Tokenized Multilingual Grammar Error Detection

Vishnu S vsabbavarapu@umass.edu
Abhinav Y ayadatha@umass.edu
Varun B vbachireddy@umass.edu
Harsha G hgudipudi@umass.edu

1 Problem statement

Grammar Error detection (GED) is the task of detecting grammatical errors in a given text. The task of Tokenized Grammar Error Detection(TGED), each and every token of the text is considered and predicts whether each word is grammatically correct or not. In other words, it performs binary text classification of each token as 'Correct' or 'Incorrect'. It is important to note that the task of Tokenized Grammar Error detection makes its prediction on two grammatical aspects : vocabulary and context of each token in the sentence. This task is very useful for language learning applications where learners get to identify and correct their grammar , especially if they are new to the language. This task is also very helpful in an education setting where the students want to improve their writing skills, proofread their essays and assignments. Most of the recent methods in the field of grammar error detection have been focused on Grammar Error Correction, where the grammatically incorrect text is parsed by the model to suggest a correction. While most of the existing methodologies for Grammar Error Correction which have state-of-the-art detection systems, we are trying to accomplish tokenized grammar error detection, where each token would be categorized as Incorrect or Correct based on its vocabulary and the context. Another challenge is that most of the work is on grammar detection for a single language, we don't have a lot of resources for bi-lingual grammar error detection . We intend to perform analysis of multiple models for the task of tokenized error detection. We intend to fine-tune pre-existing models such as Google's BERT and OpenAi's GPT for the task of tokenized grammar error detection. We plan to extend the model's functionality by training two different mono-lingual models and then training a bilingual model for tokenized grammar error detection.

2 What you proposed vs. what you accomplished

- ~~Process the FCE, MERLIN, REALEC and SweLL datasets provided by the MultiGED-2023 shared task to be used for various Language Models.~~
- ~~Fine-tune a base DisitilBERT model as a Baseline and evaluate it for English dataset.~~
- ~~Fine-tuning the BERT model for multilingual error detection on English, Italian, Swedish and Czech datasets.~~
- ~~Also, Fine-tune more complex LM's like T5, ROBERTA and GPT2 for GED of English language effectively as mentioned earlier.~~
- ~~Comparison of results with the Baselines chosen for metrics like Precision, Recall and F0.5 score comprehensively.~~

3 Related work

Grammatical Error Detection (GED) in natural language text can be primarily addressed through the use of language models. The emergence of advanced language models such as BERT and GPT-3 has spurred the development of numerous approaches to tackle GED. However, the idea of addressing grammatical errors has its roots in the past, leading to the proposal of several approaches.

Among these approaches, rule-based and classification-based methods failed to identify dependent errors. However, the introduction of Statistical Machine Translation (SMT) models significantly advanced GED research. (Yuan and Felice, 2013) described the development of an SMT model that highlighted critical issues in error detection.(Ng et al., 2014) successfully evaluated

various systems for detecting and correcting all types of grammatical errors, with many of these systems proposed in CoNLL-2014 being based on Language models.

Neural Machine Translation (NMT) using bi-directional encoders and attention-based decoders emerged as another approach for detecting grammatical errors. (Grundkiewicz and Junczys-Dowmunt, 2018) introduced a hybrid model combining SMT and NMT, which achieved state-of-the-art results. Their approach involved a phrasal-based SMT system and a 5-gram Language Model. For the NMT system, an attentional Encoder-Decoder model with a bidirectional single-layer encoder and decoder was employed.

The introduction of pre-trained models based on the Transformer architecture((Vaswani et al., 2017)), such as GPT((Radford and Narasimhan, 2018)) and BERT(Devlin et al., 2019), triggered significant progress in various NLP solutions. Fine-tuning these models with task-specific data has yielded remarkable outcomes for multiple NLP tasks. Consequently, leveraging such pre-trained models for the task of Grammatical Error Detection (GED) appears to be a feasible approach. (Alikaniotis and Raheja, 2019)recently revived the effectiveness of various transformer language models like BERT, GPT-1, and GPT-2 in GED, providing valuable insights that have motivated us to undertake this task.

By summarizing the development and evolution of GED approaches, highlighting key contributions and advancements, we are inspired to explore and build upon the state of the art in GED research.

