**Named Entity Recognition (NER) Model Report**

**Part 1: Data Preprocessing**

**Dataset Used**

* **CoNLL-2003 Named Entity Recognition Dataset** was selected for training the model due to its well-structured format and widespread usage in benchmarking NER models.

**Preprocessing Steps**

1. **Data Loading:**
   * Loaded the dataset using ‘Datasets’ module and examined the structure for text and entity labels.
2. **Tokenization:**
   * Used Hugging Face's tokenizer (AutoTokenizer) for BERT/RoBERTa to split text into tokens.
3. **Label Encoding:**
   * Encoded labels using integer mappings for better model compatibility.
4. **Data Splitting:**
   * Split the data into **80% training** and **20% testing**.

**Part 2: Model Selection & Optimization**

**Model Choice**

* **BERT (Bidirectional Encoder Representations from Transformers)** was selected due to its strong contextual understanding.

**Training Pipeline**

* Used Hugging Face's pipeline() to simplify the model creation.
* Fine-tuned the BERT NER model on the preprocessed dataset.

**Model Evaluation**

* Achieved high precision (0.94), recall (0.94), and F1-score (0.94) using metrics from seqeval.
* Performance improved after adjusting hyperparameters like:
  + **Batch Size:** 16
  + **Learning Rate:** 3e-5
  + **Epochs:** 3

**Model Saving**

* Saved the fine-tuned model and tokenizer.

**Part 3: Deployment**

**API Development**

* Used **Flask** for lightweight and efficient API deployment.
* Implemented a /predict endpoint that:
  + Accepts JSON input with a "text" field.
  + Returns identified entities and their confidence scores.

**Authentication**

* Added HTTP Basic Authentication with username-password verification to restrict access to authorized users.

**Streamlit UI Integration**

Streamlit was integrated with the NER API to improve user experience.

**Dockerization**

1. **Dockerfile Setup:**
   * Defined base image (python:3.11-slim).
   * Installed necessary dependencies from requirements.txt.
   * Set app.py as the entry point for the application for both Flask API and Streamlit UI.
   * To Build and Run docker containers , please use the below command-
   * **docker-compose up –build**

**Access the Apps**

* **Flask API** → <http://localhost:8000>
* **Streamlit UI** → http://localhost:8501

**Part 4: API Usage Guide**

**API Endpoint**

* **URL:** http://localhost:8000/predict

**Authentication**

* **Username:** admin
* **Password:** password

**Sample Request (Postman)**

**Request Body:**

{

"text": "Barack Obama was born in Hawaii and worked at Microsoft."

}

**Response:**

{

"entities": [

{"text": "Barack", "label": "LABEL\_1", "score": 0.9961},

{"text": "Obama", "label": "LABEL\_2", "score": 0.9966},

{"text": "Hawaii", "label": "LABEL\_5", "score": 0.9987},

{"text": "Microsoft", "label": "LABEL\_3", "score": 0.997}

]

}