Experiment No.7
Implement Named Entity Recognizer for the given text input.
Date of Performance:
Date of Submission:

Aim: Implement Named Entity Recognizer for the given text input.

Objective: Understand the importance of NER in NLP and Implement NER.

Theory:

The named entity recognition (NER) is one of the most data preprocessing task. It involves the identification of key information in the text and classification into a set of predefined categories. An entity is basically the thing that is consistently talked about or refer to in the text.

NER is the form of NLP.

At its core, NLP is just a two-step process, below are the two steps that are involved:

- Detecting the entities from the text
- Classifying them into different categories

Some of the categories that are the most important architecture in NER such that:

- Person
- Organization
- Place/ location

Other common tasks include classifying of the following:

- date/time.
- expression
- Numeral measurement (money, percent, weight, etc)
- E-mail address

Ambiguity in NE

For a person, the category definition is intuitively quite clear, but for computers, there is some ambiguity in classification. Let's look at some ambiguous example:

England (Organisation) won the 2019 world cup vs The 2019 world cup happened in England(Location).

Washington(Location) is the capital of the US vs The first president of the US was Washington(Person).

Implementation:

```
import spacy

nlp = spacy.load("en_core_web_sm")

text = "Apple Inc. is a company based in Cupertino, California. John works for Google in Mountain View."

doc = nlp(text)

named_entities = []

def extract_named_entities(doc):
    entities = []
    current_entity = None
    for token in doc:
        if token.ent_type_:
            if current_entity and token.ent_type_ == current_entity[1]:
                current_entity = (current_entity[0] + " " + token.text, token.ent_type_)
            else:
```

```
            if current_entity:
                entities.append(current_entity)
            current_entity = (token.text, token.ent_type_)
        else:
            if current_entity:
                entities.append(current_entity)
            current_entity = None
    if current_entity:
        entities.append(current_entity)
    return entities
named_entities = extract_named_entities(doc)
for entity, label in named_entities:
    print(f"Entity: {entity}, Label: {label}")
```

```
Entity: Apple Inc., Label: ORG
Entity: Cupertino, Label: GPE
Entity: California, Label: GPE
Entity: John, Label: PERSON
Entity: Google, Label: ORG
Entity: Mountain View, Label: GPE
```

Conclusion:

Comment on the results and identify the correctly named entities if not correct or for words which ought to be recognized but weren't.

The script uses SpaCy's `en_core_web_sm` to extract named entities from the text:

Expected Correct Entities:

1. Apple Inc. (ORG)
2. Cupertino (GPE)
3. California (GPE)
4. John (PERSON)
5. Google (ORG)
6. Mountain View (GPE)

Issues:

- Consecutive entity tokens may be incorrectly grouped, potentially splitting multi-word entities like "Apple Inc." or "Mountain View."
- Entity detection may occasionally miss certain names or locations.

Recommendations:

SpaCy's built-in `doc.ents` handles multi-word entities better. Using it would simplify and improve accuracy.