4 Dataset

MultiGED-2023 dataset : The data for the task of tokenized Grammatical Error Detection will be used from the github repository: MultiGED-2023 Github repository. This data contains tokenized classification text where several sentences are split into tokens and each token is classified as a grammatically correct or incorrect token. The dataset is not just confined to English but also contains text in Czech, German, Italian and Swedish. There are several familiar datasets used in the past for English corpus.For the English language, there are two datasets which we used for this task : FCE and REALEC.

Source corpus	Split	Proportion	Nr sentences	Nr tokens	Nr errors	Error rate
FCE	Total	1.0	33,243	531,416	50,860	0.096
	• train	0.855	28,357	454,736	42,899	0.094
	• dev	0.065	2,191	34,748	3,460	0.100
	• test	0.079	2,695	41,932	4,501	0.107

Figure 1: FCE Dataset statistics

Source corpus	Split	Proportion	Nr sentences	Nr tokens	Nr errors	Error rate
REALEC	Total	1.0	8136	177,769	16,608	0.093
	• train	n/a	n/a	n/a	n/a	n/a
	• dev	0.5	4067	88,008	8103	0.092
	• test	0.5	4069	89,761	8505	0.095

Figure 2: REALC Dataset statistics

FCE : The FCE tokenized dataset is a widely used dataset for this task in English. The FCE Corpus (Yannakoudakis et al., 2011) consists of essays written by candidates for the First Certificate in English (FCE) exam (now ”B2 First”) designed by Cambridge English to certify learners of English at CEFR level B2. The publicly released subset of the dataset, named FCE-public, consists of 33,673 sentences split into test and training sets of 2,720 and 30,953 sentences, respectively.

REALEC : The Russian Error-Annotated Learner English Corpus (REALEC) is a freely accessible corpus provided by HSE University. It comprises around 18,700 texts authored by HSE students who took the Independent English Language Test between 2014 and 2020. These texts were produced in response to two different task types, resulting in a total word count of approximately 4,336,000 words. The publicly released subset of the dataset, named REALEC-public, consists of 8136 sentences which we used for the purpose of training along with FCE dataset.

Example Input Sentence : ’I went to saws the movie.’. The input sentence is then tokenized.Tokenized Input Sentence : [I, went, to, saws, the, movie]From the tokenized input sentence, for each token a there is a label: correct or incorrect. This indicates if the token is grammatically and contextually correct.

Label predictions : [c,c,c,c,c,i]

Here, the label ’c’ indicates that the token is gramatically or contextually correct and the label ’i’ indicates that the token is gramatically or contextually incorrect. It is visible from the example that the word ’saws’ is incorrect

Italian MERLIN Dataset :

Italian data

Source corpus	Split	Proportion	Nr sentences	Nr tokens	Nr errors	Error rate
MERLIN	Total	1.0	7949	99,698	14,893	0.149
	• train	0.806	6394	80,336	12,190	0.152
	• dev	0.092	758	9144	1,211	0.132
	• test	0.102	797	10,218	1,492	0.146

Figure 3: MERLIN Dataset statistics

Czech data

Source corpus	Split	Proportion	Nr sentences	Nr tokens	Nr errors	Error rate
GECCC	Total	1.0	35,453	399,742	84,041	0.210
	• train	0.833	29,795	332,875	67,525	0.203
	• dev	0.080	2,780	31,940	8040	0.252
	• test	0.087	2,878	34,927	8476	0.243

Figure 4: GECCC Dataset statistics

The MERLIN dataset consists of a total of 7,949 sentences and 99,698 tokens. It contains 14,893 errors, resulting in an overall error rate of 0.149. The train subset comprises 80.6 percent of the dataset, with 6,394 sentences, 80,336 tokens, and an error rate of 0.152. The dev and test subsets represent 9.2% and 10.2% of the dataset, respectively, with 758 sentences, 9,144 tokens, and an error rate of 0.132 for the dev subset, and 797 sentences, 10,218 tokens, and an error rate of 0.146 for the test subset.

Example Input sentence : ”Sono molto communicative e seria.” This sentence is then tokenized into smaller tokens by the tokenizer.

tokens = [Sono , molto, communicative, e, seria,.] From the tokenized input sentence, for each token a there is a label: correct or incorrect. This indicates if the token is grammatically and contextually correct.

