

Grammar Correction using SQLite + Transformers

Building an Automated Grammar
Correction Pipeline

Presented by Kashaf Saeed



Introduction

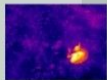


Grammar Correction using NLP and Transformers

This project focuses on the use of deep learning and NLP techniques for grammar correction, creating an efficient and effective tool for enhancing sentence structure and clarity.

Project Overview

The aim of this project is to develop an automated system that analyzes grammatical structure in sentences to identify errors and suggest corrections, resulting in a robust data management system.



Technologies Used

This project utilizes various technologies for efficient data storage. It leverages NLP and deep learning techniques to analyze large volumes of text and identify grammatical errors, ensuring the system is scalable and efficient for data management and processing.



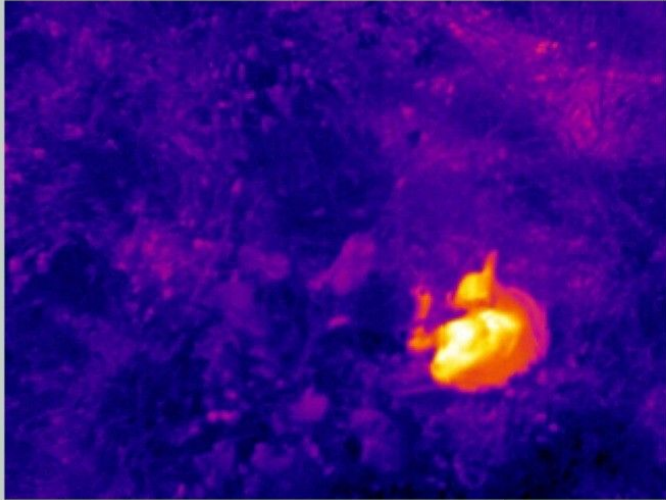
Grammar Correction using SQLite + Transformers

This project leverages SQLite for data storage and T5 Transformer for grammar correction, creating an efficient and effective tool for enhancing sentence structure and accuracy.

Project Overview

The aim of this project is to develop an automated system that enhances grammatical accuracy in sentences by utilizing a pre-trained transformer model combined with a robust data management system.





Technologies Used

Key technologies include SQLite for efficient data storage, T5 Transformer from HuggingFace for natural language processing, and libraries like Pandas and Scikit-learn for data manipulation and model training.

Objective and Data Storage

Objective

The main objective of this project is to automatically correct grammar or sentence using natural language processing. This automatic grammar correction tool is designed to enhance writing accuracy effectively.



Data Storage Structure

The data is stored in an SQL database with a single table containing all data collected. It includes user information, error correction, and correction suggestions. The system also stores the original and corrected versions of the input text for future reference.



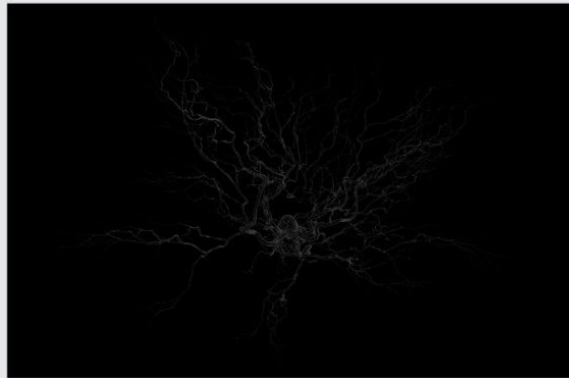
Sample Data Entries

Sample entries in the database include:
1. User ID: 123456789
2. Original Text: "The quick brown fox jumps over the lazy dog."
3. Corrected Text: "The quick brown fox jumps over the lazy dog."
4. Error Type: Grammar
5. Suggestion: "The quick brown fox jumps over the lazy dog."



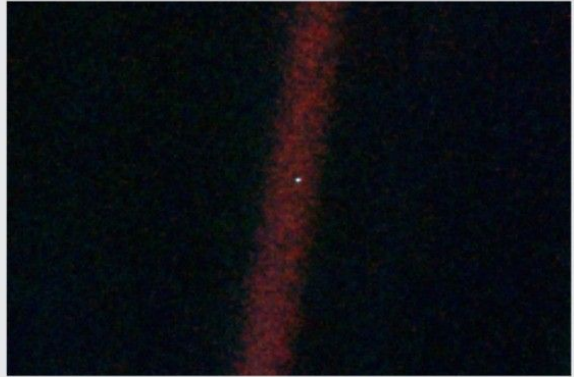
Objective

The main objective of this project is to automatically correct grammar in sentences using a trained transformer model. This innovative approach leverages Natural Language Processing techniques to enhance writing accuracy effortlessly.



Data Storage Structure

The data is stored in an SQLite database with a simple table structure consisting of three columns: id, incorrect, and correct. This layout facilitates quick access and organization of correction data for the model.





