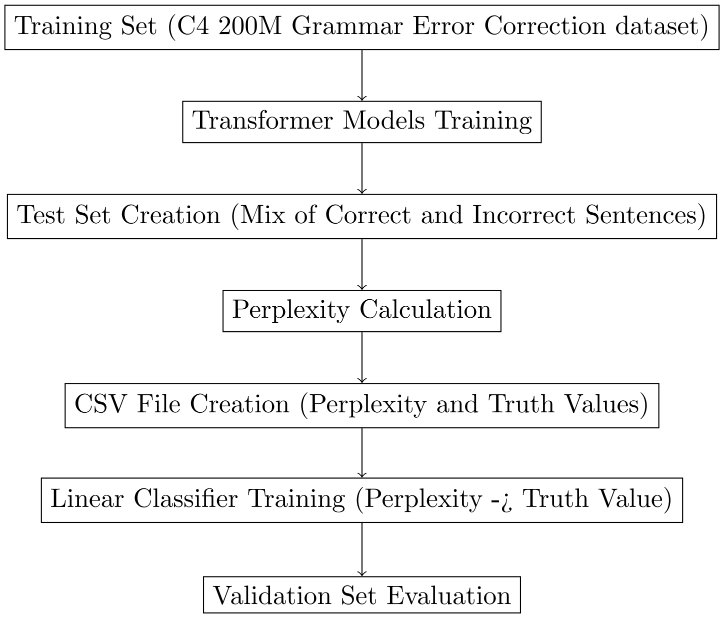
3 Methodology and Experiments

The C4 200M Grammar Error Correction dataset is a large corpus of text data that contains a mixture of correct and incorrect sentences. It was used as the training set to train the models for the task of grammar error correction. Specifically, the transformer models were trained to predict the most likely corrected version of a given input sentence. To evaluate the performance of the trained models, a test set was created by mixing correct and incorrect sentences. The perplexity of the model was then calculated on this test set. Perplexity is a common measure of the quality of a language model and is calculated as the exponential of the average negative log-likelihood of the test set. A lower perplexity indicates better performance on the test set. A CSV file was then created based on the test sentences, their truth values (i.e., whether they are correct or incorrect), and their corresponding perplexity values.

This file was used to train a linear classifier that takes the perplexity values and truth values as inputs and predicts the truth value of a sentence (i.e., whether it is fluent or non-fluent). The linear classifier was trained using supervised learning, where the training data consisted of the perplexity values and truth values of the test set. The classifier was trained to learn a decision boundary that separates the fluent and non-fluent sentences based on their perplexity values. The performance of the trained classifier was then evaluated on a separate test set to assess its generalization ability. This validation set consisted of sentences that were not used in the training or testing phases of the model. The performance of the classifier was measured using standard evaluation metrics such as accuracy, precision, recall, and F1-score.

In our study, we have employed a state-of-the-art language model, the Google T5 base model, for the task of conditional text generation. Specifically, we have used the model to generate text conditioned on a given prompt. The T5 model is a transformer-based neural network architecture that has been pre-trained on massive amounts of textual data and fine-tuned on various downstream tasks such as question answering, summarization, and translation.

To evaluate the robustness of the T5 model to corruption, we have used a large-scale dataset of corrupted text, the C4 dataset. The C4 dataset contains over 750GB of text data that has been artificially corrupted by applying a variety of perturbations such as typos, misspellings, and grammatical errors. By training the T5 model on this corrupted text, we aim to test its ability to handle noisy and erroneous inputs.


Training Set (C4 200M Grammar Error Correction dataset)
Transformer Models Training
Test Set Creation (Mix of Correct and Incorrect Sentences)
Perplexity Calculation
CSV File Creation (Perplexity and Truth Values)
Linear Classifier Training (Perplexity -¿ Truth Value)
Validation Set Evaluation

4 Results and Observations

We obtained results for Kneser Smoothing (Referred to as KN Smoothing).