Labels predictions = [c,c,i,c,c,c]. These outputs indicate that the token ’communicative’ is grammatically incorrect.

CZECH Dataset GECCC : Grammar Error Correction Corpus for Czech (GECCC) consists of 83,058 sentences and covers four diverse domains, including essays written by native students, informal website texts, essays written by Romani ethnic minority children and teenagers and essays written by nonnative speakers. The statistics of the dataset are shown in the above figure.

Swedish MultiGED dataset:

The Swedish MultiGED dataset is based on the SweLL-gold corpus that contains essays written by adult learners of Swedish. The corpus is described in Volodina et al. (2019) The statistics of

Swedish data

Source corpus	Split	Proportion	Nr sentences	Nr tokens	Nr errors	Error rate
SweLL-gold	Total	1.0	8,553	145,525	27,274	0.187
	• train	0.791	6,729	115,203	21,615	0.188
	• dev	0.108	911	15,685	2970	0.189
	• test	0.101	913	14,619	2689	0.184

Figure 5: Swedish MultiGED dataset statistics

the dataset are shown in the above figure.

4.1 Data preprocessing

For all of the datasets : FCE, REALEC and MERLIN and their corresponding models, we have done pre-processing in the same fashion. Since the data in the datasets is already divided into tokens, we pass the dataset through a function which processes the given tokens and returns a dictionary of the sentences along with the labels of individual tokens for the dataset. This step is performed for the following reasons. Firstly, sequence-based models require fixed-length inputs, and joining sentences helps ensure consistent input sequence lengths. Secondly, joining the sentences preserves contextual information, enabling the model to capture dependencies between tokens effectively. Thirdly, it improves training efficiency by enabling techniques such as batching and parallel processing.

In addition to this, we are adding the labels i.e c or i(correct or incorrect) as feature to the dataset.

Next step of pre-processing which we have implemented are dependent on the models. For BERT model, we have enclosed the sentences with [CLS] and [SEP] tokens. By using the [CLS] and [SEP] tokens, the token classification model can effectively process and understand the relationship between tokens, capture the overall context of the input sequence, and make accurate predictions based on the specific task requirements. On the other hand GPT-2 model architecture, operates based on the Transformer architecture and employs a ”masked language modeling” objective. It predicts the next word in a sequence based on the context of previous words. Therefore, it does not require explicit classification or separation tokens like [CLS] and [SEP] as used in BERT. We included the <s>tag and task-specific prefixes, to provide explicit information to the T5 model about the intended task and help guide its behavior during training and inference.

The next step of the pre-processing of the

dataset is to tokenize and align the training data. Different tokenizers were used for the compatible model architectures. For example, GPT-2Fasttokenizer is used when we are training GPT-2 model, BertTokenzier for training of the BERT model etc. The following steps are performed: Mapping all tokens to their corresponding word with the wordids method is done. Assigning the label -100 to the special tokens [CLS] and [SEP](for BERT) so they're ignored by the PyTorch loss function.Only labeling the first token of a given word. Assign -100 to other subtokens from the same word. In addition to this, padding is done to ensure that all input sequences have the same length, allowing for efficient batch processing and parallelization. Additionally, padding retains the positional information of the tokens within the sequence, which is crucial for transformers to learn the relationships between different positions.

5 Baselines

5.1 Bi-Directional LSTM: For our sequence labeling experiments, we adopt the baseline architecture introduced by Rei et al. (2016), which utilizes a neural network model. This model operates on individual sentences that are tokenized and assigns labels to each token through the utilization of a bidirectional LSTM.