Sample Data Entries

Example entries in the database include: 1) Incorrect: "She have a car." and Correct: "She has a car."; 2) Incorrect: "They goes to school." and Correct: "They go to school." These samples illustrate common grammatical errors and their corrections.

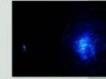
Data Processing Steps for Grammar Correction

Data Preprocessing

Data is collected from the following sources, where each entry includes an original sentence and its corresponding corrected version. The data is then used to create a robust foundation for training the machine learning model.



Data Splitting



Data is split into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate the model's performance. This process ensures that the model is trained on a diverse set of data and can generalize to new, unseen data.

Dataset Conversion

Dataset is converted into a format suitable for training the machine learning model. This involves converting the dataset into a structured format, such as a CSV file, and ensuring that the data is properly labeled and formatted.



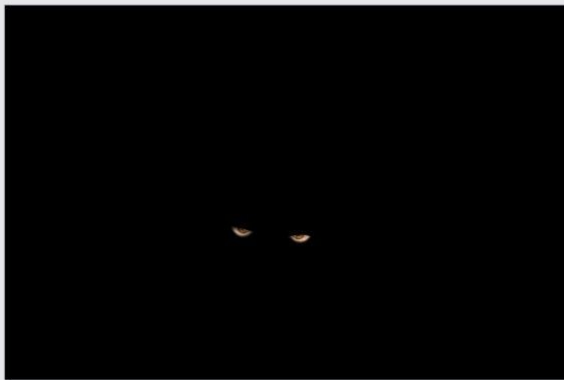
Tokenization Process

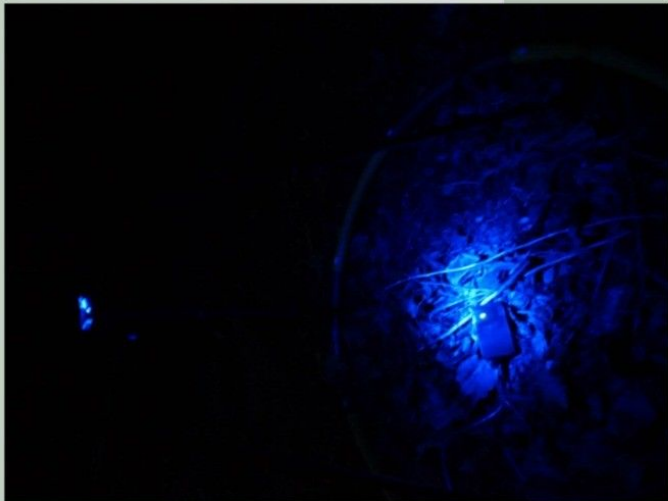


Preprocessing is essential to ensure that the training data is in a format suitable for the machine learning model. This process involves converting the dataset into a structured format, such as a CSV file, and ensuring that the data is properly labeled and formatted.

Data Preprocessing

Data is retrieved from the SQLite database, where each entry includes an incorrect and its corresponding correct sentence. Efficient data retrieval ensures a robust foundation for training the transformer model.



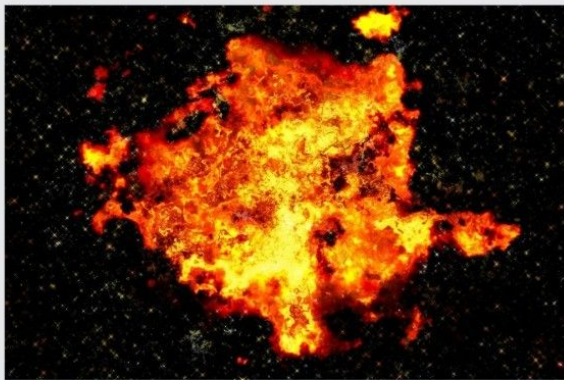


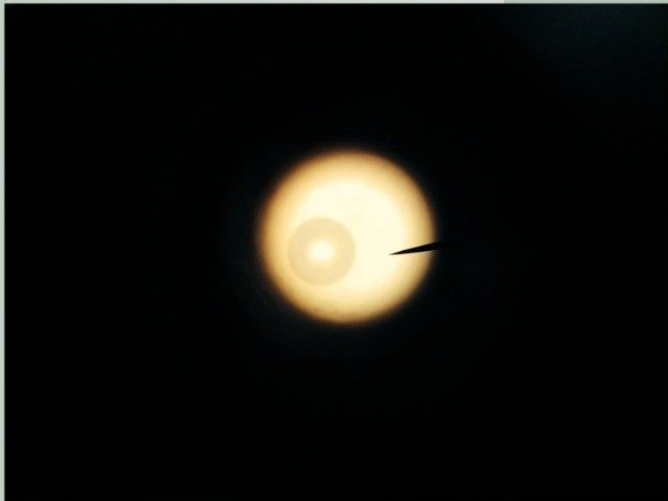
Data Splitting

Train, validation, and test data splits are performed using Scikit-learn, ensuring distinct subsets for training the model and validating its accuracy. This methodology promotes unbiased performance evaluation and model tuning.