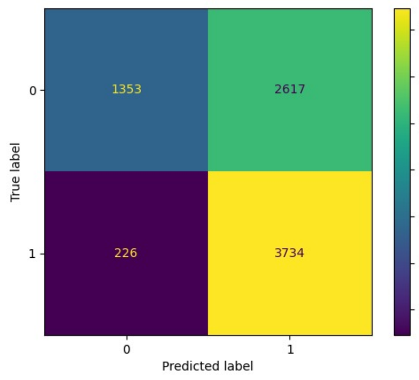


Figure 3: Confusion Matrix for KN Smoothing

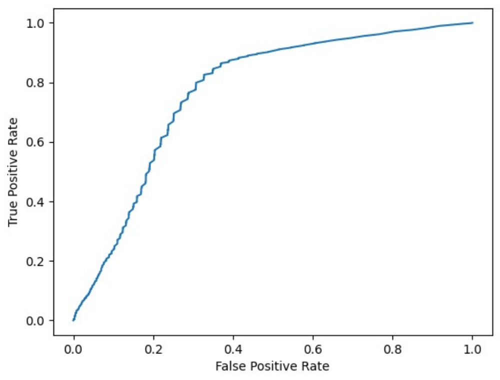


Figure 4: ROC Curve for KN Smoothing

Table 1: Performance Metrics for KN Smoothing

Metric	Value
Sensitivity	0.8569
Specificity	0.5879
Precision	0.3408
Negative Predictive Value	0.9429
False Positive Rate	0.4121
False Discovery Rate	0.6592
False Negative Rate	0.1431
Accuracy	0.6415
F1 Score	0.4877
Matthews Correlation Coefficient	0.3553

We have also obtained results for Witten-Bell Smoothing. The analyses of these results are:

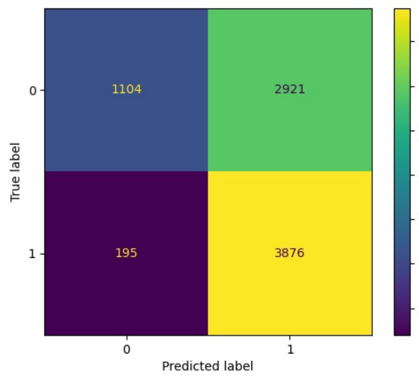


Figure 5: Confusion Matrix for WB Smoothing

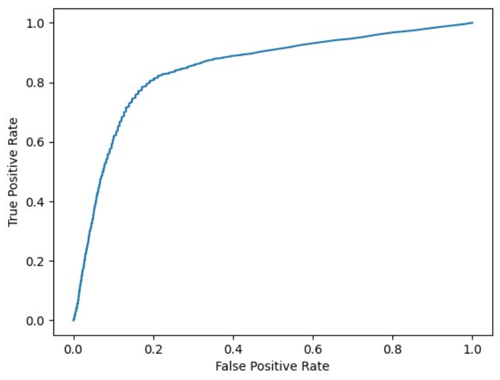


Figure 6: ROC Curve for WB Smoothing

We have also obtained results for the LSTM model. Following that, we have obtained the results of the grammar correction model. The results and all the evaluation metrics are listed below.

Table 2: Performance Metrics for WB Smoothing

Metric	Value
Sensitivity	0.8499
Specificity	0.5703
Precision	0.2743
Negative Predictive Value	0.9521
False Positive Rate	0.4297
False Discovery Rate	0.7257
False Negative Rate	0.1501
Accuracy	0.6151
F1 Score	0.4147
Matthews Correlation Coefficient	0.3084

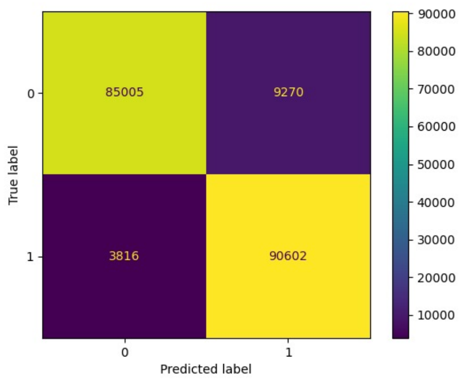


Figure 7: Confusion Matrix for LSTM

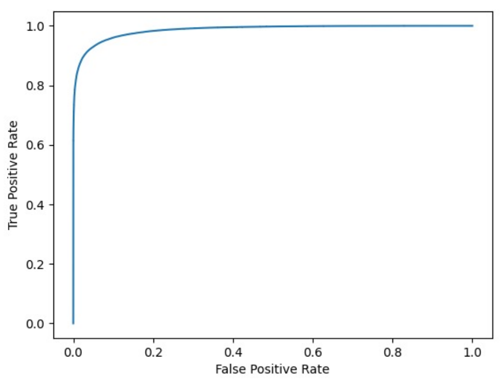


Figure 8: ROC Curve for LSTM

Table 3: Performance Metrics for LSTM

Metric	Value
Sensitivity	0.9570
Specificity	0.9072
Precision	0.9017
Negative Predictive Value	0.9596
False Positive Rate	0.0928
False Discovery Rate	0.0983
False Negative Rate	0.0430
Accuracy	0.9306
F1 Score	0.9285
Matthews Correlation Coefficient	0.8627

