AI/ML Advanced Task Report

DeveloperHub

Author:
Zoya Hafeez

Submitted To:
DevelopersHub Corporation

Date:
July 17, 2025

Task 1: News Topic Classifier Using BERT

Problem Statement & Objective

The goal of this task is to develop a News Topic Classifier using the `bert-base-uncased` model. The model should classify news headlines and descriptions into topic categories using the AG News dataset.

Dataset Loading & Preprocessing

- Used the AG News Dataset from Hugging Face.
- Combined title and description fields into a single text field.
- Tokenized text using the BERT tokenizer with padding and truncation.
- Encoded and wrapped data into a custom PyTorch Dataset.

Model Development & Training

- Fine-tuned BERT using `Trainer` from Hugging Face Transformers.
- Model: `BertForSequenceClassification`.
- Training configured with batch size, epochs, and logging strategy.

Evaluation

- Metrics used: Accuracy and F1-score.
- Model evaluated on the test dataset using `Trainer.predict()`.

Final Summary / Insights

- BERT performed well in classifying news topics.
- Fine-tuning leveraged transfer learning for improved performance.
- Future improvements could include additional model tuning and more visual diagnostics.

Task 2: End-to-End ML Pipeline with Scikit-learn

Problem Statement & Objective

Develop an end-to-end machine learning pipeline using scikit-learn to predict customer churn from the Telco Churn dataset. The pipeline should be production-ready and reusable.

Dataset Loading & Preprocessing

- Dataset: Telco Customer Churn Dataset.
- Handled missing values, encoded categorical variables, and scaled numerical features.
- Used `Pipeline` and `ColumnTransformer` for modular preprocessing.

Model Development & Training

- Models used: Logistic Regression and Random Forest.
- Applied `GridSearchCV` for hyperparameter tuning.

Evaluation

- Evaluated using accuracy, precision, recall, F1-score, and ROC-AUC.

Final Summary / Insights

- Pipelines increased reusability and maintainability.
- Logistic Regression provided explainability; Random Forest gave better accuracy.
- Model was exported using `joblib` for deployment.

Task 4: Context-Aware Chatbot Using LangChain

Problem Statement & Objective

The objective of this task is to build a conversational chatbot that can maintain context across interactions and retrieve external information from a custom document store. The chatbot must be implemented using LangChain or Retrieval-Augmented Generation (RAG) and deployed with Streamlit.

Dataset Loading & Preprocessing

- A custom knowledge base was created using publicly available documents (e.g., Wikipedia or internal text files).
- Text documents were chunked into manageable segments.
- Each chunk was converted into vector embeddings using OpenAI or HuggingFace embedding models.
- A vector store (e.g., FAISS or Chroma) was created to allow similarity-based retrieval.

Model Development & Training

- Implemented `ConversationalRetrievalChain` using LangChain.
- Integrated a memory buffer to store and reuse chat history (context-awareness).
- The model retrieves the most relevant document chunks in real-time based on the user query and passes them to the LLM.
- Used a pre-trained LLM (e.g., claude AI or open-source alternative) to generate human-like responses.

Evaluation

- Manual evaluation of chatbot performance based on:
  - Relevance of responses
  - Context retention across multiple queries
  - Accuracy of retrieved information
- Also tracked latency and quality of document retrieval.

Final Summary / Insights

- The chatbot successfully maintained conversational context using memory integration.
- Retrieval-Augmented Generation helped the model provide grounded, factual answers using the embedded documents.
- The Streamlit app provided a lightweight, user-friendly interface.
- Future enhancements may include multi-document indexing, semantic reranking, and improved chat history visualization.

Task 5: Auto Tagging Support Tickets Using LLM

Problem Statement & Objective

Automatically classify customer support tickets into predefined categories using Large Language Models (LLMs). The objective was to experiment with zero-shot, few-shot, and fine-tuning techniques to understand how well LLMs can generalize to the task of auto-tagging.

Dataset Loading & Preprocessing

- Used a free-text support ticket dataset containing user-generated support queries.
- Preprocessed the text by lowercasing, removing special characters, and trimming whitespace.
- Constructed prompts suitable for zero-shot and few-shot learning scenarios.
- Encoded labels for fine-tuning and split dataset into train/test sets.

Model Development & Training

- **Zero-shot classification**: Used a pre-trained LLM (e.g., OpenAI/GPT-3.5) with simple prompts and category labels. No training data was used.
- **Few-shot classification**: Added a few representative labeled examples in the prompt to guide the LLM's prediction.
- **Fine-tuned model**: Used a transformer-based model fine-tuned on the labeled dataset for 3 epochs on GPU. Training included adapting a classification head to predict 5 ticket categories.

Evaluation

- Compared results across zero-shot, few-shot, and fine-tuned models.
- Evaluation Metrics:
  - Accuracy: **20%**
  - Precision, Recall, F1-score (Macro and Weighted Average)
- **Fine-tuned Model Results:**
  - Eval Loss: 1.6092
  - Eval Runtime: 3.84s
  - Precision (macro avg): 0.15
  - Recall (macro avg): 0.19
  - F1-score (macro avg): 0.15
- Detailed Performance:
  - **Technical issue**: Precision = 0.15, Recall = 0.05, F1 = 0.08
  - **Billing inquiry**: Precision = 0.00, Recall = 0.00, F1 = 0.00
  - **Cancellation request**: Precision = 0.19, Recall = 0.10, F1 = 0.13
  - **Product inquiry**: Precision = 0.20, Recall = 0.40, F1 = 0.26
  - **Refund request**: Precision = 0.21, Recall = 0.39, F1 = 0.27

Final Summary / Insights

- **Zero-shot classification** was fast and required no labeled data but lacked accuracy and class sensitivity.
- **Few-shot classification** slightly improved performance, especially in more frequent categories, due to contextual examples.
- **Fine-tuning** achieved the highest numerical accuracy (**20%**) but revealed significant issues:
  - Class imbalance led to poor precision and recall for multiple categories.
  - **Billing inquiry** was never predicted correctly.
  - The model overfit to categories like *Product inquiry* and *Refund request*, which had higher recall.
- Future Improvements:
  - Use class weights or data augmentation to address imbalance.
  - Improve label alignment during training and inference.
  - Integrate confidence thresholds for better zero-shot prediction control.