Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)
Lab Manual
Sub: Advanced Computational Linguistics **Year/Sem: BTech/VII**

<u>Experiment No 6</u>

Name – Jhanvi Parekh

Sapid – 60009210033

Batch – D11

Aim: Implement Information Retrieval for performing Name Entity Recognition, Relation Extraction and template Creation.

Theory:

Information extraction (IE) in natural language processing (NLP) refers to the process of automatically extracting structured information from unstructured or semi-structured text data. The goal is to convert the textual data into a more organized and structured form that can be easily analysed and processed by machines. Information extraction typically involves identifying entities, relationships, and attributes within the text.

Here are the key components and steps involved in information extraction: **1.**

Named Entity Recognition (NER):

- Named Entity Recognition is the task of identifying and classifying specific entities mentioned in the text, such as names of people, organizations, locations, dates, percentages, etc.
- For example, in the sentence "Apple Inc. was founded by Steve Jobs in Cupertino," NER would identify "Apple Inc." as an organization, "Steve Jobs" as a person, and "Cupertino" as a location.

2. Relation Extraction:

- Relation extraction involves identifying and classifying relationships between entities in the text.
- For example, from the sentence "Barack Obama was born in Hawaii," relation extraction would determine that "Barack Obama" is related to "Hawaii" through the "born_in" relationship.

3. Event Extraction:

- Event extraction focuses on identifying events and their participants from the text.

- For instance, from the sentence "Apple unveiled its new product," event extraction would identify "Apple" as the entity performing the action "unveiled" and "new product" as the event.

4. Attribute Extraction:

- Attribute extraction involves identifying descriptive attributes associated with entities.
- In the sentence "The Eiffel Tower is a tall iron structure in Paris," attribute extraction would identify "tall" and "iron" as attributes of "Eiffel Tower."

5. Dependency Parsing:

- Dependency parsing analyzes the grammatical structure of a sentence to identify how words depend on each other.
- It helps in understanding the relationships between words and their roles in the sentence.

6. Coreference Resolution:

- Coreference resolution identifies when different expressions in the text refer to the same entity.
- For instance, resolving that "he" in the sentence "John is an engineer. He builds bridges" refers to the same person.

7. Template Filling:

- Template filling involves populating predefined templates with extracted information to create structured data.
- For instance, from the sentence "Microsoft was founded by Bill Gates in 1975," the template "ORG was founded by PERSON in YEAR" can be filled to create structured data.

8. Information Integration:

- After extracting relevant information, it can be integrated with existing knowledge bases or databases to enhance the understanding of relationships and context.

Information extraction has applications in various fields, including text mining, data enrichment, question answering, sentiment analysis, knowledge graph construction, and more. It often involves a combination of rule-based methods, machine learning techniques, and domain-specific knowledge to accurately extract and structure information from textual data.