To capture the contextual information of individual words, the initial input tokens are converted into a sequential arrangement of distributed word embeddings : [x1, x2, x3, ..., xT]. To achieve this, two LSTM (Long Short-Term Memory) components, based on the architecture proposed by Hochreiter and Schmidhuber in 1997, traverse the sentence in opposing directions. These LSTM modules utilize the hidden state from the previous time step and the current word embedding as input, generating updated hidden states. By merging the hidden representations obtained from both forward and backward directions, we obtain context-specific representations for each word, effectively incorporating the entire sentence's context.

$$\overrightarrow{h_t} = LSTM\left(x_t, \overrightarrow{h_{t-1}}\right), \overleftarrow{h_t} = LSTM\left(x_t, \overleftarrow{h_{t+1}}\right)$$

$$\mathrm{h}_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$$

The concatenated representation is passed through a feedforward layer, mapping the components into a joint space and allowing the model to learn features based on both context directions:

	FCE DEV	FCE TEST		
	$F_{0.5}$	P	R	$F_{0.5}$
Baseline	48.78	55.38	25.34	44.56
+ dropout	48.68	54.11	23.33	42.65
+ LMcost	**53.17**	**58.88**	**28.92**	**48.48**

Figure 6: Baseline results

$$d_t = \tanh\left(W_d h_t\right)$$

where W_d is a weight matrix and $\tanh$ is used as the non-linear activation function.

In order to predict a label for each token, we use either a softmax or CRF output architecture. For softmax, the model directly predicts a normalised distribution over all possible labels for every word, conditioned on the vector d_t.

$$P\left(y_t \mid d_t\right) = softmax\left(W_o d_t\right) = \frac{e^{W_{o,k} d_t}}{\sum_{\tilde{k} \in K} e^{W_{o,\tilde{k}} d_t}}$$

The model is optimised by minimising categorical crossentropy, which is equivalent to minimising the negative log-probability of the correct labels:

$$E = -\sum_{t=1}^{T} \log\left(P\left(y_t \mid d_t\right)\right)$$

In addition to the aforementioned approach, we incorporate a character-level component based on Rei et al.'s (2016) methodology. This component constructs an alternative representation for each word by mapping individual characters to character embeddings and feeding them through a bidirectional LSTM. The final hidden states from both directions are concatenated and processed by a nonlinear layer. This resulting vector representation is then combined with a regular word embedding, utilizing a dynamic weighting mechanism that adjusts the balance between word-level and character-level features. This framework enables the model to learn patterns based on characters and effectively handle previously unseen words while leveraging the benefits of word embeddings.

The optimal hyperparameters chosen for the BiLSTM implementation were:

$$epochs = 200$$

$$learning\_rate = 1.0,$$

$$learningrate\_decay = 0.9,$$

$$max\_batch\_size = 32$$

The method used to choose the optimal hyperparamters was chosen using grid search on the above parameters with epochs range [3, 100, 200], learning rate [0.1, 0.5, 1] and max_batch_size [16, 32] and we ran the implementation on sequence labeler implementation for getting the baseline results. The train/validation/test split for the dataset is the same as the original FCE dataset split which is shown in the FCE dataset information in the Dataset section. We chose this baseline as the implementation for the sequence-labeler was readily available and was easy to fine-tune by simply changing the hyper-parameters in the config files of the codebase.

Distil-BERT:

We used another major base line which is the Distil-BERT for comparing the models we fine-tuned. Distil-BERT is a smaller, faster, cheaper and lighter version of the BERT model and is made readily accessible in the hugging face library. Distil-BERT is based on the idea of Knowledge distillation. Knowledge distillation is a compression technique where a smaller model, known as the student, is trained to replicate the behavior of a larger model or ensemble of models, referred to as the teacher. In supervised learning, the usual objective is to maximize the estimated probability of correct labels and minimize cross-entropy between the model's predicted distribution and the empirical distribution of training labels. A well-performing model predicts a high probability for the correct class and low probabilities for other classes. However, the varying "near-zero" probabilities indicate the model's generalization abilities and its expected performance on the test set.

In this work, the student model, named DistilBERT, follows a similar architecture to BERT but with some modifications. The token-type embeddings and pooler components are removed, and the number of layers is reduced by half. The linear layer and layer normalization operations in the Transformer architecture are optimized in modern linear algebra frameworks. It is observed that variations in the hidden size dimension have a smaller impact on computation efficiency compared to variations in the number of layers. Therefore, the focus is on reducing the number of layers. Additionally, the student model is initialized by taking every other layer from the teacher model, leveraging their shared dimensionality.

