Fake News Detection Using Fine-Tuned DistilBERT

Powered by Hugging Face Transformers and Gradio

Presented by: Ganga Vamsik S

Introduction

- What is Fake News?
- Misleading or false information presented as news.
- Why Fake News Detection?
- To combat misinformation and improve the credibility of information.

Objective

- Primary Goal: Build a machine learning model to classify news as 'Fake' or 'Real.'
- Secondary Goals:

  Fine-tune a pre-trained transformer model.

  Create an interactive user interface for predictions.

Dataset Overview

- Dataset Used: Binary classification dataset with 'text' and 'label' fields.
- Statistics:

  Number of samples: 2

  Dataset Source: Kaggle.com

Preprocessing

- Steps:

  Text tokenization using the DistilBERT tokenizer.

  Padding and truncation to fit the maximum sequence length (512 tokens).

- Why Preprocessing?

  To standardize input data for the model.

Model Selection

- Why DistilBERT?
- - Lightweight version of BERT for faster performance.
- - Suitable for binary classification tasks.
- Model Architecture:
- - Pre-trained transformer with added classification head.
- - Outputs logits for binary labels.

Model Training Setup

- Training Libraries:
- - `transformers` for model loading and training.
- - `torch` for handling tensors and training steps.
- Training Configuration:
- - Optimizer: AdamW
- - Scheduler: Linear Warm-Up
- - Metrics: Accuracy, Precision, Recall, F1 Score

Model Training Code

```python
# Load the pre-trained model with two output labels
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2)

# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)

# Train the model
trainer.train()

# Save the trained model and tokenizer
save_path = "/content/fine_tuned_model"  # path to save your model
model.save_pretrained(save_path)
tokenizer.save_pretrained(save_path)

print(f"Model and tokenizer saved to {save_path}")
```

Saving the Model

- Purpose:

  Store the fine-tuned model for future use.

- Key Code:

```
# Save th Loading... model and tokenizer
save_path = "/content/fine_tuned_model"  # path to save your model
model.save_pretrained(save_path)
tokenizer.save_pretrained(save_path)

print(f"Model and tokenizer saved to {save_path}")
```

Gradio Interface

- Why Gradio?

  Simplifies deployment of machine learning models with an intuitive interface.

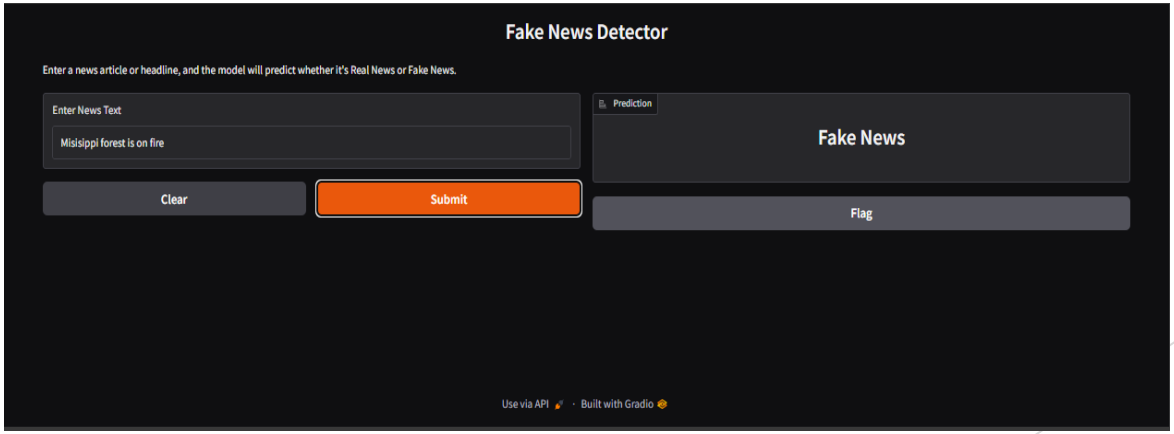
- Features:
- - Text input box for user input.
- - Real-time prediction display.

Gradio Interface Code

```python
import gradio as gr
import torch
import numpy as np
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Path to your saved fine-tuned model
saved_model_path = "/content/fine_tuned_model"  # Change this to saved model directory

# Load the fine-tuned model and tokenizer
tokenizer = AutoTokenizer.from_pretrained(saved_model_path)
model = AutoModelForSequenceClassification.from_pretrained(saved_model_path)

# Prediction function
def predict_text(text):
    # Tokenize the input text
    inputs = tokenizer(
        text,
        return_tensors="pt",
        padding="max_length",
        truncation=True,
        max_length=512
    )
    with torch.no_grad():
        # Make predictions
        outputs = model(**inputs)
    logits = outputs.logits
    prediction = np.argmax(logits.numpy(), axis=1)[0]
    return "Fake News" if prediction == 1 else "Real News"

# Create Gradio interface
interface = gr.Interface(
    fn=predict_text,                    # Function to process input
    inputs=gr.Textbox(label="Enter News Text"),  # Input box for user text
    outputs=gr.Label(label="Prediction"),       # Display prediction
    title="Fake News Detector",
    description="Enter a news article or headline, and the model will predict whether it's Real News or Fake News."
)

# Launch  interface
interface.launch()
```

Deployment

- Steps to Deploy:
- 1. Load the fine-tuned model and tokenizer.
- 2. Use Gradio for the interface.
- 3. Launch the app locally or on the web.
- Demo:



Results

- Metrics Achieved:
- - Accuracy: [Insert Value]
- - F1 Score: [Insert Value]
- Example Prediction:
- - Input: 'Breaking news: Scientists discover a new planet.'
- - Output: ‘Fake News'

Challenges & Improvements

- Challenges:
- - Limited dataset size.
- - Balancing the trade-off between speed and accuracy.
- Future Work:
- - Explore larger transformer models like BERT or RoBERTa.
- - Add support for multilingual fake news detection.
- - Fine-tune on more diverse datasets.

Conclusion

- Key Takeaways:
- - Successfully built and deployed a fake news detector.
- - Leveraged state-of-the-art NLP models and tools.
- - User-friendly interface for predictions.
- Acknowledgments:
- - Hugging Face, Gradio, and PyTorch communities.