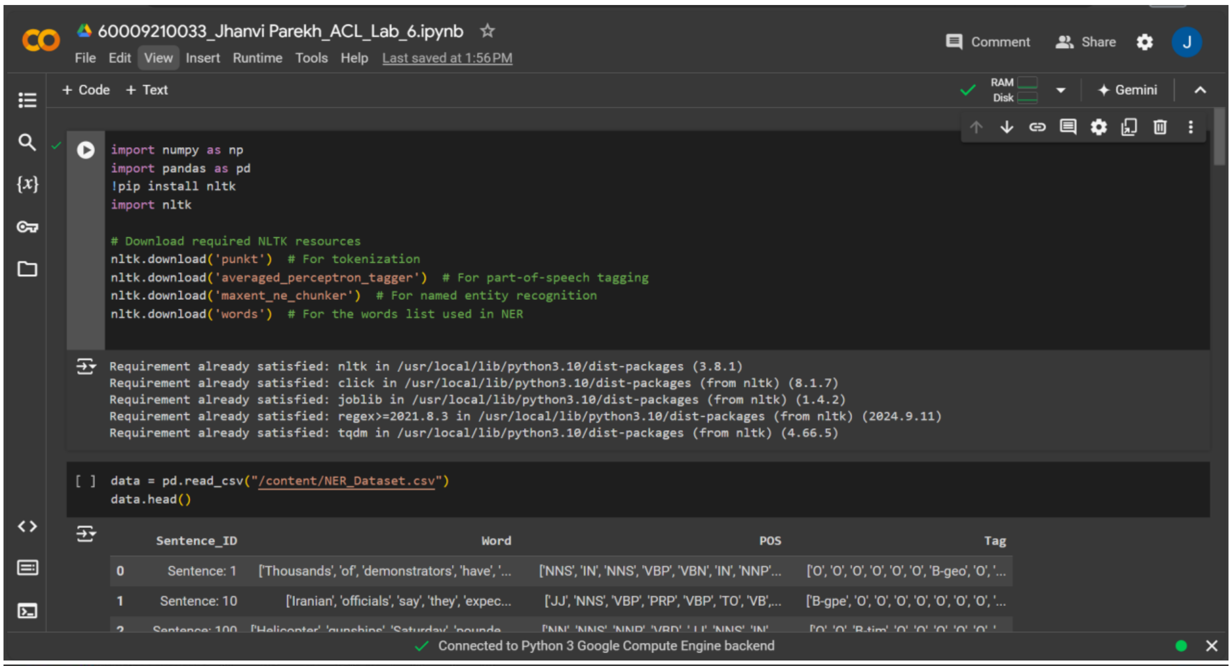
Lab Assignment to be performed

Dataset: Select any Text Paragraph containing relationships between various entities

Exercise 1: Perform Name Entity Recognition using NLTK

Exercise 2: Perform Relationship Extraction using NLTK

Exercise 3: Create a Template for the given text using NLTK using Information Extraction.
Colab Link:
https://colab.research.google.com/drive/12l9UQAIHQnCMbhZ-VjxkpyeKYbCjvJ-z?usp=sharing


60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini

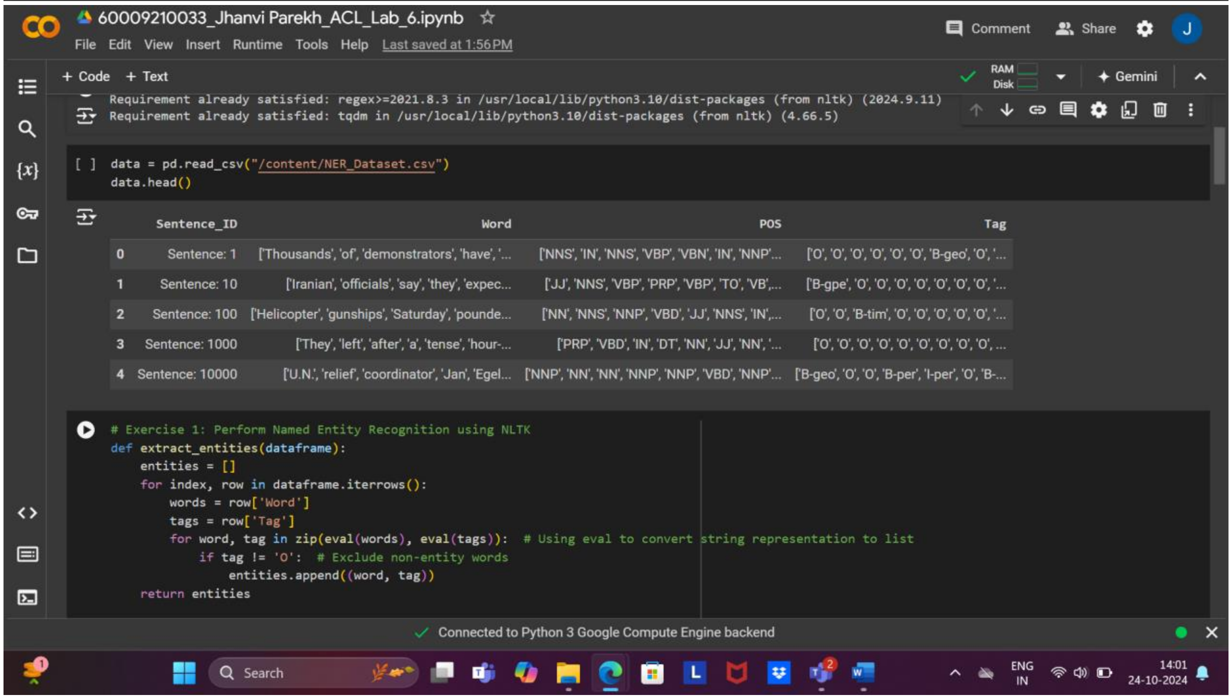
```
import numpy as np
import pandas as pd
!pip install nltk
import nltk

# Download required NLTK resources
nltk.download('punkt')  # For tokenization
nltk.download('averaged_perceptron_tagger')  # For part-of-speech tagging
nltk.download('maxent_ne_chunker')  # For named entity recognition
nltk.download('words')  # For the words list used in NER
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.5)
```

```
[ ] data = pd.read_csv("/content/NER_Dataset.csv")
    data.head()
```