We tuned the DistilBERT model in the same way as that of BERT using (Mosbach et al., 2021) by selecting optimal hyperparameters of

$$train\_batch\_size = 4,$$

$$valid\_batch\_size = 2,$$

$$epochs = 1,$$

$$learning\_rate = 3e - 05$$

Again, we used GridSearch with the optimal ranges given in the paper (Mosbach et al., 2021) on the values of EPOCHS, LEARNING_RATE and BATCH_SIZE. We used this as a baseline as we are primarily focussing on achieving the best values of metrics for the BERT model and we considered that Distil-BERT could be a valid baseline as the BERT model could achieve comparatively better performance. Also, we were easily able to access the DistilBERT model from the hugging face library which made it easy to evaluate. The train/validation/test split for the dataset is the same as the original FCE dataset split which is shown in the FCE dataset information in the Dataset section.

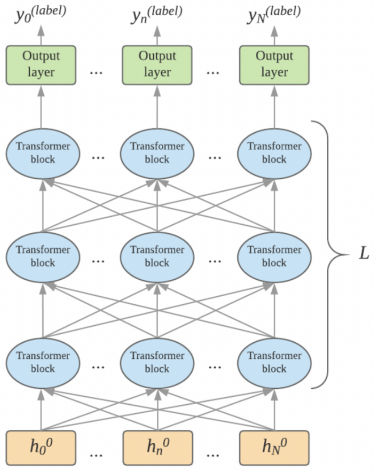
6 Approach

We used several pre-trained models to fine-tune on the task of token level predictions for Grammar Error detection in a multi-lingual setup. We primarily leveraged the pre-trained models from the **Hugging Face library**. We employed several models like BERT(encoder only), GPT-2(Open-GPT model)-decoder only, RoBERTa, T5(encoder-decoder model) from the hugging face library and fine-tuned each of these models separately in separate Collab python notebooks. We elaborated on the fine-tuning process of each of the models in the following sections.

BERT model: BERTs are used to understand bi-directional information by understanding the contexts from both sides. Since it is known that BERTs achieved outstanding results on several tasks such as Named Entity Recognition, question answering, etc, we plan to use the same sequence labeling task. Considering a sequence of words $w_0, w_1, w_3, ..., w_n, ..., w_N$, Bert uses multiple layers of transformer_blocks for predicting the output as follows:

$$h_n^0 = W_e w_n + W_p$$
$$h_n^l = transormer\_block(h_0^{l-1}, h_1^{l-1}, ..., h_n^{l-1})$$
$$y^{(BERT)} = softmax(W_o h_n^L + b_o)$$

Each transformer_block includes self-attention and fully connected layers which takes all the output values from previous transformer_blocks $h_0^{l-1}, h_1^{l-1}, ...h_{l-1}^N$ as inputs and generates output h_n^l . In the above formula L denotes the number of layers, b_o is the bias for the output layer, $y_n^{(BERT)}$ is the predicted label.



(a) BERT.

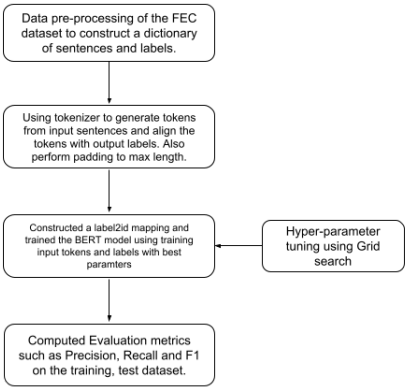
The pipeline consisted of fine-tuning BERT model, bert-base-multilingual-cased model (104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters) for the task of token-level-classification of Grammar Error detection. The model has been fine-tuned by selecting the optimal hyperparameters with MultiGED-2023 dataset. We followed the (Mosbach et al., 2021) for optimal hyper-parameter selection for the BERT model and fine-tuned it further for our downstream task. For fine-tuning we had to perform several steps in the pipeline. The pipeline is shown in the Figure 7.

Tokenization in BERT :

BERT uses sub-word tokenizer and this tokenizer also adds special tokens such as the [CLS] and [SEP] token. Input : ['@paulwalk', 'is', 'living', 'in', 'Empire', 'State', 'Building', '=', 'ESB', '.']

Output: ['[CLS]', '@', 'paul', 'walk', 'is', 'living', 'in', 'empire', 'state', 'building', '=', 'es', 'b', '.', '[SEP]'] We also append each of the tokenized sentences with padding([PAD]) tokens to generate each of the tokenized sentences with a max length of pre-defined parameter MAX-LEN.

