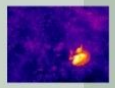
Grammar Correction using SQLite + Transformers

Building an Automated Grammar Correction Pipeline

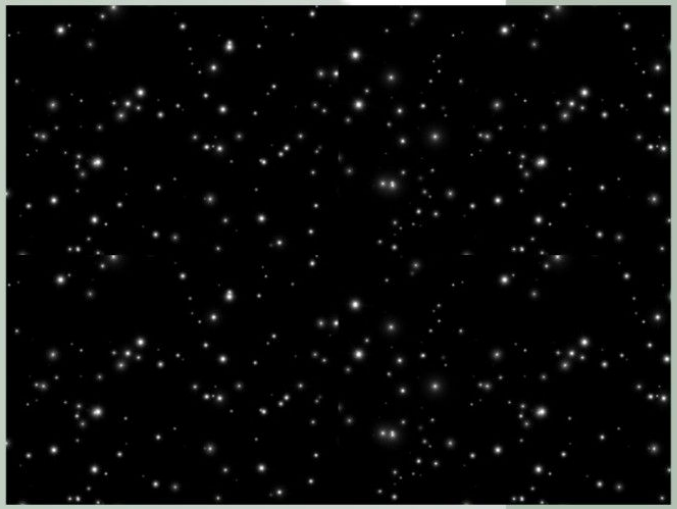
Presented by Kashaf Saeed

Introduction

Grammar Correction using SQLite + Transformers



Technologies Used

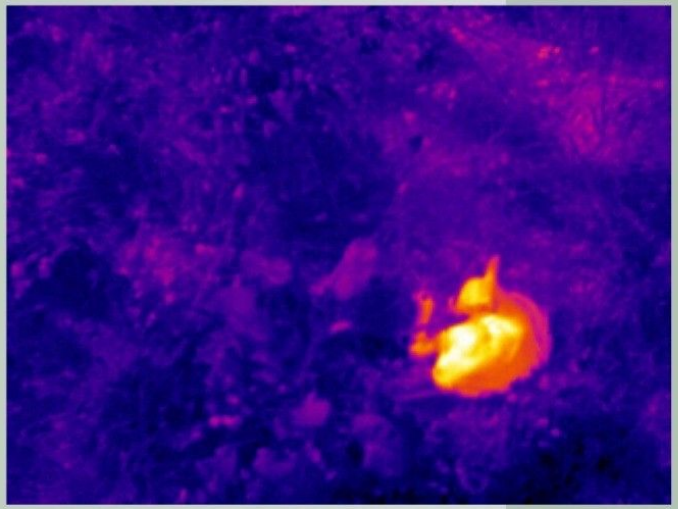


Grammar Correction using SQLite + Transformers

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.



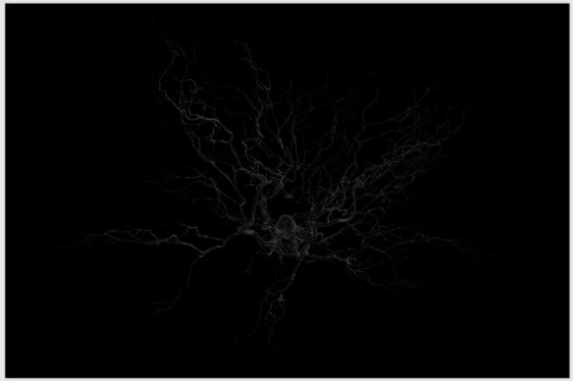
Technologies Used

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

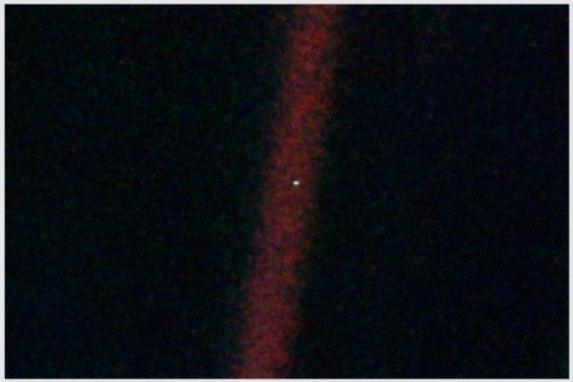
Objective

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



Data Storage Structure

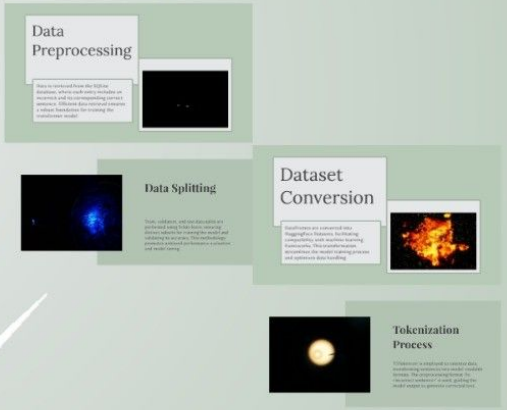
The data is stored in an SQLite database with a simple table structure consisting of three columns: id, incorrect, and correct. This layout facilitates quick access and organization of correction data for the model.