	Sentence_ID	Word	POS	Tag
0	Sentence: 1	['Thousands', 'of', 'demonstrators', 'have', '...	['NNS', 'IN', 'NNS', 'VBP', 'VBN', 'IN', 'NNP'...	['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', '...
1	Sentence: 10	['Iranian', 'officials', 'say', 'they', 'expec...	['JJ', 'NNS', 'VBP', 'PRP', 'VBP', 'TO', 'VB',...	['B-gpe', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '...

Connected to Python 3 Google Compute Engine backend

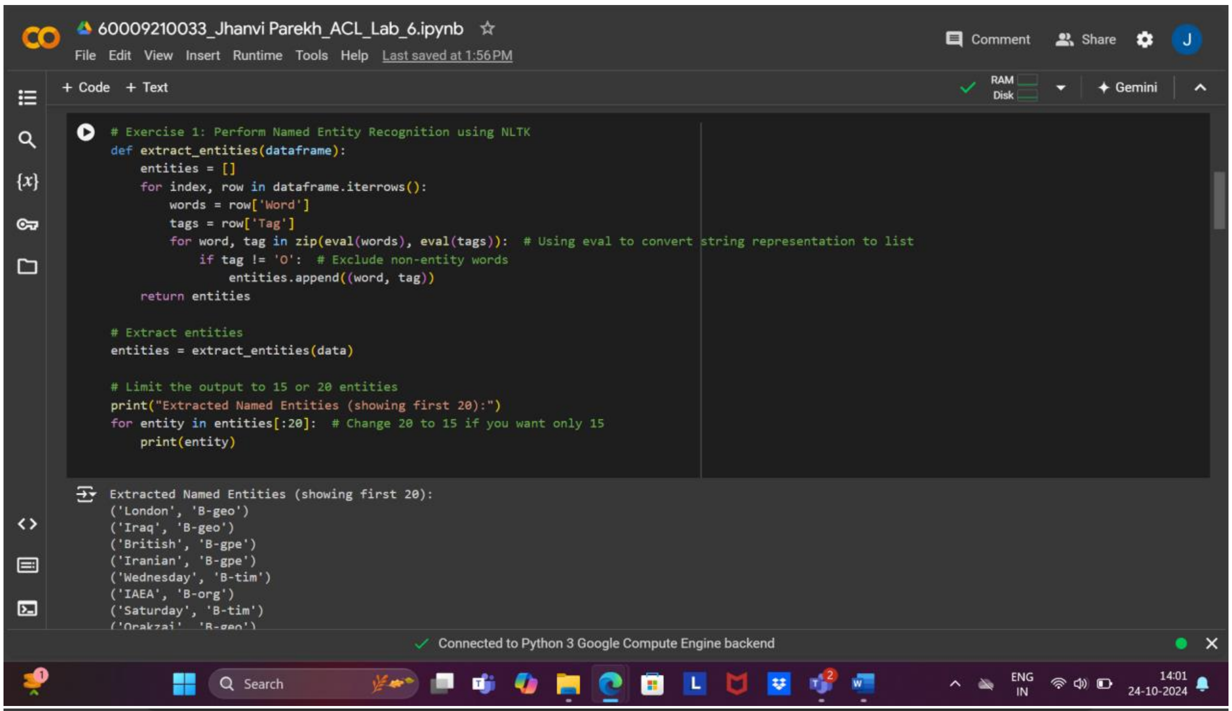

60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini

```
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.5)
```

```
[ ] data = pd.read_csv("/content/NER_Dataset.csv")
    data.head()
```