Figure 7: Pipeline flow for different models



We use the DataCollatorForTokenClassification and it's more efficient to dynamically pad the sentences to the longest length in a batch during collation, instead of padding the whole dataset to the maximum length.

After the tokens for each sentence are obtained, we follow the tokenize_and_align_labels methodology described in Data Pre-Processing step. While aligning the labels and tokens, the token_id's of the special tokens set to -100 so that the predictions for the special tokens will be ignored by the Loss function in computing the final metrics.

We faced difficulty in training the pre-trained BERT model for epochs beyond 5 on Collab notebook. We found this training process was computationally expensive and utilized resources for BERT in comparison to GPT-2, RoBERTa. We set the following **hyper-parameters** for fine-tuning BERT using GridSearch, MAX_LEN = 128, TRAIN_BATCH_SIZE = 4, VALID_BATCH_SIZE = 2, EPOCHS = 1, LEARNING_RATE = 3e-05

GPT-2 and RoBERTa:

GPT2ForTokenClassification:

For evaluating the performance of models with BERT, we fine-tuned several other models only on the English dataset and evaluated the performance on FCE and REALEC datasets. We considered the Open AI GPT-2 model. The original GPT-2 model has the parameters ranging from 117 million to 1.5 billion. However, we are using a compact pre-trained model shown below to fine-tune for the downstream task. The overall pipeline for this

model remained the same as Figure 7. We used the pre-trained model GPT2ForTokenClassification. from_pretrained("brad1141/gpt2-finetuned-comp2") And fine-tuned using the FCE dataset.

There is a difference in the tokenizer output that GPT-2 model from the BERT tokenized output. Sample output from the GPT2 tokenizer can be seen in the below example:

Output: ['Ġ@', 'p', 'aul', 'walk', 'Ġis', 'Ġliving', 'Ġin', 'ĠEmpire', 'ĠState', 'ĠBuilding', 'Ġ=', 'ĠES', 'B', 'Ġ.']

The "Ġ" character is specific to the GPT-2 tokenizer and is a convention used by the Hugging Face Transformers library. It helps to distinguish between word boundaries when a word is broken down into subword units during tokenization.

RobertaForTokenClassification:

RoBERTa tokenizer, derived from the GPT-2 tokenizer, uses byte-level Byte-Pair-Encoding. Used the following model for fine-tuning on token level classification task. RobertaForTokenClassification. from_pretrained("Jean-Baptiste/roberta-large-ner-english")

Output: ['<s>', 'Ġ@', 'p', 'aul', 'walk', 'Ġis', 'Ġliving', 'Ġin', 'ĠEmpire', 'ĠState', 'ĠBuilding', 'Ġ=', 'ĠES', 'B', 'Ġ.', '<s>']

We observed that both the above models trained seamlessly for 7 epochs. We set the following **hyperparameters** for fine-tuning both GPT-2 and RoBERTa using GridSearch, learning_rate=2e-5, per_device_train_batch_size=16, per_device_eval_batch_size=16, num_train_epochs=2, weight_decay=0.01.

T5:

T5 is based on the Transformer architecture, which is a deep learning model known for its success in natural language processing (NLP) tasks. The T5 model is pretrained on 34 billion tokens from the C4 corpus. For fine-tuning T5 on specific tasks, including ours, a different process is applied as shown in Figure 8.

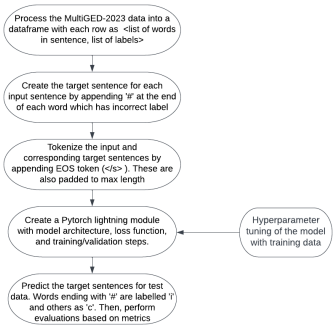
The following example showcases the difference between actual vs predicted sentences for a sentence in FCE test data.

Actual Entities: i was impressed when i heard that she liked playing# puzzle# alone .

Predicted Entities: i was impressed when i heard that she liked playing puzzle# alone .