Sample Data Entries

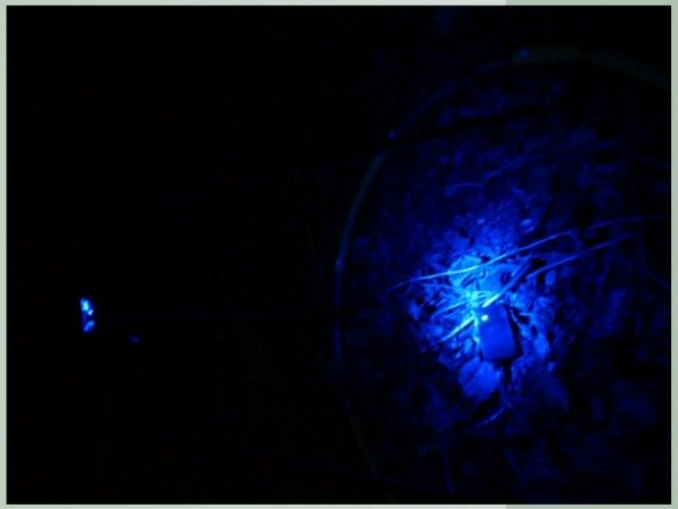
Example entries in the database include: 1) Incorrect: "She have a car." and Correct: "She has a car."; 2) Incorrect: "They goes to school." and Correct: "They go to school." These samples illustrate common grammatical errors and their corrections.

Data Processing Steps for Grammar Correction



Data Preprocessing

Data is 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.

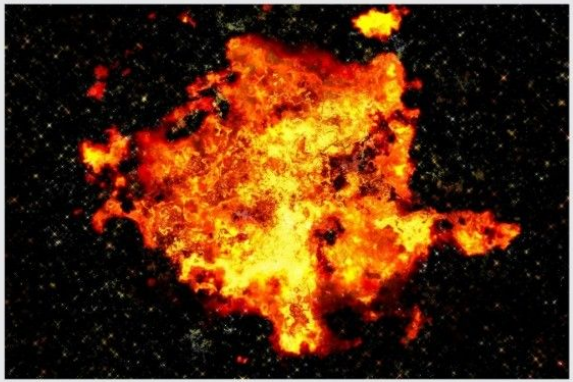


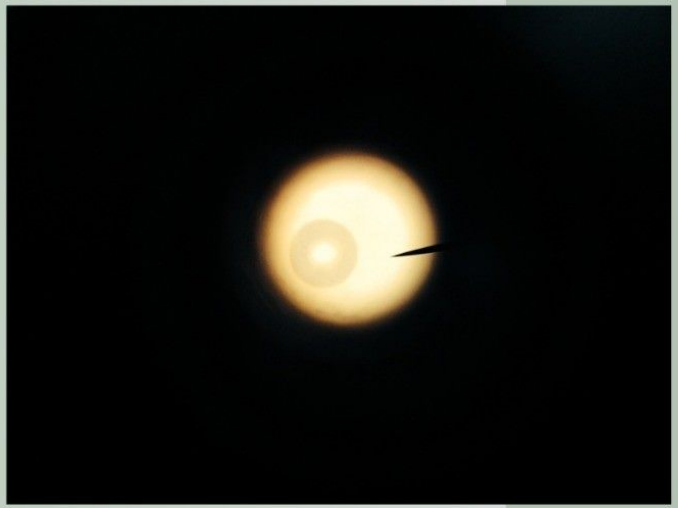
Data Splitting

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.

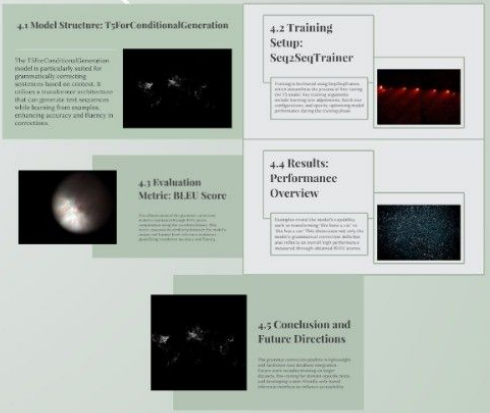




Tokenization Process

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

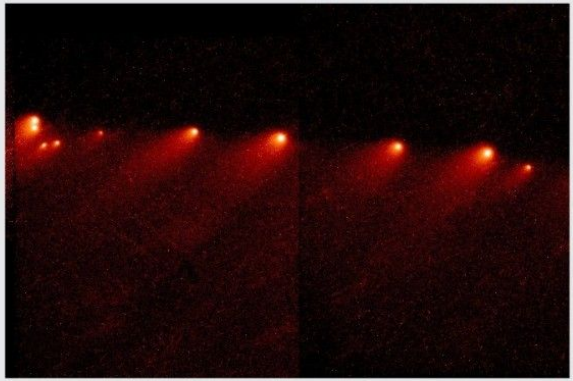


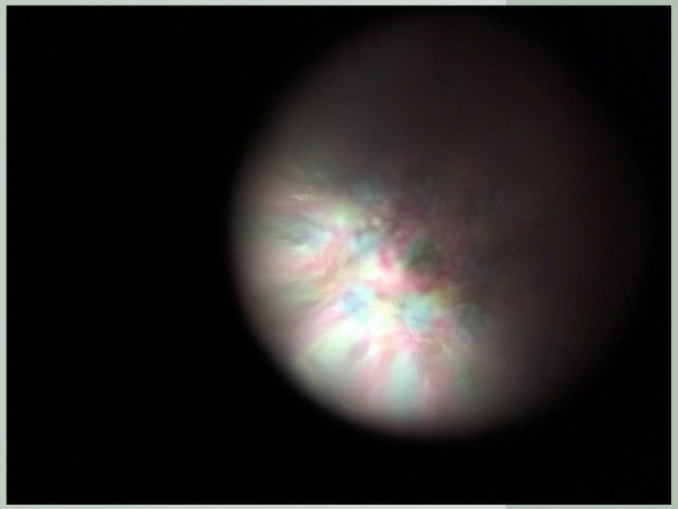
4.1 Model Structure: T5ForConditionalGeneration

The T5ForConditionalGeneration model is particularly suited for 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.