	Sentence_ID	Word	POS	Tag
0	Sentence: 1	['Thousands', 'of', 'demonstrators', 'have', '...	['NNS', 'IN', 'NNS', 'VBP', 'VBN', 'IN', 'NNP'...	['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', '...
1	Sentence: 10	['Iranian', 'officials', 'say', 'they', 'expec...	['JJ', 'NNS', 'VBP', 'PRP', 'VBP', 'TO', 'VB',...	['B-gpe', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '...
2	Sentence: 100	['Helicopter', 'gunships', 'Saturday', 'pounde...	['NN', 'NNS', 'NNP', 'VBD', 'JJ', 'NNS', 'IN',...	['O', 'O', 'B-tim', 'O', 'O', 'O', 'O', 'O', '...
3	Sentence: 1000	['They', 'left', 'after', 'a', 'tense', 'hour-...	['PRP', 'VBD', 'IN', 'DT', 'NN', 'JJ', 'NN', '...	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', ...
4	Sentence: 10000	['U.N.', 'relief', 'coordinator', 'Jan', 'Egel...	['NNP', 'NN', 'NN', 'NNP', 'NNP', 'VBD', 'NNP'...	['B-geo', 'O', 'O', 'B-per', 'I-per', 'O', 'B-...

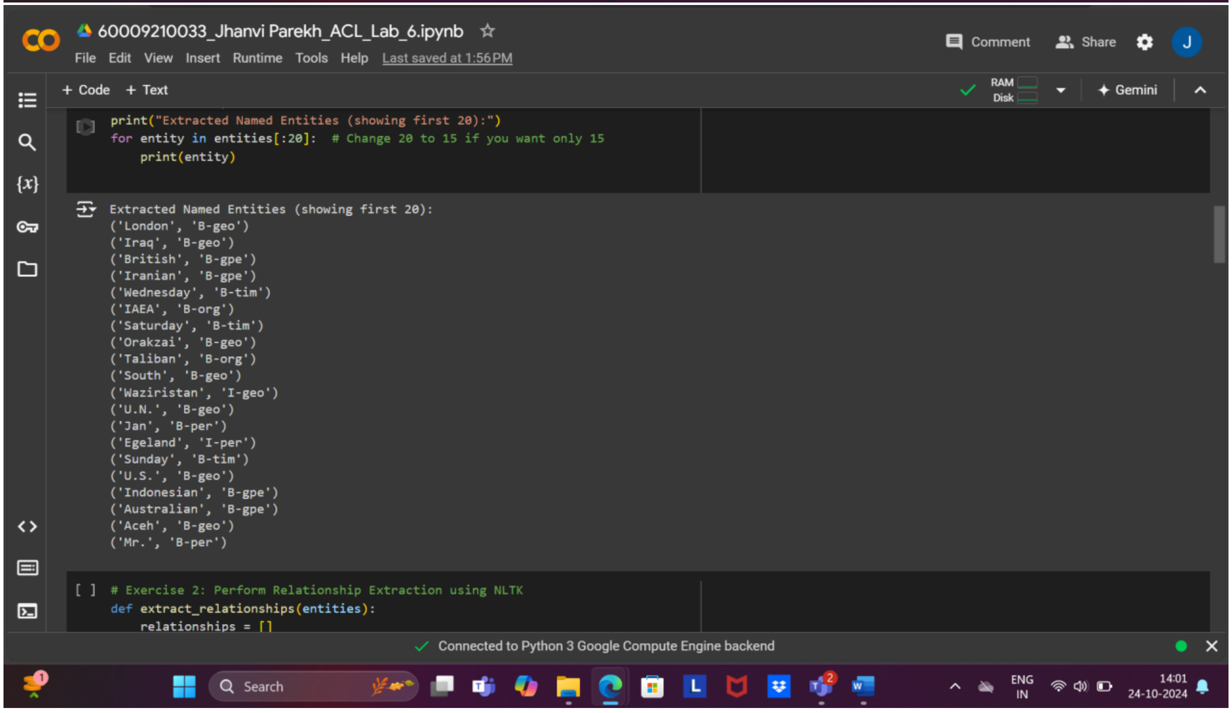
```
# Exercise 1: Perform Named Entity Recognition using NLTK
def extract_entities(dataframe):
    entities = []
    for index, row in dataframe.iterrows():
        words = row['Word']
        tags = row['Tag']
        for word, tag in zip(eval(words), eval(tags)):  # Using eval to convert string representation to list
            if tag != 'O':  # Exclude non-entity words
                entities.append((word, tag))
    return entities
```


