University of Science and Technology of Hanoi

Minor project

Grammar Autocorrector

Subject: Natural Language Processing

Group 13’s members:

Vũ Đức Duy	**22BI13127**
Nguyễn Hoài Anh	22BI13021
Nguyễn Việt Anh	22BI13032
Phạm Hoàng Anh	22BI13034
Nguyễn Át	22BI13047
Nguyễn Hoàng Lân	22BI13242

Lecturer: Pham Quang Nhat Minh

Hanoi, February 2025

Table of Contents

I. Introduction

Effective communication demands lots of clean and precise writing. It would be tiresome to read the text with the smallest grammatical error. Providing the solution to this, auto grammatical check is among many solutions, identifying the errors and also providing recommendations, making the writing easier to read and clearer than originally.

The program's job is to edit standard errors in composition, including the use of incorrect verb tenses, typographical errors, clumpy sentences, and inexact words. The program, starting from a sentence in the erroneous version, rephrases it into the accurate version, through techniques.

Grammar autocorrects are easily accessible in day-to-day applications such as word processing, emailing, and web browsing. The objective in this project is to develop such a program, how and the reasons why it runs, and the reasons the world needs it to support improved written communication.

II. Theoretical Background

1. C4 200M dataset

The C4 200M dataset is a large dataset designed for Grammar Error Correction (GEC) tasks. It contains 185 million sentence pairs, where each pair includes both incorrect sentences and its corrected version. This dataset was created by using a Tagged Corruption model on Google's C4 dataset. It is converted in Parquet format, but is available in TSV format and divided into 10 files of approximately 18 million samples each.

This dataset is most useful for training sequence-to-sequence or in short is seq2seq models for grammatical error correction. The corrections were created by Felix Stahlberg and Shankar Kumar and are licensed under CC BY 4.0.

2. Fine-Tuning T5 Model

T5 fine-tuning or Text-to-Text Transfer Transformer fine-tuning refers to the procedure of making a pre-trained T5 model more specific to a task by training it on a labeled dataset. T5 follows a Text-to-Text framework, i.e., input and output are text, and generalizes it to a variety of Natural Language Processing tasks like summarization, translation, and question answering.

3. Encoder-Decoder architecture

Encoder-Decoder is a powerful neural network model that is being widely used to address sequence-to-sequence (seq-2-seq) problems such as machine translation and grammar correction. It consists of two main parts: the decoder and the encoder.

1, The role of the Encoder: Take the input sequence and convert it into a fixed-length context vector. This Vector holds the vital information and structure of the input.

2, The role of the Decoder: Uses the encoder context vector as the input and then generates the output sequence. It generates each token based on those that came prior to it.

III. Methodology

In this section, we develop a grammar autocorrector model using encoder-decoder architecture.

1. Implement

A step-by-step implementation process is crucial to develope a grammar autocorrector model. This section outlines the key steps from setting up the environment to training and fine-tuning the model. The implementation consists of several essential stages: Environment Setup, Seed and Device Setup, Data Preprocessing, Training and Fine-tuning model. Each of these steps plays an important role in building a grammar auto-correction system. The following subsections provide a detailed explanation of each stage.

1.1. Environment Setup

Before running the code in the notebook, ensure having a perfect environment for it. If not, set up using one of two ways:

1, Using a Virtual Environment: Create a virtual environment to manage libraries. Then install all the required libraries in the “requirements.txt” file.

2, Using Google Colab: Ensure all the required libraries are installed.

If using Google Colab for the notebook, continue to connect with the Google Drive to access the data file.

1.2. Seed and Device Setup

Using a fixed random seed and computing device will increase the result and the model training efficiency.

1, Select Seed: Using 42 for seed to ensure the reproducibility of the result for the machine learning model.

2, Select Computing Device: Setting device = torch.device(“cuda”) will set the device to a GPU if available because GPU helps accelerate deep learning computations better than CPU.

1.3. Data Preprocessing

1, Download and Extract the Dataset:

- Download the Dataset: Download the dataset from Kaggle using the code below: **kagglehub.dataset_download(“dariociono/c4200m”)** and store it in a path. After that, create a folder in Google Drive to store the downloaded files: **/content/drive/My Drive/Folder’s Name**. Ensure that the folder exists for the downloaded files. Then move all the dataset from their location to a specific Google Drive directory. The dataset downloaded is in zip format.

- Extract the Dataset: After downloading the dataset, locate the folder, which stores datasets, and create a new folder in Colab, which stores datasets after extract. Then scans the dataset folder in Google Drive for any files ending with .zip, iterates them and extracts them. Move the successful extraction file to the new folder for use.

2, Merge Dataframe:

Get Rows for each CSV file: Set the number of rows to read from each CSV file (In this case we set the number of rows equal to 10000).

