Named Entity Recognition (NER) Project

1. Project Overview

This project is a web-based **Named Entity Recognition (NER)** application designed to identify and classify entities such as persons, organizations, locations, and miscellaneous types from user-provided text. The backend is built using **Flask**, while the frontend interface is created with **Streamlit**. The application uses the pre-trained model **`dslim/bert-base-NER`** from Hugging Face's Transformers library.

---

2. Technical Architecture

Component	Technology
Model	`dslim/bert-base-NER`
Backend	Flask REST API
Frontend	Streamlit
Processing	Hugging Face Transformers
Communication	HTTP POST (JSON payload)

3. Backend (Flask) - Detailed Explanation

Module Imports

```
from flask import Flask, request, jsonify
from transformers import pipeline, AutoTokenizer
from flask_cors import CORS
import re
```

- `Flask`: Creates the web server and API endpoints.
- `transformers`: Loads and uses the NER pipeline.
- `AutoTokenizer`: Automatically loads the correct tokenizer for the model.
- `CORS`: Enables Cross-Origin Resource Sharing so the frontend can communicate with the API.
- `re`: Regular expression module used for text preprocessing.

Application Initialization and Model Loading

```
app = Flask(__name__)
CORS(app)
```

- Initializes the Flask app.
- Enables CORS to allow external frontend requests.

```
model_name = "dslim/bert-base-NER"
tokenizer = AutoTokenizer.from_pretrained(model_name)
ner = pipeline(
    "ner",
    model=model_name,
    tokenizer=tokenizer,
    aggregation_strategy="average"
)
```

- Loads the NER model and tokenizer.
- `aggregation_strategy="average"` combines multi-word entities into a single one.

Text Preprocessing

```
def preprocess_text(text):
    ...
```

- Removes extra whitespaces.
- Adds spacing before punctuation to help the model identify sentence boundaries better.

Hyphenated Entity Merging

```
def merge_hyphenated_entities(entities):
    ...
```

- Joins adjacent entities that are likely meant to be one (e.g., first and last names split).
- Ensures entity continuity and improves display results.

The `/analyze` API Endpoint

```
@app.route("/analyze", methods=["POST"])
def analyze_text():
    ...
```

- Accepts a POST request with the user text.
- Applies preprocessing, runs the model, merges entities, and returns results as JSON.

Server Execution

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=False)
```

- Starts the API on port `5000`, accessible externally for local testing or deployment.

---

4. Frontend (Streamlit) - Detailed Explanation

Visual Setup

```
GROUP_COLORS, GROUP_ICONS, group_labels
```

- Defines icons, labels, and color schemes for each entity type:
  - PER (Person), ORG (Organization), LOC (Location), MISC (Miscellaneous)

Reset Functionality

```
def reset_form():
    ...
```

- Clears all session states including the input and results.

UI Components

- `st.title`: Application header.
- `st.expander`: Project usage guide and tips.
- `st.text_area`: Text input field.
- `st.button`: "Analyze" and "Clear" buttons.

Analyzing the Input

```
if analyze_clicked:
    ...
    response = requests.post(...)
```

- When "Analyze" is clicked:
  - Sends a POST request to the Flask backend with the input text.
  - Displays loading spinner and processes returned results.

Displaying Results

```
entities_dict = {}
...
st.markdown("### Analysis Results")
...
```

- Entities are grouped by type.
- Each group is displayed in a color-coded summary card.
- Entity details (word, confidence score, position) are shown using styled HTML blocks.

Full Entity List (Expandable)

- Displays every detected entity with:
  - Its label
  - Start-end position in text
  - Confidence as a progress bar

5. How It Works (Step-by-Step)

1. User enters text into the input field.
2. Clicks “Analyze”.
3. Text is sent to Flask API → Preprocessed → Analyzed by BERT model.
4. Results are sent back and shown in Streamlit UI.
5. “Clear” button resets the interface.

6. Known Limitations

- Recommended input length: **~500 characters**
- Less common entity types might have **low accuracy**
- Classification is **context-sensitive**
- Hyphenated or joined words may need **manual formatting**

7. Conclusion

This project demonstrates a modern NLP pipeline that combines state-of-the-art **BERT-based NER** models with a clean and interactive user interface. By leveraging Hugging Face Transformers and Streamlit, the application offers both technical accuracy and accessibility for non-developers.

The modular architecture allows for:

- Easy extension with additional models or languages.
- Deployment in education, news analysis, resume parsing, chatbots, and more.
- Further improvements such as multilingual support, model switching, or file uploads.

In conclusion, this NER application not only showcases the power of transformer-based NLP but also provides a practical, reusable tool for real-world entity recognition tasks.