60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini

```
# Exercise 1: Perform Named Entity Recognition using NLTK
def extract_entities(dataframe):
    entities = []
    for index, row in dataframe.iterrows():
        words = row['Word']
        tags = row['Tag']
        for word, tag in zip(eval(words), eval(tags)):  # Using eval to convert string representation to list
            if tag != 'O':  # Exclude non-entity words
                entities.append((word, tag))
    return entities

# Extract entities
entities = extract_entities(data)

# Limit the output to 15 or 20 entities
print("Extracted Named Entities (showing first 20):")
for entity in entities[:20]:  # Change 20 to 15 if you want only 15
    print(entity)
```

```
Extracted Named Entities (showing first 20):
('London', 'B-geo')
('Iraq', 'B-geo')
('British', 'B-gpe')
('Iranian', 'B-gpe')
('Wednesday', 'B-tim')
('IAEA', 'B-org')
('Saturday', 'B-tim')
('Orakzai', 'B-geo')
```

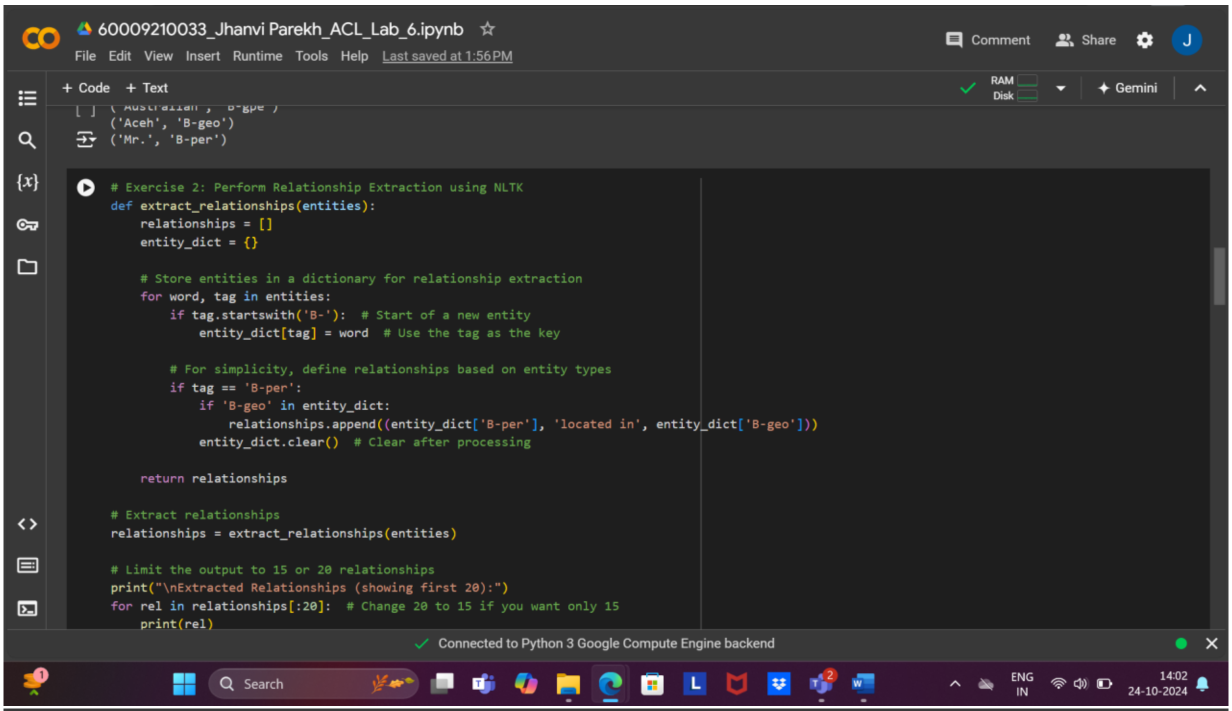
Connected to Python 3 Google Compute Engine backend
Search
ENG IN
14:01
24-10-2024


60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini

```
print("Extracted Named Entities (showing first 20):")
for entity in entities[:20]:  # Change 20 to 15 if you want only 15
    print(entity)
```

```
Extracted Named Entities (showing first 20):
('London', 'B-geo')
('Iraq', 'B-geo')
('British', 'B-gpe')
('Iranian', 'B-gpe')
('Wednesday', 'B-tim')
('IAEA', 'B-org')
('Saturday', 'B-tim')
('Orakzai', 'B-geo')
('Taliban', 'B-org')
('South', 'B-geo')
('Waziristan', 'I-geo')
('U.N.', 'B-geo')
('Jan', 'B-per')
('Egeland', 'I-per')
('Sunday', 'B-tim')
('U.S.', 'B-geo')
('Indonesian', 'B-gpe')
('Australian', 'B-gpe')
('Aceh', 'B-geo')
('Mr.', 'B-per')
```