We faced challenges using larger models like T5-base or MT5 due to the limited compute resources. Instead, we had to utilize the T5-small model, which has fewer parameters and lower computational requirements. However, even with its smaller size, the T5-small model demonstrated the ability to understand grammar-related patterns.

Figure 8: Fine-tuning for T5



6.1 Results:

Table 1: FCE dataset

Model	P	R	F-0.5
Baseline	0.82	0.61	0.71
BERT	**0.844**	**0.687**	**0.786**
GPT-2	0.823	0.620	0.724
T5	0.842	0.569	0.651
RoBERTa	0.504	0.506	0.49

Table 2: REALEC dataset

Model	P	R	F-0.5
BERT	**0.807**	**0.64**	**0.743**
GPT-2	0.675	0.619	0.639
T5	0.69	0.572	0.632
RoBERTa	0.503	0.51	0.483

Table 3: Evaluation Metrics for multi-lingual BERT

Dataset	P	R	F-0.5
MERLIN	0.873	0.587	0.70
SweLL-gold	0.541	0.4044	0.467
GECCC	0.873	0.807	0.851

Inference from Results: All the above results are from the test-data present in MultiGED-2023 dataset. Some of the languages has unlabeled test-data and this data had to be evaluated on the Official Data evaluation platform for the

Figure 9: Training and Validation loss vs. Epochs for BERT

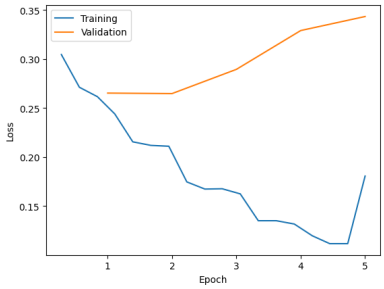
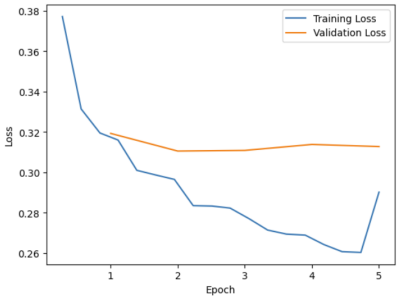


Figure 10: Training and Validation loss vs. Epochs for GPT-2



MultiGED-23 shared task The Baseline considered in the Table 1 is Distil-BERT. Our models were found to be performing exceedingly well on BiLSTM and Distil-BERT baselines on the metrics of Precision(P), Recall(R) and F-0.5 score. Being an encoder only model, BERT is found to perform better than the traditional baseline models that compared viz. Bi-directional LSTM, Distil-BERT. This could be explained due to various reasons. 1. Attention Mechanism: BERT utilizes the self-attention mechanism, which enables the model to attend to different parts of the input sequence when making predictions. This attention mechanism allows BERT to capture long-range dependencies and understand the relationships between words, even if they are far apart. 2. Large-Scale Pre-training: BERT is pre-trained on massive amounts of text data, allowing it to learn general language representations. 3. Fine-Tuning: After pre-training, BERT can be fine-tuned on task-specific labeled data. Fine-tuning allows BERT to adapt its learned representations to the specific token-level classification task at hand, further improving its performance. This fine-tuning process helps BERT to leverage its pre-trained knowledge effectively and make better predictions.

It was surprising that generative models like GPT-2 and T-5 also showed comparable performance with BERT in beating the Bi-LSTM baseline. It is evident that GPT and T-5's strength lies in generating coherent and contextually relevant text. While it can be fine-tuned for token-level classification, it may not perform as well as BERT since it lacks the bidirectional context modeling that BERT offers. Overall we found our models to perform better on examples which the baseline failed to perform well.

We relied on help from existing implementations present in the Token classification implementation in Hugging Face library . We tried training on the FCE dataset with increased number of epochs, but due to resource constraints on Collab bigger models like BERT gave out of memory issue after 4-5 epochs and GPT-2, T-5, RoBERTa ran reasonably well until 7 epochs.

7 Error analysis

We have collected set of 140 examples from the FCE test data where the Baseline model fails i.e DistilBERT model incorrectly assigns labels for some of the words in each of these sentences. This is better explained by one of the following collected examples.

Token	True Label	Predicted Label
It	c	i
takes	c	i
25	c	c
minutes	c	i
that	i	i
is	c	c
convenient	c	i
to	i	i
us	c	c
.	c	c

We have observed the following common patterns in the collected errant sentences.