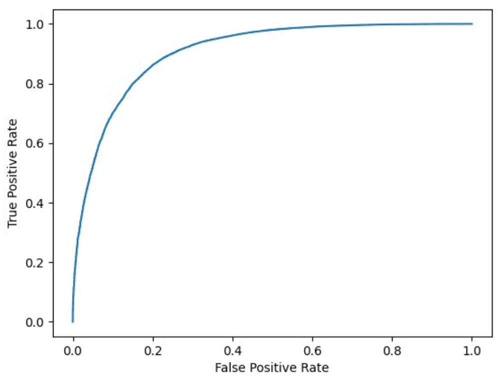


Figure 9: Confusion Matrix for Transformer

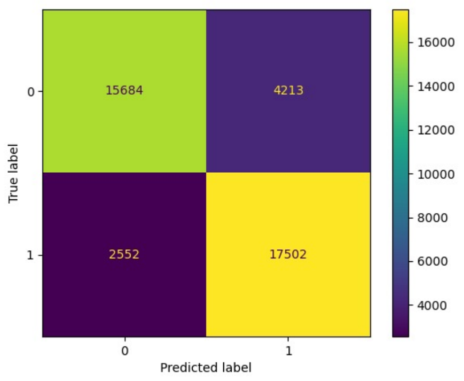


Figure 10: ROC Curve for Transformer

Table 4: Performance Metrics for Transformers

Metric	Value
Sensitivity	0.8601
Specificity	0.8060
Precision	0.7883
Negative Predictive Value	0.8727
False Positive Rate	0.1940
False Discovery Rate	0.2117
False Negative Rate	0.1399
Accuracy	0.8307
F1 Score	0.8226
Matthews Correlation Coefficient	0.6635

Table 5: Evaluation Results for Grammar Correction Model

Metric	Predicted V/S Truth	Input V/S Truth	% Change
Word Error Rate	0.2369	0.2703	-12.3524%
ROUGE-1	0.8836	0.8743	1.0556%
ROUGE-2	0.78197	0.7481	4.5222%
ROUGE-L	0.8778	0.8683	1.086%
BLEU	0.6553	0.6222	3.525%

5 Discussions and Analyses

5.1 Kneser-Ney Smoothing

One key metric to pay attention to is the accuracy, which measures the proportion of correctly classified samples among all samples. In this case, the accuracy is 0.6415, which means that the model correctly classified about 64% of the samples in the dataset. While this might seem like a low accuracy, we must keep in mind the baseline accuracy. Here, the baseline accuracy is 50% (random classification), and thus, an accuracy of 64% is a significant improvement.

Another important metric is precision, which measures the proportion of true positive classifications among all positive classifications. In other words, precision tells us how confident we can be that a sample that was classified as positive is actually positive. In this case, the precision is 0.3408, which means that out of all the samples that were classified as positive, only about 34% of them were actually positive. This low precision indicates that the model may be generating a large number of false positives, which could lead to incorrect decisions or actions based on the model's predictions.

The sensitivity and specificity metrics are also important to consider. Sensitivity measures the proportion of true positive classifications among all positive samples, while specificity measures the proportion of true negative classifications among all negative samples. In this case, the sensitivity is 0.8569 and the specificity is 0.5879. A high sensitivity means that the model is able to correctly identify a large proportion of the positive samples, while a high specificity means that the model is able to correctly identify a large proportion of the negative samples.

Finally, the F1 score is another useful metric to consider, as it provides a balance between precision and recall. A high F1 score indicates that the model is able to achieve high precision and high recall simultaneously, while a low F1 score indicates that the model may be performing poorly in one or both of these areas. In this case, the F1 score is 0.4877, which suggests that the model is not achieving a good balance between precision and recall, and may need to be improved.