- Read Data from each CSV file: Loop through each CSV file in a data file, then read each file while the number of rows is still correct. Rename columns to “Input” and “Target”. Then add columns to a dataframe.
- Merge Data: Using pd.concat() to merge all dataframes into a single dataframe for training.

1.4. Training and Fine-tuning Model

Before Training Model, from sklearn.model_selection, import train_test_split library. Then divide the dataframe into two parts: training data-contains 70% of the dataset, temp data-contains 30% remaining and split temp data to valid and test data, both of them have 15% of temp data.

1, Create Tokenizer: Using AutoTokenizer class to automatically select the matching tokenizer for the T5-base model, which is from Hugging Face’s Transformers library. This tokenizer converts text into tokenized representation like this:

Input IDs length: 44, Attention Mask length: 44, Labels length: 42

so that the model can understand and process.

2, Encoding: Encoding is necessary before training the model. Make a loop to run through each row in the dataframe, then extract the “Input” and “Target” columns. Both the text inputs and the corresponding target output are tokenized with padding and truncation to fix the lengths. At the end of the loop, processed will return a list that contains the data inside.

3, Training the Model: Using transformers library:

- Select Model: Import the AutoModelForSeq2SeqLM to select the model: t5-base, a Text-to-Text Transfer Transformer (T5) model.
- Data Collector: Import the DataCollatorForSeq2Seq to efficiently batch and preprocess the data for training model. The “t5-base” which is set as the model to reference, with the tokenizer ensuring consistent tokenization. Set the return tensors to “pt” to transform data into Pytorch tensors. And set padding to “longest” to ensure all sequences in the batch align with the longest one.
- Training Parameter: Import the Seq2SeqTrainingArguments to support the T5 model’s fine-tuning process. Assign the checkpoint and trained model storage location. Next, set parameters like: the use of component eval_strategy to carry out evaluation at the end of each epoch/period, num_train_epochs, learning_rate for balance and stability, v.v. While the model is training, these parameters support the model’s optimal operation.
- Training the Model: Import the Seq2SeqTrainer to train the model. The ROUGE metric is called beforehand via evaluate.load(“rouge”) because of evaluating text generation quality. The tokenized train and validation datasets, training arguments, and model to be trained are the key components of a model training process. The tokenizer and data_collator to ensure input formatting and compute_metrics to evaluation criteria. Finally, using function train() to start the training process.
- Evaluation Model: Evaluate the performance of the model with the ROUGE scores. First, decode the model’s predictions with the tokenizer and remove special tokens. After that,

replace -100 values with the ID of the padding token in the tokenizer before decoding. The ROUGE measurement computes with the help of stemming in order to refine precision in comparison.

2. Tools and Libraries

For this project, we use Google Colab to develop, train and evaluate NLP models. Here is the libraries we use:

*** Python Libraries**

Library	Description
os	Interact with the operating system (e.g. path, files)
random	Generates random numbers or shuffle data
re	Provides expressions for text processing
zipfile	Handles creating, extracting, and reading ZIP files
shutil	Provides file operations like copying, moving, and deleting files or directories

*** NLP Libraries**

Library	Description
nltk	Natural Language Toolkit (e.g tokenization)
contractions	Expands contractions in text (e.g. can’t → cannot)

*** NLP Models**

Library	Description
transformers	Provides pretrained NLP models and tools.
AutoModelForSeq2SeqLM	Expands contractions in text (e.g. can’t → cannot)
AutoTokenizer	Tokenize text for NLP models
DataCollatorForSeq2Seq	Prepare data for sequence-to-sequence models
Seq2SeqTrainingArguments	Defines training arguments for sequence-to-sequence models
Seq2SeqTrainer	Handles training for sequence-to-sequence models

*** Data Handling and Processing**

Library	Description
pandas	Data manipulation and analysis with DataFrames
numpy	Numerical computation and array operations

*** Machine Learning**

Library	Description
torch	Core PyTorch library for building and training neural networks
sklearn.model_selection.train_test_split	Splits datasets into training and testing sets

*** Evaluation and Metrics**

Library	Description
evaluate	Metrics library for Machine Learning, including NLP
rouge_score	Calculate ROUGE metrics for summarization evaluation
rouge	Alternative library for ROUGE metrics

*** Other libraries**

Library	Description
tqdm	Displays progress bars
google.colab.drive	Connect Google Drive in Google Colab for file access