Dataset Conversion

DataFrames are converted into HuggingFace Datasets, facilitating compatibility with machine learning frameworks. This transformation streamlines the model training process and optimizes data handling.





Tokenization Process

T5Tokenizer is employed to tokenize data, transforming sentences into model-readable formats. The preprocessing format 'fix: <incorrect sentence>' is used, guiding the model output to generate corrected text.



Model Training and Evaluation

4.1 Model Structure: T5forConditionalGeneration

The T5forConditionalGeneration model is particularly suited for generative tasks where the model needs to generate text based on a given context. It utilizes a transformer architecture that can generate text sequences while learning from examples, enhancing accuracy and fluency in corrections.



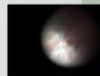
4.2 Training Setup: Seq2SeqTrainer

Training is conducted using the Seq2SeqTrainer, which facilitates the process of feeding the model training examples and generating predictions. The trainer monitors performance during the training phase.



4.3 Evaluation Metric: BLEU Score

The BLEU score is a standard metric for evaluating the quality of text generated by the model. It compares the generated text against reference text, providing a quantitative measure of performance.



4.4 Results: Performance Overview

The results show that the model's performance is significantly improved after training. The Seq2SeqTrainer effectively optimizes the model's parameters, leading to higher BLEU scores and more accurate predictions.



4.5 Conclusion and Future Directions

The generated correction results are highly accurate and provide valuable insights into the model's performance. Future work will focus on refining the model's architecture and exploring its application in other domains.



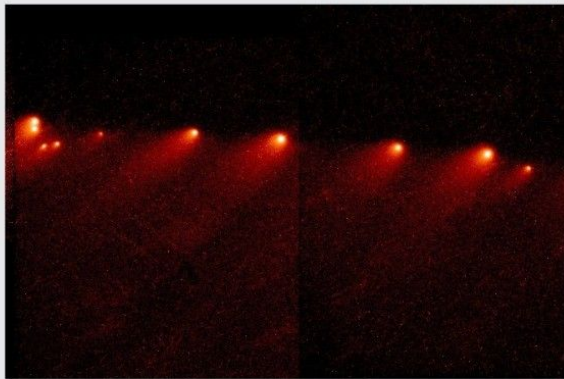
4.1 Model Structure: T5ForConditionalGeneration

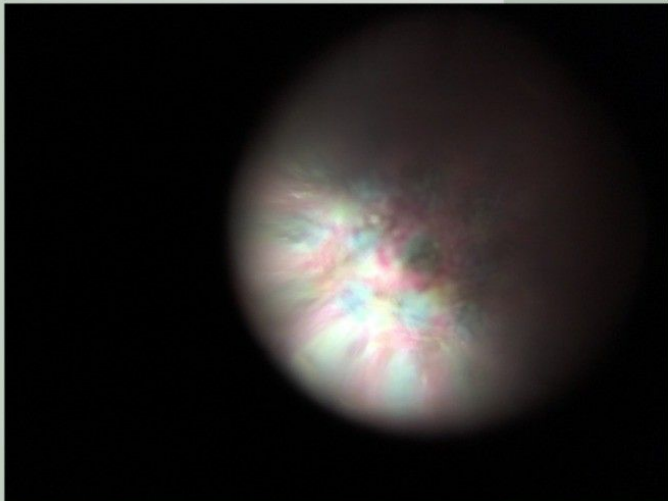
The T5ForConditionalGeneration model is particularly suited for grammatically correcting sentences based on context. It utilizes a transformer architecture that can generate text sequences while learning from examples, enhancing accuracy and fluency in corrections.



4.2 Training Setup: Seq2SeqTrainer

Training is facilitated using Seq2SeqTrainer, which streamlines the process of fine-tuning the T5 model. Key training arguments include learning rate adjustments, batch size configurations, and epochs, optimizing model performance during the training phase.





4.3 Evaluation

Metric: BLEU Score

The effectiveness of the grammar correction model is evaluated through BLEU score computation using the sacrebleu library. This metric assesses the similarity between the model's output and human-level reference sentences, quantifying translation accuracy and fluency.

4.4 Results: Performance Overview

Examples reveal the model's capability, such as transforming 'She have a car.' to 'She has a car.' This showcases not only the model's grammatical correction skills but also reflects an overall high performance measured through obtained BLEU scores.





4.5 Conclusion and Future Directions

The grammar correction pipeline is lightweight and facilitates easy database integration. Future work includes training on larger datasets, fine-tuning for domain-specific texts, and developing a user-friendly web-based inference interface to enhance accessibility.