Project Report: A Multi-Task AI Microservice

Author: Abhinav Dev V M **Course:** Artificial Intelligence.

1. Executive Summary

This project demonstrates the successful creation and deployment of a lightweight, multi-task Artificial Intelligence service. The service is built as a Python Flask API that operationalizes three distinct, pre-trained Natural Language Processing (NLP) models from the Hugging Face Transformers library. The API exposes endpoints for performing Sentiment Analysis, advanced Zero-Shot Classification, and Named Entity Recognition (NER) on user-provided text. The result is a powerful, flexible, and easy-to-use web service that transforms complex AI capabilities into simple, accessible tools, demonstrating a practical application of MLOps (Machine Learning Operations).

2. Introduction & Objectives

2.1 Problem Statement

State-of-the-art AI models, while incredibly powerful, are often large, complex, and resource-intensive. For developers or simple applications, accessing their capabilities presents a significant barrier. The "last mile" of AI implementation—making the model usable in a real-world product—is a critical challenge.

2.2 Project Objectives

The primary objective of this project was to bridge this gap by:

1. **Selecting** and loading multiple pre-trained NLP models.
2. **Wrapping** these models in a simple, high-performance web server (an API).
3. **Exposing** each AI task as a distinct, easy-to-use endpoint.
4. **Demonstrating** the ability to perform three different, sophisticated AI tasks (Sentiment, Classification, and Extraction) in real-time, using simple HTTP GET requests.

3. Methodology

The methodology for this project was centered on **model selection** and **API implementation**.

3.1 Core Technologies

- **Python:** The core programming language used for logic and server development.
- **Flask:** A lightweight Python web framework used to create the API server and route incoming requests to the correct AI functions.
- **Hugging Face transformers:** The key AI library used to download and run state-of-the-art, pre-trained models. The pipeline() function was used to simplify the complex tokenization and inference process into a single step.

3.2 AI Models and Tasks

Three distinct AI tasks and models were chosen to demonstrate a breadth of NLP capabilities.

1. Task: Sentiment Analysis

- **Model:** distilbert-base-uncased-finetuned-sst-2-english

- **Method:** This model is a "distilled" (smaller, faster) version of the BERT model. It was fine-tuned on the Stanford Sentiment Treebank (SST-2) dataset, making it highly accurate for simple binary (POSITIVE/NEGATIVE) sentiment classification. It provides a fast, reliable baseline for text analysis.

2. Task: Zero-Shot Classification

- **Model:** facebook/bart-large-mnli
- **Method:** This is a significantly more advanced AI technique. This model, based on the BART architecture and trained on the Multi-Genre Natural Language Inference (MNLI) dataset, can classify text into *any* set of labels provided at runtime. It does this without being explicitly trained on those labels. It works by re-framing the text and a candidate label as a "hypothesis" and "premise," determining if the text "entails" the label. This provides incredible flexibility.

3. Task: Named Entity Recognition (NER)

- **Model:** dbmdz/bert-large-cased-finetuned-conll03-english
- **Method:** This is a classic "information extraction" task. This large BERT model was fine-tuned on the CoNLL-2003 dataset to find and categorize specific entities in text. It can identify words and group them into categories such as Persons (PER), Organizations (ORG), and Locations (LOC). This demonstrates the AI's ability to extract structured data from unstructured text.

3.3 API Implementation

The Flask application was structured to load all three models into memory upon startup. It then defines three unique endpoints:

- /analyze for Sentiment Analysis
- /classify for Zero-Shot Classification
- /extract for Named Entity Recognition

Each endpoint accepts GET requests and reads input data from URL query parameters (e.g., ?text=... and &labels=...). The service then routes this data to the correct AI pipeline and returns the model's output in a clean, machine-readable JSON format.

4. Results & Demonstration

The AI service was successfully deployed on a local server (http://127.0.0.1:5000). The results obtained from testing each endpoint are as follows:

Result 1: Sentiment Analysis (/analyze)

- **Input URL:** .../analyze?text=This project was surprisingly easy and powerful.
- **JSON Output:**
- {
- "label": "POSITIVE",
- "score": 0.99988

- }
- **Observation:** The model correctly identified the positive sentiment with very high confidence.

Result 2: Zero-Shot Classification (/classify)

- **Input URL:** [127.0.0.1:5000 /classify?text=Apple just announced the new M5 chip&labels=technology,sports,cooking]
- **JSON Output:**


```
{
  "labels": [
    "technology",
    "sports",
    "cooking"
  ],
  "scores": [
    0.9892957210540771,
    0.006294487975537777,
    0.004409788642078638
  ],
  "sequence": "Apple just announced the new M5 chip"
}
```


- **Observation:** The model correctly identified "technology" as the most relevant label from the user-provided list, demonstrating its flexibility.

Result 3: Named Entity Recognition (/extract)

- **Input URL:** .../extract?text=My name is Rohan and I work for Google in India.
- **JSON Output:**
- [
- {
- "entity_group": "PER", "score": 0.999, "word": "Rohan", "start": 11, "end": 16
- },
- {
- "entity_group": "ORG", "score": 0.998, "word": "Google", "start": 33, "end": 39
- },
- {
- "entity_group": "LOC", "score": 0.999, "word": "India", "start": 43, "end": 48
- }
-]

- **Observation:** The model successfully parsed the sentence and extracted three distinct entities, correctly classifying them as a Person, Organization, and Location.

5. Conclusion

This project successfully achieved its objective of building a multi-task AI microservice. It proves that complex, state-of-the-art NLP models can be readily operationalized and served via a simple, scalable web API. The project demonstrates a practical understanding of key AI concepts (Sentiment, Classification, NER) and the MLOps principles required to integrate them into real-world applications.

Future Work:

- **Frontend:** A simple HTML/JavaScript frontend could be built to provide a user-friendly interface for the API.
- **Containerization:** The app could be containerized using Docker for easy deployment to any cloud platform.
- **Add Models:** The service could be easily extended with more models, such as text summarization or translation.

GitHub Repo link for reference: https://github.com/abhinavdev369/MiniAI.git