2025

Comparison of Grammar Correction Models: LanguageTool vs. BERT

NEGAR LALEH

Table of Contents

Introduction

Grammar correction plays a crucial role in natural language processing (NLP). In this report, we compare two different approaches for automatic grammar correction:

1. **LanguageTool** (Rule-Based and Dictionary-Based Approach)
2. **BERT (Bidirectional Encoder Representations from Transformers)** (Deep Learning Approach)

We analyze their performance in terms of **Accuracy** on a dataset containing sentences with and without grammatical errors.

Dataset

The orignal dataset consists of eight columns:

- **Number of corrections**: 0 for grammatically correct sentences and more for incorrect ones, the number of corrections shows the number of the different corrections this sentence has.
- **Serial number**
- **the URL of the entry**
- **Sentence number**: 0 is the title Sentence written by a learner of English
- **Corrected Sentences** (If exists)
- **Original sentence**

The original has to go through some changes to be usable, these are the steps taken to achive that:

1. **Load the Data:**
    - Read th CSV file (entries.train.csv) into a Pandas DataFrame (df).
    - Remove unnecessary columns 1, 2, and 3.
    - Filter out rows where the column 'Good luck on your new start !'(original sentences) has missing values.
2. **Rename Columns and Clean Data:**
    - Assign meaningful column names (Number_of_corrections, Original_sentence, Correction_1, etc.).
    - Replace missing values with empty strings.
3. **Prepare Training Data:**

- Iterates through each row, extracting the original sentence and its corrections, If no corrections exist, the original sentence is paired with itself. Otherwise, each correction is paired with the original sentence.
- To make a label a list (has_correction_column) tracks whether corrections exist (1 if there's a correction, 0 if not).

4. **Create and Save Training Dataset:**
   - Convert the processed data into a new DataFrame (training_df) with columns Input, Output, and Has_Correction, and save it.
5. **Final Processing and Splitting:**
   - Reads the saved file and renames the Has_Correction column to label.
   - because of the huge dataset and the lack of proper equipment only 50 percent of this data set has been used.

Model Descriptions

1. LanguageTool Approach

This approach involves:

- Using LanguageTool to apply grammar correction rules.
- Comparing the final output with the original input to classify sentences as correct or incorrect.
- The predicted label is **1** if the final corrected text differs from the input, otherwise **0**.

LanguageTool is an Open Source proofreading software for English, Spanish, French, German, Portuguese, Polish, Dutch, and more than 20 other languages. It finds many errors that a simple spell checker cannot detect. It works by applying rule-based methods, statistical models, and machine learning techniques to detect and correct errors in text.

2. BERT-Based Classification Approach

This method leverages BERT, a pre-trained transformer model:

Data Preprocessing

The input data consists of sentences (df_end.Input.values) and their corresponding labels (df_end.label.values), where labels are binary:

- 0 → No correction needed.
- 1 → Correction required.

To prepare the data for BERT, the following steps have been applied:

- Adding BERT-Specific Tokens:
  Each sentence is enclosed with [CLS] (classification token) at the beginning and [SEP] (separator token) at the end.

- Tokenization**:**
  To ensure the words not present in BERT's vocabulary can still be processed, the BertTokenizer is used to tokenize the sentences into subword units.

- Padding and Truncation**:**
  To ensure uniform input length, all tokenized sentences are truncated or padded to a maximum length of 128 tokens.

- Attention Masks:
  An attention mask is created, where:1 indicates a real token.0 indicates padding.This allows BERT to ignore padding during processing.

Model Initialization

The BERT model for sequence classification (BertForSequenceClassification) is initialized with num_labels=2 for binary classification. The model is then moved to a GPU for hardware acceleration.

Training Setup and Optimization

- Batch Processing:
  The dataset is converted into PyTorch tensors and loaded into a DataLoader for mini-batch training with a batch size of 32.

- Optimizer:
  The Adam optimizer (BertAdam) is used with a samll learning rate of and a weight decay strategy to prevent overfitting.

Training Process

The model is trained for 4 epochs with the following steps for each batch:

1. Forward pass: Compute the loss using BERT's built-in classification head.
2. Backpropagation: Compute gradients and update weights using gradient descent.
3. Loss tracking: The average training loss is recorded after each epoch.

Model Evaluation

After training, the model is set to evaluation mode, and its performance is assessed on the validation dataset. The evaluation follows these steps:

1. Predictions are generated without gradient computation (torch.no_grad()).
2. Predicted logits are converted to class labels.
3. Classification accuracy is computed:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

We evaluate both models using Accuracy:

Model	Accuracy
LanguageTool	0.58
BERT	0.72

- LanguageTool is rule-based and works well for common grammatical structures but may struggle with complex grammatical constructs and contextual errors.
- BERT leverages deep learning and contextual embeddings, providing better generalization, especially for nuanced grammar mistakes.
- Speed: The rule-based approach is generally faster, while BERT is computationally expensive but more accurate.

Conclusion

Based on the results, the choice of the model depends on the use case:

- If speed and efficiency are priorities with moderate accuracy, LanguageTool is preferable.
- If accuracy and contextual understanding are crucial, BERT is the better choice.

Further improvements could involve combining both approaches, using BERT for classification and rule-based tools for refinement.