ELL884 - Deep Learning in NLP

Assignment 3

Grammar Error Correction

Name: Somesh Agrawal
Roll No. : 2024AIB2292

Grammar Error Correction

Grammar Error Correction (GEC) is the task of automatically detecting and fixing grammatical mistakes in text. The goal is to take an incorrect sentence (like *"She go to school yesterday."*) and correct it to a grammatically correct one (like *"She went to school yesterday."*).

GEC is useful in applications like grammar checkers, writing assistants, and language learning tools. It typically uses machine learning models — especially encoder-decoder models like **T5** or **BART** — trained on large datasets of incorrect and correct sentence pairs to learn how to make such corrections automatically.

Parameter Efficient FIne Tuning:
It is a way to fine-tune big language models (like T5 or BART) without changing all of their parameters. Instead, we only update a small part of the model, which makes training faster, uses less memory, and needs less data.

Lora(Low Rank Adaptation)
The **LoRA algorithm** is used to **fine-tune large models efficiently**. Instead of updating the full weight matrices during training, LoRA:

1. **Freezes** the original model weights (keeps them unchanged)
2. **Adds two small matrices** (`A` and `B`) to each large weight matrix we want to adapt.
    - These small matrices have **much fewer parameters**.
    - Their product (B × A) is added to the original weights during training.

This way, the model learns **only the added low-rank updates**, not the full weight matrix.

Using these optimization methods has resulted in higher accuracy and better scores while reducing training weights. For this assignment, I trained both a T5-Base Model and a BART-Large Model for error correction techniques.

The original models presented challenges: larger batch sizes created memory issues, while smaller batch sizes significantly increased training time. To address these limitations, I implemented LoRA (Low-Rank Adaptation) for fine-tuning. The final report covers the two models trained:

1. BART-Large
2. T5-Base

BART-Large

BART-Large is a powerful language model built with an encoder-decoder structure. It works like a smart text editor that can understand, fix, or rewrite sentences. BART is trained by corrupting text (like removing or shuffling words) and then learning to restore the original version. This helps it become very good at tasks like grammar correction, summarization, and text generation. The “Large” version simply means it has more layers and capacity, making it better at handling complex language problems.

T5-Base

T5-Base is a text-to-text transformer model that treats every language task as a problem of converting one piece of text into another. Whether it’s translation, question answering, or grammar correction, the task is given as part of the input (e.g., “Translate English to French: Hello”). It also uses an encoder-decoder setup like BART. T5 is very flexible and can handle a wide range of tasks by simply changing the input prompt. The “Base” version is the standard-sized model—smaller than T5-Large, but still powerful and efficient.

A lot of experiments have been done on the T5 model since it was easy to train and took less time allowing a lot of experiments to be done on the model.

Results:

T5- Base Model

Predefined Configurations Used in the code:

Batch Size:16

Epochs: 7-10

Learning Rate: 0.00005

Pretrained Tokenizers has been used for the Bart and T5.

LoRA was implemented for training purposes as the original model was time-consuming and caused memory issues with larger batch sizes. Specifically, the Query and Value matrices were adapted using LoRA techniques.

For inference, I employed beam search with a beam size of 10 units to optimize output quality. The final reports reflect experiments across multiple hyperparameters, including various learning rates, optimizer configurations, epoch counts, and dropout rates across different model architectures.

For inference, we have run experiments on the beam search.

Training Experiments:

Model	**Training Set Bleu Score**	**Training Set Exact Match(in%)**	**Validation Set Bleu Score**	**Training Set Exact Match(in%)**
Base Model	79.03	19.37	80.05	19.42
Base Model + lr= 5e-5	75.01	11.02	76.26	10.68
Base Model + lr= 10e-5 and epochs=10	77.58	16.09	77.89	13.59
Base Model + lr = 3e-4 + epochs=13 + batch size:32	79.51	20.45	80.12	18.45

With Adam Optimizer and Cosine Scheduler and Learning Rate: 3e-4

Epoch: 5	80.71	23.07	81.65	22.33
Epoch: 15	81.08	23.58	82.11	26.21
lr: 5e-4 Epoch: 13	82.02	26.19	82.77	27.18
Adding Weight Decay with lr: 5e-4 Epoch: 13	79.57	20.71	80.10	19.42

Using Adam Torch Fused lr: 3e-4 Epoch: 13	81.23	24.53	81.54	22.34

Validation

Here the above base model has been tweaked to make the following changes with learning rate to be 0.00005 and number of epochs has been defined to be 10.

Model	Training Set Bleu Score	Training Set Exact Match(in%)	Validation Set Bleu Score	Training Set Exact Match(in%)
Beam Size: 5	77.58	16.09	77.89	13.59
Beam Size : 10	79.03	19.37	80.05	19.42
Beam Size: 15	80.71	23.07	81.65	22.33

For BART Model

Batch Size:32

Epochs: 7

Learning Rate: 0.0003

Model	**Training Set Bleu Score**	**Training Set Exact Match(in%)**	**Validation Set Bleu Score**	**Training Set Exact Match(in%)**
Base Model	76.28	14.37	77.32	14.62
Base Model + lr= 5e-5	77.23	16.43	77.54	15.48

Conclusions:

BART-Large performance was lower than the performance of the T5-Base Model. A large number of models were not trained for BART as it was taking a lot of time training a heavy model like BART.

With the increase in Beam Size, the BLEU Score and Exact Match Accuracy increased for validation and training sets even for this specific task. However, the issue with increasing beam size is the corresponding increase in RAM memory consumption.

Some models didn't show much change in their BLEU Score and Exact Match Accuracy when increasing the beam size.

With increasing epochs, the training loss and validation loss decreased, but depending upon the case, a plateau was reached for validation and training, and further significant improvements couldn't be observed.

Exact Match accuracy was very low and potentially misleading due to changes in word ordering that shifted the relative positions of words, hampering the final model accuracy. However, BLEU score took care of the semantics, and hence much change wasn't observed, since semantics don't change due to grammatical errors.

With increases in learning rates, the model trained faster, but with higher learning rates such as 0.0005, we observed jumping loss values due to these high learning rates. Conversely, low learning rates in the range of 10^-5 weren't helpful as they significantly increased model training time.

Higher epochs led to model overtraining, while lower numbers of epochs didn't allow the model to adequately learn from the data.

Adding weight decay didn't help much in training for decreasing the training and validation loss.

Final Model:

T5 Model has been shared, which has been trained for 13 epochs, and beam size 15, keeping learning rate 3e-4. Adam Optimizer and Cosine Scheduler has been used while training the model.

Some Outputs:

Positive Outputs:

Given Input	Corrected Output
I look forward hearing from you.	I look forward to hearing from you .
""" Sagrada Familia "" a real holly building ."	""" Sagrada Familia "" , a real holy building ."
It has a interesting past .	It has an interesting past .

Negative Outputs

Given Input	Corrected Output	Actual Output
The entery will be free .	The entrance will be free .	The entry will be free.
THE HYSTORICAL MUSEUM	THE HYSTORICAL MUSEUM	THE HISTORICAL MUSEUM

References:

1. https://blog.stackademic.com/fine-tuning-t5-for-grammar-correction-a-step-by-step-guide-edba96ada787
2. https://www.kaggle.com/code/aisuko/fine-tuning-t5-small-with-lora