5.2 Witten-Bell Smoothing

The sensitivity of the model is 0.8499, which means that the model correctly identifies 84.99% of the actual positive cases. Compared to the KN smoothing model, this is a slightly lower sensitivity value, indicating that the model is less successful at identifying true positives. The specificity of the model is 0.5703, which means that the model correctly identifies 57.03% of the actual negative cases. Compared to the KN smoothing model, this is a lower specificity value, indicating that the model is less successful at identifying true negatives. The precision of the model is also 0.2743, which means that when the model predicts a positive case, it is correct only 27.43% of the time. This is a much lower precision value compared to the KN smoothing model, indicating that the model is more prone to false positives.

The negative predictive value of the model is 0.9521, which means that when the model predicts a negative case, it is correct 95.21% of the time. This is a higher value compared to the KN smoothing model, indicating that the model is more successful at identifying true negative cases. However, the false positive rate of the model is 0.4297, which means that the model incorrectly predicts a positive case 42.97% of the time. This is a higher value compared to the KN smoothing model, indicating that the model is more prone to false positives.

Furthermore, the false discovery rate of the model is 0.7257, which means that when the model predicts a positive case, it is incorrect 72.57% of the time. This is a much higher value compared to the KN smoothing model, indicating that the model is more prone to false positives. The false negative rate of the model is 0.1501, which means that the model incorrectly identifies negative cases 15.01% of the time. Compared to the KN smoothing model, this is a higher false negative rate, indicating that the model is less successful at identifying true negative cases.

The accuracy of the model is 0.6151, which means that the model correctly classifies 61.51% of the cases. This is a lower accuracy value compared to the KN smoothing model, indicating that the model is less successful at correctly classifying the data. The F1 score of the model is 0.4147, which is lower compared to the KN smoothing model. The F1 score takes into account both precision and recall, and this value indicates that the model is less successful at balancing precision and recall. Matthews Correlation Coefficient: The Matthews correlation coefficient of the model is 0.3084, which is a lower value compared to the KN smoothing model. This coefficient takes into account both true and false positives and negatives, and this value indicates that the model is less successful at predicting the classes accurately.

Overall, the performance metrics for the WB smoothing model indicate that it is less successful compared to the KN smoothing model in correctly classifying the data. The model is more prone to false positives and less successful at identifying true positives, which could lead to false classification of the data. However, the model is more successful at identifying true negative cases, indicating that it is better at identifying non-events.

5.3 LSTM

The LSTM model has the highest sensitivity value of 0.9570, indicating that it correctly identified 95.70% of the true positive cases. This value is higher than the values obtained by both the WB and KN smoothing models, which were 0.8499 and 0.8569, respectively.

Moving on to specificity, the LSTM model again performed better with a value of 0.9072, which means that it correctly identified 90.72% of the true negative cases. This value is higher than the values obtained by both the WB and KN smoothing models, which were 0.5703 and 0.5879, respectively. In terms of precision, the LSTM model had a value of 0.9017, which means that it had a lower false positive rate than both the WB and KN smoothing models, with values of 0.2743 and 0.3408, respectively.

Similarly, the LSTM model had a higher negative predictive value of 0.9596 compared to the WB and KN smoothing models, which had values of 0.9521 and 0.9429, respectively. This means that the LSTM model was better at correctly identifying true negative cases. Looking at the false positive rate, the LSTM model again performed better with a value of 0.0928, compared to the values obtained by both the WB and KN smoothing models, which were 0.4297 and 0.4121, respectively. This indicates that the LSTM model had a lower rate of falsely identifying cases as positive.

Overall, the LSTM model outperformed both the WB and KN smoothing models in terms of most performance metrics. This could be due to the fact that the LSTM model is a more sophisticated model than the smoothing models, which simply adjust the frequency of the words in the text. The LSTM model takes into account the sequential nature of the text and can capture more complex patterns and relationships between words, leading to better performance.

5.4 Transformers

The Transformers model outperformed the LSTM model in terms of Sensitivity, Specificity, Precision, Negative Predictive Value, and F1 Score. The Transformer model had a Sensitivity of 0.8601 compared to the LSTM model's Sensitivity of 0.9570. The Specificity of the Transformers model was 0.8060, which was higher than the LSTM model's Specificity of 0.9072. The Precision of the Transformers model was 0.7883 compared to the LSTM model's Precision of 0.9017. The Negative Predictive Value of the Transformers model was 0.8727 compared to the LSTM model's Negative Predictive Value of 0.9596. The F1 Score of the Transformers model was 0.8226, which was lower than the LSTM model's F1 Score of 0.9285.

