# README: Grammar Detection and Correction Pipeline

## Introduction

This project implements a pipeline for Grammar Detection and Correction using pretrained transformer models. It consists of two components:  
1. Grammar Detection: Identifies whether a sentence contains grammatical errors.  
2. Grammar Correction: Corrects the grammatical errors in flagged sentences.

## Prerequisites

Ensure the following Python packages are installed:  
- pandas  
- transformers  
- datasets  
- torch  
- scikit-learn  
- nltk  
  
Installation Command:  
`!pip install pandas transformers datasets torch scikit-learn nltk`

## Dataset

The dataset contains ungrammatical sentences, their corrections, and error types. It has the following columns:  
- `Ungrammatical Statement`: Input sentence with errors.  
- `Standard English`: Corrected sentence.  
- `Error Type`: Type of grammatical error (optional).  
  
The dataset is preprocessed to:  
- Remove missing values.  
- Create binary labels (`is\_error`) for grammar detection.  
- Prepare input-output pairs for correction tasks.

## Components

### 1. Grammar Detection

The detection component uses a BERT-based model to classify sentences as grammatically correct or incorrect.  
Steps:  
- Tokenize sentences using `bert-base-uncased` tokenizer.  
- Prepare data for training and testing.  
- Train a BERT model for binary classification.

### 2. Grammar Correction

The correction component fine-tunes a T5 model to transform ungrammatical sentences into corrected English.  
Steps:  
- Add prefixes ('grammar correction: ') to input sentences.  
- Tokenize inputs and targets using the T5 tokenizer.  
- Train the T5 model using the Hugging Face `Trainer` API.

## Training

The models are trained as follows:  
- BERT: Fine-tuned for 3 epochs to detect grammar errors.  
- T5: Fine-tuned for 3 epochs to correct grammar errors.

## Testing

For Grammar Detection:  
- Use classification metrics (accuracy, precision, recall, F1-score).  
  
For Grammar Correction:  
- Input sentences with errors into the T5 model.  
- Compare outputs to ground truth corrections.

## Saving Models

Both models are saved for future use:  
- Grammar Detection Model: `./bert\_grammar\_detection`  
- Grammar Correction Model: `./t5\_grammar\_correction`

## Running the Pipeline

1. Load the dataset and preprocess it.  
2. Train the BERT model for detection and T5 model for correction.  
3. Test the pipeline by:  
 - Detecting errors in input sentences using the BERT model.  
 - Correcting errors using the T5 model.

## Issues and Debugging

- Ensure the dataset is properly formatted and preprocessed.  
- Check model outputs for unexpected behavior (e.g., language inconsistencies).  
- Fine-tune parameters (e.g., learning rate, epochs) for better performance.

## Conclusion

This pipeline efficiently combines grammar detection and correction tasks using state-of-the-art transformer models. It is flexible and can be adapted to various datasets and language processing tasks.