CONTEXTUAL LANGUAGE UNDERSTANDING WITH TRANSFORMER MODELS

PHASE 3: DEVELOPMENT

COLLEGE NAME: NAGARJUNA COLLEGE OF ENGNEERING AND TECHNOLOGY

TEAM MEMBERS: 4 MEMBERS

Name: LIKHITHA T R
CAN ID Number: 35246733

Name: BHARANI H D
CAN ID Number: 35257542

Name: AISHWARYA N
CAN ID Number: 35257595

Name: ASIF AHMED
CAN ID Number: 35246985

1. Introduction

Phase 3 marks the transition from theoretical research to hands-on development of a transformer-based system for contextual language understanding. Contextual language models such as BERT, GPT, and RoBERTa revolutionize Natural Language Processing (NLP) by capturing deeper linguistic representations than traditional models. This phase involves implementing, fine-tuning, and evaluating such models on real-world tasks like sentiment analysis, text classification, or question answering.

The goal is to develop a practical prototype that demonstrates the power of transformer models in understanding the nuances of human language within context.

---

2. Key Goals of the Development Phase

- Implement a transformer-based architecture using pre-trained models.
- Preprocess and tokenize raw text data for compatibility with transformers.
- Fine-tune the model on a downstream NLP task (e.g., classification, QA).
- Evaluate the performance using robust metrics.
- Ensure reproducibility and model portability for deployment

3. System Architecture Overview

csharp
CopyEdit
[Raw Text Input]
↓
[Text Cleaning & Preprocessing]
↓
[Tokenizer (BERT/RoBERTa)]
↓
[Transformer Model (Encoder)]
↓
[Classifier Head (Dense Layer)]
↓
[Softmax/Output Layer]
↓
[Prediction]

This pipeline integrates preprocessing, tokenization, contextual encoding, and classification. The classifier head is task-specific and added on top of the transformer.

4. Tools & Frameworks

Component	**Technology Used**
Language	Python 3.10+
Framework	PyTorch / TensorFlow
Pretrained Models	BERT, RoBERTa, DistilBERT
Libraries	Hugging Face Transformers, Datasets, Scikit-learn
IDEs	Jupyter Notebook, Google Colab, VS Code
Visualizations	Seaborn, Matplotlib

---

5. Dataset and Task

Dataset Used: IMDb Movie Reviews (for Sentiment Analysis)

- Contains 50,000 reviews (25k train, 25k test).
- Binary classification: Positive / Negative sentiment.
- Text length varies, necessitating truncation/padding.
- *Alternate datasets for similar tasks may include SQuAD (for question answering), AG News (for topic classification), or TREC (for question type classification).*

---

6. Preprocessing Steps

1. **Text Cleaning:** Removal of punctuation, HTML tags, stopwords.
2. **Tokenization:** Using AutoTokenizer to convert text to input IDs & attention masks.
3. **Padding/Truncation:** To uniform sequence lengths (e.g., max_length = 128).
4. **Encoding Labels:** Binary or categorical encoding for labels.

Example Code Snippet (Hugging Face):

python
CopyEdit
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt")

---

7. Model Implementation

Pretrained Model: BERT-base-uncased

- 12-layer transformer encoder
- 110 million parameters
- Trained with masked language modeling + next sentence prediction

Fine-tuning Process:

- Classification head: 1 or 2 linear layers added on top of BERT's [CLS] token embedding.
- Loss: CrossEntropyLoss
- Optimizer: AdamW
- Learning Rate: 2e-5
- Epochs: 3 to 5

Training is done using PyTorch's Trainer API or custom training loop.

---

8. Evaluation Metrics

- **Accuracy:** Overall correct predictions
- **Precision & Recall:** For imbalanced classes
- **F1 Score:** Harmonic mean of precision & recall
- **Confusion Matrix:** Visualizing classification performance
- **ROC Curve / AUC:** (optional for binary tasks)

Results Example:

Metric	Value
Accuracy	93.4%
F1 Score	0.931
Precision	0.928
Recall	0.936

---

9. Model Interpretability & Visualization

To better understand the model's predictions:

Attention Maps: Visualize which tokens receive more focus.

LIME / SHAP: Explain individual predictions.

Attention maps reveal how transformers prioritize different words in the input.

---

10. Challenges Encountered

Challenge	Mitigation Strategy
High GPU memory usage	Used smaller models (DistilBERT) or Colab Pro
Long training time	Early stopping, checkpointing
Token limit issues	Truncation, sliding window approach
Dataset imbalance	Applied class weights or data augmentation

---

11. Output and Model Export

- Model checkpoint saved using model.save_pretrained() and tokenizer.save_pretrained().
- Compatible with APIs or deployment pipelines (e.g., FastAPI, Flask).
- Inference-ready format supports batch or real-time queries