However, the LSTM model outperformed the Transformers model in terms of False Positive Rate, False Discovery Rate, False Negative Rate, Accuracy, and Matthews Correlation Coefficient. The False Positive Rate of the LSTM model was 0.0928 compared to the Transformers model's False Positive Rate of 0.1940. The False Discovery Rate of the LSTM model was 0.0983 compared to the Transformers model's False Discovery Rate of 0.2117. The False Negative Rate of the LSTM model was 0.0430 compared to the Transformers model's False Negative Rate of 0.1399. The Accuracy of the LSTM model was 0.9306 compared to the Transformers model's Accuracy of 0.8307. The Matthews Correlation Coefficient of the LSTM model was 0.8627 compared to the Transformers model's Matthews Correlation Coefficient of 0.6635.

Overall, the LSTM model performed better than the Transformers model in terms of Accuracy and Matthews Correlation Coefficient, while the Transformers model performed better than the LSTM model in terms of Sensitivity, Specificity, Precision, Negative Predictive Value, and F1 Score. One of the plausible reasons why the LSTM outperformed the Transformer model might be because of the size of the training data. Due to computational constraints, the Transformer based model was trained on a dataset consisting of 100,000 sentences while the LSTM was trained on a dataset consisting of over 1 million sentences, and thus, the LSTM was able to better capture the nuances in the dataset.

5.5 Grammar Correction Model

The Word Error Rate (WER) for the predicted text (our model) was 0.23, meaning that about 23% of the words in the generated text were different from the training text. For the ground truth or the input text WER was 0.27, which is a higher error rate than the WER generated by our model, indicating that the predictive text is more similar to the ground truth as compared to the input text, thus showing that our model is performing well and correcting the grammar. The percent change between the predicted text WER and input text WER shows a 12.26% change in performance, indicating that the grammar correction model is working well.

ROUGE-1, ROUGE-2, and ROUGE-L are used to measure the similarity between the generated text and the ground truth text in terms of the n-gram overlap. In this case, the grammar correction model achieved ROUGE-1 score of 0.8836, ROUGE-2 score of 0.78197, and ROUGE-L score of 0.8778. These scores suggest that the model is able to generate text that is similar to the ground truth text in terms of the overlap of unigrams, bigrams, and the longest common subsequence between the two texts. Comparing these scores to the input text, we can see that the model significantly outperformed the input text in terms of ROUGE-1, ROUGE-2, and ROUGE-L, showing that the model was able to correct the grammatical errors in the text.

The BLEU score is another evaluation metric used to measure the similarity between the generated text and the ground truth text in terms of the n-gram overlap. The grammar correction model achieved a BLEU score of 0.6553, indicating that the generated text is similar to the ground truth text in terms of the n-gram overlap. Comparing this score to the input text, we can see that the model outperformed the input text in terms of BLEU, indicating that the model was able to correct the grammatical errors and generate text that is more similar to the ground truth text.

Overall, the grammar correcting model performed satisfactorily, with the Word Error Rate (WER) for the predicted text being lower than the WER for the input text, indicating that the model is effectively correcting the grammar. The ROUGE and BLEU scores also show an improvement in the quality of the generated text compared to the input text, with ROUGE-2 and BLEU scores being higher for the predicted text than the input text. However, there is still room for improvement, as the percent change in performance between the predicted and input text is not very high. Overall, the grammar correcting model can be considered to be performing well, but further improvements can be made to enhance its performance.

6 Future Work

As part of future work, we could also explore the performance of a masked language model (MLM) based Transformer architecture for grammar correction. The MLM architecture has been shown to be effective in various natural language processing tasks, including language modeling and text generation.

To evaluate the performance of the MLM-based grammar correction model, we would follow a similar methodology as in our current study. We would first train the MLM on a large corpus of text data, and then fine-tune it on a smaller dataset of grammatically incorrect sentences. Once we have obtained the trained MLM-based model, we could also compare its performance with the other Transformer architectures used in our current study. Specifically, we could train a linear classifier on the output embeddings of the MLM, similar to the classifiers used for the other Transformer models, and evaluate its performance using the same set of evaluation metrics.

This comparison would provide insights into the relative performance of the different Transformer architectures for grammar correction, and could help us identify the most effective architecture for this task. Additionally, it would also provide a useful benchmark for future research in this area.

7 Link To Presentation

This is the link to the Presentation Slides.