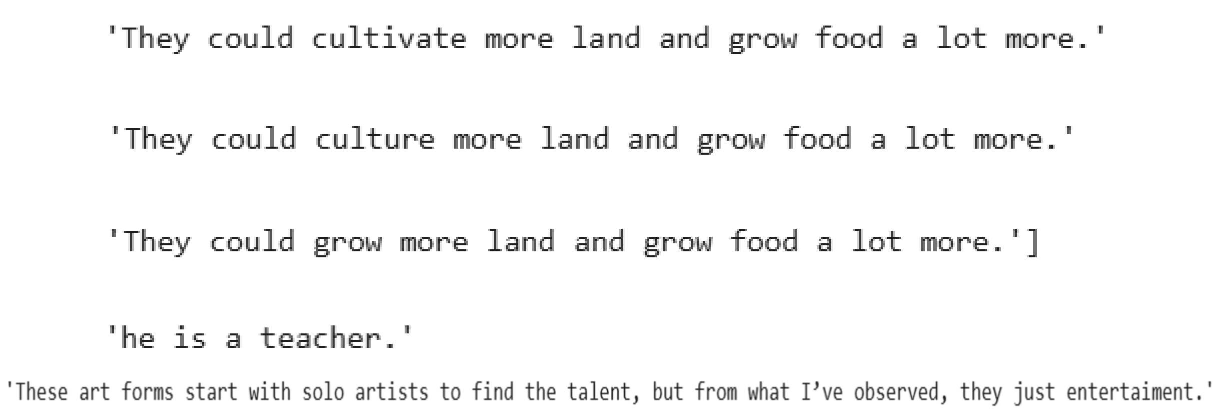
IV. Result

Our project has tested 2 simple cases like normal text (including short text and long text) and abnormal text (including punctuation and abbreviation errors), and also the key functions are carefully implemented while still keeping the system functioning without errors.

1. Testing with Normal text

In the first case, we tested some easy sentences for the short text such as `"They could culture more land and grows food a lot more.", "He are an teachers", "These art forms start with sologans to find the talent, but from what I've observed, they just entertaiment."`

These sentences are incorrect in their logic, vocabulary and grammar. After our system detects errors, it leads to the possible correct sentences



```
'They could cultivate more land and grow food a lot more.'

'They could culture more land and grow food a lot more.'

'They could grow more land and grow food a lot more.']

'he is a teacher.'
'These art forms start with solo artists to find the talent, but from what I've observed, they just entertaiment.'
```

```
'These art forms start with solo artists to find the talent, but from what I’ve observed, they just entertain.'
```

In the second case, there are sentences whose structures are redundant making the paragraph unclear and wrong in logic. Based on those sentences, our system has divided the section into small ones and then processed each. After that, we summarize into a complete paragraph. Here is a sample we have tested so far:

```
"""Today gift shows are popular in many countries, and purpose of these shows
finds talented people, and help them to introduce themselves to each other
.Actually, many people now watch this shows, and during this years find more
fans that cause increase the Viewer, and many sponsors Keen on for sponsoring
this shows, because gift shows has benefits for them, and this programs
convert to tools that earn money, and present their services.

Firstly, result this programme has  a massive effect on the society, because
many people get a chance to represent their gift. On the other hand, many
people have gift, but they do not know, so they have the opportunity to find
their gift, and encourage them to follow their interests.

secondly, many audiences, and viewers watch this shows, so it is a big chance
for companies by sponsoring in this program. They can find new customers and
introduce their services to each other.For instance, they commercials between
the shows certify this issue.Furthermore TV is one of the tools that
entertain people, although the target finds gift, so part of this shows for
entertaining people.

As a result, the aim of  producing this shows impressive, so part of the
society following this shows for entertaining, and the part of the people
persuade to find their talents. In fact, this topic has two side that
everyone can according to own opinion.

"""
```

Here is the correctness after processing:

```
“Today gift shows are popular in many countries, and purpose of these shows
finds talented people, and helps them to introduce themselves to each
other.Actually, many people now watch this shows, and during this years find
more fans that cause increase the Viewer, and many sponsors Keen on for
sponsoring this shows, because gift shows have benefits for them, and this
programs convert to tools that earn money, and present their services.
because many people get a chance to represent their gift. On the other hand,
many people have gift, but they do not know, so they have the opportunity to
find their gift, and encourage them to follow their interests. secondly, many
audiences, and viewers watch this shows, so it is a big chance for companies
to sponsor in this program. They can find new customers and introduce their
services to each other.For instance, the commercials between the shows
certify this issue. tools that entertain people, although the target finds
gift, so part of this shows for entertaining people. As a result, the aim of
producing this shows is impressive, so part of the society following this
shows for entertaining, and the part of the people persuade to find their
talents. In fact, this topic has two sides that everyone can according to
their own opinion.”
```

=> Based on the two cases presented above, we can see that the accuracy of our grammar autocorrect model is determined by sentence structure. The model, while capable of accurately correcting short and basic sentences (Case 1) (there is a chance that some sentences may lose their original meaning or consistency), it has some challenges when correcting long and complicated sentences (Case 2) and frequently fails to correct errors.