- Sentence structure: Many sentences lack proper grammar and sentence structure. There are instances of run-on sentences, missing verbs, and incorrect word order.

  Example: About the party in the end of conference we organise in the same hotel whict have a nice salon .

- Informal language: The examples often contain informal language and a conversational

tone, with the use of contractions, abbreviations, and incomplete sentences.

Example: The Hotel I was be booked is Palace Hotel wiche is in the center of London just tow blokes from Victoria stations .

- Lack of coherence: The examples lack coherence and organization, with ideas and information presented in a disjointed manner.

  Example: The maching was used for read this cad was too big you use to need a room for our Computer .

The patterns that were not captured by the baseline model could be due to the model's reliance on the patterns learned from the training data, which may not have adequately represented these patterns. To overcome this, a more diverse and balanced dataset comprising of different language styles, sentence structures, and coherent text could be used. Additionally, the small size of the pretrained corpus in the DistilBERT model could have also affected its ability to capture these patterns.

8 Contributions of group members

- Vishnu Sabbavarapu: collected the data and preprocessed for the baseline model (DistlBERT, BiLSTM). Did research on fetching the baseline model and its metrics. Finetuned and Evaluated the models
- Varun Bachireddy: collected the data and preprocessed for BERT and GPT2. And did research on fetching the models. Did finetuning and evaluation on them.
- Abhinav Reddy Yadatha: collected the data and preprocessed for T5. And did research on fetching the models. Did finetuning and evaluation on them.
- Harsha Kanaka Eswar Gudipudi: collected the data and preprocessed for Roberta. And did research on fetching the models. Did finetuning and evaluation on them.

We collaborated as a team to collectively work on the report and carried out the error analysis.

9 Conclusion

Our study on Multilingual Grammatical Error Detection, inspired by the MultiGED-2023 shared task, involved investigating and comparing the performance of various advanced Language Models. However, the process was challenging due to the large size of the models and limited computing resources, making fine-tuning a tedious task.

In our analysis, we found that models like BERT and GPT2 surpassed our Baseline models, achieving F0.5 scores of 0.78 and 0.73, respectively. Interestingly, BERT outperformed GPT2, which could be attributed to its ability to capture both left and right context effectively, encode contextual information in token representations, and adapt its knowledge through fine-tuning. In contrast, GPT2 relies on the preceding context for computing embeddings.

For our future work, we aim to address a few aspects. Firstly, we intend to expand and diversify our dataset, ensuring it covers a broader range of challenging grammatical patterns. Additionally, we plan to explore fine-tuning other models using multilingual data, which holds promising potential for further improvement in performance.

10 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes, ChatGPT

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - Related work: Understand the writing style of reference Related work as given below and rewrite the given data in the same style.
  - Data section: rephrase 'The Russian Error-Annotated Learner English Corpus (REALEC) is an open access corpus available from HSE university. It contains approximately 18,700 texts written by HSE students in the Independent English Language Test between 2014-2020 in response to two types of tasks (approximately 4,336,000 words).'

  - Data section : Describe the statistics of this dataset(MERLIN) and pasted the table with the values.
  - Data section : Describe the statistics of this dataset(FCE) and pasted the table with the values.
  - Data section : Describe the statistics of this dataset(GECCC) and pasted the table with the values.
  - BERT vs GPT-2 for token level classification tasks
  - DistilBERT vs BERT performance
  - what does special character Ġ represent in tokens output by GPT2Tokenizer
  - for token-level-classification task BERT is performing better compared to Bidirectional LSTM. explain why?
  - Error Analysis: How can we capture grammar w.r.t informal use?
  - Conclusion: Does increasing compute resources be useful to carry out the work on grammatical detection?

- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  - your response here

References

Alikaniotis, D. and Raheja, V. (2019). The unreasonable effectiveness of transformer language models in grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 127–133, Florence, Italy. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2018). Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Mosbach, M., Andriushchenko, M., and Klakow, D. (2021). On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., and Bryant, C. (2014). The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Rei, M. (2017). Semi-supervised multitask learning for sequence labeling.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Yuan, Z. and Felice, M. (2013). Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.