Smart Text Categorization using Transformer + SQL Backend + HTML Frontend with FastAPI

Objective

This project aims to build an end-to-end application that uses a pre-trained Transformer model to classify customer queries fetched from a MySQL database. The predictions are then written back to a new table in the same database.

Key Features

- **End-to-End NLP Pipeline:** From data ingestion to model training and prediction.
- **Transformer-Based Classification:** Utilizes a fine-tuned DistilBERT model for accurate text categorization.
- **MySQL Integration:** Manages data seamlessly with a robust SQL backend.
- **Modular Codebase:** Well-organized scripts for database management, model training, and prediction.
- **Web Interface:** A simple, user-friendly web UI for real-time text classification.

Project Structure

```
text_classifier_project/
├── app.py                  # Main application script for API and UI
├── evaluate.py             # Script to evaluate the model performance
├── index.html              # HTML file for the user interface
├── data/
│   ├── labeled_data.csv    # Sample labeled data for training
│   └── unlabeled_data.csv  # Sample unlabeled data for prediction
├── db/
│   ├── create_db.py        # Script to create and populate the database
│   ├── fetch_queries.py    # Script to fetch data from the database
│   └── write_predictions.py# Script to write predictions back to the database
├── model/
│   ├── train_model.py      # Script to fine-tune the Transformer model
│   └── fine_tuned_model/   # Saved fine-tuned model files
├── predict.py              # Script for making predictions with the fine-tuned model
├── requirements.txt        # Python dependencies
└── README.md               # Project documentation
```

End-to-End Workflow

1. **Database Setup:**
    - The `db/create_db.py` script initializes a MySQL database named `customer_queries`.
    - It creates two tables: `labeled_queries` and `unlabeled_queries`.
    - It populates the tables with sample data from `data/labeled_data.csv` and `data/unlabeled_data.csv`.

2. **Model Fine-Tuning:**
    - The `model/train_model.py` script fine-tunes the `distilbert-base-uncased` model on the labeled data from the `labeled_queries` table.
    - The fine-tuned model is saved to the `model/fine_tuned_model/` directory.

3. **Inference and Prediction:**
    - The `predict.py` script loads the fine-tuned model from the `model/fine_tuned_model/` directory.
    - It provides a function to classify new text queries.

4. **Main Application & UI:**
    - The `app.py` script orchestrates the entire process, running a FastAPI server.
    - It fetches unlabeled queries from the `unlabeled_queries` table.
    - It uses the fine-tuned model to classify these queries.
    - It writes the classified queries (with their predicted categories) into the `labeled_queries` table.
    - It serves a web interface at the `/ui` endpoint for interactive classification.

Model Evaluation

The model's performance is evaluated using the `evaluate.py` script.

- **Accuracy:** The model achieves an overall accuracy of **85%** on the test set.
- **Classification Report:** The report shows strong performance across the four categories:

Category	Precision	Recall	F1-Score
cancellation	1.00	0.25	0.40
return	0.75	1.00	0.86
shipping	0.83	1.00	0.91
warranty	1.00	1.00	1.00

- **Insights:**
    - The model shows perfect precision and recall for the `warranty` category.
    - It has perfect precision for the `cancellation` category, though with lower recall, indicating it is very confident when it predicts `cancellation`, but it may miss some.

To run the evaluation, you can execute the script directly:

```
python evaluate.py
```

Web Interface (UI)

This project includes a simple web interface for interactive text classification.

- **File:** `index.html`
- **Framework:** The UI is served by a FastAPI backend from `app.py`.
- **Functionality:** Users can enter text into a text area, and upon submission, the predicted category is displayed. This is achieved by making an API call to the `/predict/` endpoint.

How to Start This Project

Prerequisites

- Python 3.8+
- MySQL Server

1. Clone the Repository

```
git clone <repository-url>
cd text_classifier_project
```

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Set up MySQL Database

1. Make sure your MySQL server is running.
2. Update the database connection details (host, user, password) in `db/create_db.py`, `db/fetch_queries.py`, and `db/write_predictions.py`.

4. Run the Application

To run the end-to-end pipeline and the web UI, execute the main application script:

```
python app.py
```

This will start the FastAPI server.

- The API will be accessible at `http://127.0.0.1:8000`.
- The user interface will be available at `http://127.0.0.1:8000/ui`.