2. Testing with Abnormal text

Abnormal text refers to sentences that contain grammatical inconsistencies, logical errors, or unusual structures that make them difficult to understand. These errors often occur due to misspellings, missing words, or incorrect word order

First, we will load the library of contractions. The library condenses words and makes sentences shorter. But words in a contraction are made longer in a weird way because text is not properly processed and structure is ambiguous. Weird text could contain repeated words multiple times, poor punctuation, and loss of intent because automated repairs are made. We'll need to locate and repair these issues in order to maintain grammar and better how automated repairs are made.

We have tested a sample:

```
"""I can't believe it's already December,time flies so fast! I haven't seen
him since last year, he probably won't come to the party.Btw,Do you think she
is going to make it? I don't know, but she's been really busy lately, so
maybe she won't."""
```

Our system has checked and implemented all the issues which leaded to the correct version:

```
I cannot believe it is already december, time flies so fast! I have not seen
him since last year,  he probably will not come to the party. By the way, do
you think she is going to make it? I do not know,  but she has been really
busy lately,so may be she will not.
```

=> In summary, our system addressed how issues such as missing punctuation, and incorrect word usage can affect sentence clarity. Through testing, we observed that the autocorrection system successfully fixed most of the errors, improving grammatical accuracy. However, there is a small lack of maintaining the original meaning of the text. Continuous improvements in detection and correction mechanisms will enhance the system's effectiveness in handling abnormal text.

V. Conclusion and Future work

1. Conclusion

In this project, we developed a grammar autocorrect system using the T5 model, based on the C4 200M dataset. Our system was designed to address common grammatical errors like incorrect vocabulary, typographical errors, and sentence structure issues. By utilizing the encoder-decoder architecture, we were able to create a model that processes input text and generates corrected output, improving the overall grammatical accuracy of the text and paragraph.

The results of our testing demonstrated that the model performs well on short and simple sentences, effectively correcting grammatical errors but still has a chance to misinterpret the original meaning, when dealing with longer and more complex sentences, the model still has many challenges in the logical flow and consistency of the text. In addition, the system successfully corrected a large number of grammatical errors, particularly in cases involving punctuation, contractions, and basic sentence structure.

In summary, the effort emphasizes the potential with the use of pre-trained models, such as T5, in the area of grammar-related tasks. The system creates a foundation for future refinement, particularly in dealing with complex grammatical constructs without loss of the original message.

2. Future work

Given the shortcomings currently being faced, future efforts are intended to rectify these shortcomings and boost the operational efficiency of the system.

Real-Time Correction: The system currently functions in a batch mode. It is possible future research could explore the construction of a real-time grammar correcting tool incorporated in word processors, email clients, and Web pages such that users could receive instantaneous responses while they are typing.

Multilingual Support: The model is currently mostly focused on correcting English grammar. The system could in the future accommodate other languages and turn the system into a tool with a broader utility worldwide.

Addressing Complicated Sentencing: The model struggles with complex and long sentences with a lack of logical consistency. Subsequent efforts could strengthen the model in dealing with even complex sentence constructions, maybe with the addition of complementary information or with the implementation of more advanced models like GPT and BERT.

Degree of Precision: Users have a variety of stylistic preferences and conventional usage tendencies. In future versions, the system could accommodate users' desired degree of precision with a feature allowing a strict following of conventional usage and a loser, pragmatically-driven adjustment.

VI. References and Resource

[1] LeewanHung. “T5_Grammar.” Kaggle, 2023. https://www.kaggle.com/code/leewanhung/t5-grammar

[2] Cquentin48. “Grammar correction.” Kaggle, 2024. https://www.kaggle.com/code/cquentin48/grammar-correction#Grammar-correction-using-custom-deep-learning-model
[3] Dario Cioni. “C4 200M Grammar Error Correction dataset.” Kaggle, 2023. https://www.kaggle.com/datasets/dariocioni/c4200m
[4] Papers with Code. “Grammatical Error Correction.” Papers with Code, 2023. https://paperswithcode.com/task/grammatical-error-correction
[5] Microsoft. “UniML.” GitHub, 2023. https://github.com/microsoft/unilm
[6] Yassin522. “English-Grammar-Error-Correction.” GitHub, 2024. https://github.com/Yassin522/English-Grammar-Error-Correction
[7] Ben Newman. “S24 Hugging_Face_Transformers_Tutorial.” Google Colab, Spring 2024. https://colab.research.google.com/drive/13r94i6Fh4oYf-eJRSi7S_y_cen5NYkBm
[8] Jacob Murel Ph.D, Joshua Noble. “What is an encoder-decoder model?.” IBM Think. 2024. https://www.ibm.com/think/topics/encoder-decoder-model