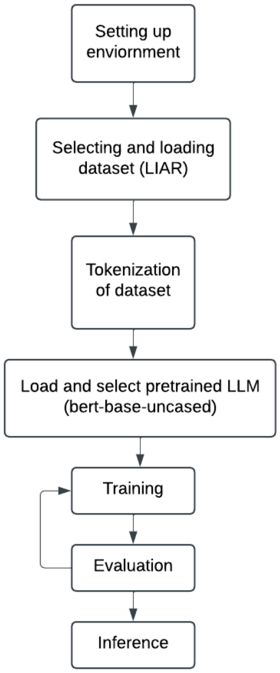
Factify: Fake News Detection using Fine-Tuned LLM

Overview

Factify is an AI-powered tool designed to detect false information in news articles and statements. By fine-tuning a pre-trained language model on the **LIAR dataset**, Factify classifies statements into six categories: 'False', 'Half-True', 'Mostly-True', 'True', 'Barely-True', and 'Pants-Fire'. This project demonstrates how fine-tuning a transformer model like BERT can be applied to the task of fake news detection.


Setting up enviornment
Selecting and loading dataset (LIAR)
Tokenization of dataset
Load and select pretrained LLM (bert-base-uncased)
Training
Evaluation
Inference

Objective

The goal of this project is to create an efficient and accurate fake news detection system that can classify the truthfulness of a given statement. By training the model on the LIAR dataset, which contains labeled examples of statements along with their truthfulness ratings, Factify aims to distinguish between true and false claims.

How It Works

1. **Dataset**: I used the **LIAR dataset**, which consists of 12,800 human-written short statements categorized as true, false, or somewhere in between. These labels are as follows:
    - **False**
    - **Half-True**
    - **Mostly-True**
    - **True**
    - **Barely-True**
    - **Pants-Fire**
2. **Model Choice**: I chose **BERT (Bidirectional Encoder Representations from Transformers)**, a pre-trained language model known for its effectiveness in various natural language processing tasks. The model was fine-tuned on the LIAR dataset to learn the patterns in text that differentiate between true and false statements.
3. **Fine-Tuning Process**:
    - I used the **transformers** library by Hugging Face to load the pre-trained BERT model (`bert-base-uncased`).
    - The **LIAR dataset** was preprocessed by tokenizing the text to make it compatible with the BERT input format.
    - The model was fine-tuned using **PyTorch** and trained for 3 epochs, using a batch size of 8 and a learning rate of 2e-5.
4. **Evaluation**: After training, the model's performance was evaluated using accuracy metrics, with results showing the model's ability to classify the statements accurately.
5. **Inference**: The trained model can now be used to classify new statements. When a statement is input, it is tokenized, passed through the model, and a prediction is made, mapping the output to one of the six truthfulness categories.

Key Technologies Used

- **Transformers** (Hugging Face): For pre-trained models and fine-tuning.
- **PyTorch**: For training and evaluation of the model.
- **LIAR Dataset**: The dataset used to train the model.
- **Python**: The main programming language used for the project.

Results

After training and evaluation, the model achieved promising results in classifying statements. The evaluation showed an **eval_loss** of approximately **1.74**, with a reasonable processing speed of **32.91 samples per second** during inference.

This was run on colab and the model selection and epochs were kept due to low computation power and colab crash issues.

Future Work

In the future, I plan to:

- Further optimize the model by experimenting with different architectures, hyperparameters, and fine-tuning techniques.
- Explore additional datasets to expand the model's knowledge and improve accuracy.
- Integrate the model into a web application for real-time fake news detection.