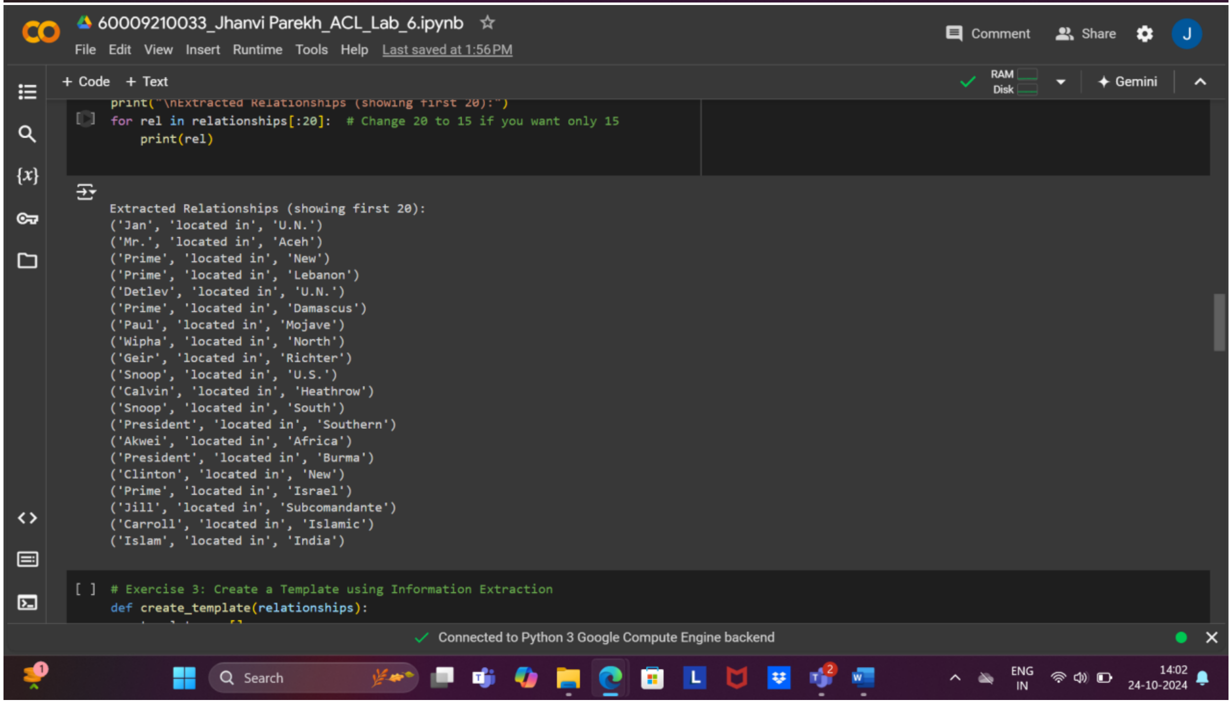
```
[ ] # Exercise 2: Perform Relationship Extraction using NLTK
    def extract_relationships(entities):
        relationships = []
```



```
('Aceh', 'B-geo')
('Mr.', 'B-per')
```

```python
# Exercise 2: Perform Relationship Extraction using NLTK
def extract_relationships(entities):
    relationships = []
    entity_dict = {}

    # Store entities in a dictionary for relationship extraction
    for word, tag in entities:
        if tag.startswith('B-'):  # Start of a new entity
            entity_dict[tag] = word  # Use the tag as the key

        # For simplicity, define relationships based on entity types
        if tag == 'B-per':
            if 'B-geo' in entity_dict:
                relationships.append((entity_dict['B-per'], 'located in', entity_dict['B-geo']))
            entity_dict.clear()  # Clear after processing

    return relationships

# Extract relationships
relationships = extract_relationships(entities)

# Limit the output to 15 or 20 relationships
print("\nExtracted Relationships (showing first 20):")
for rel in relationships[:20]:  # Change 20 to 15 if you want only 15
    print(rel)
```

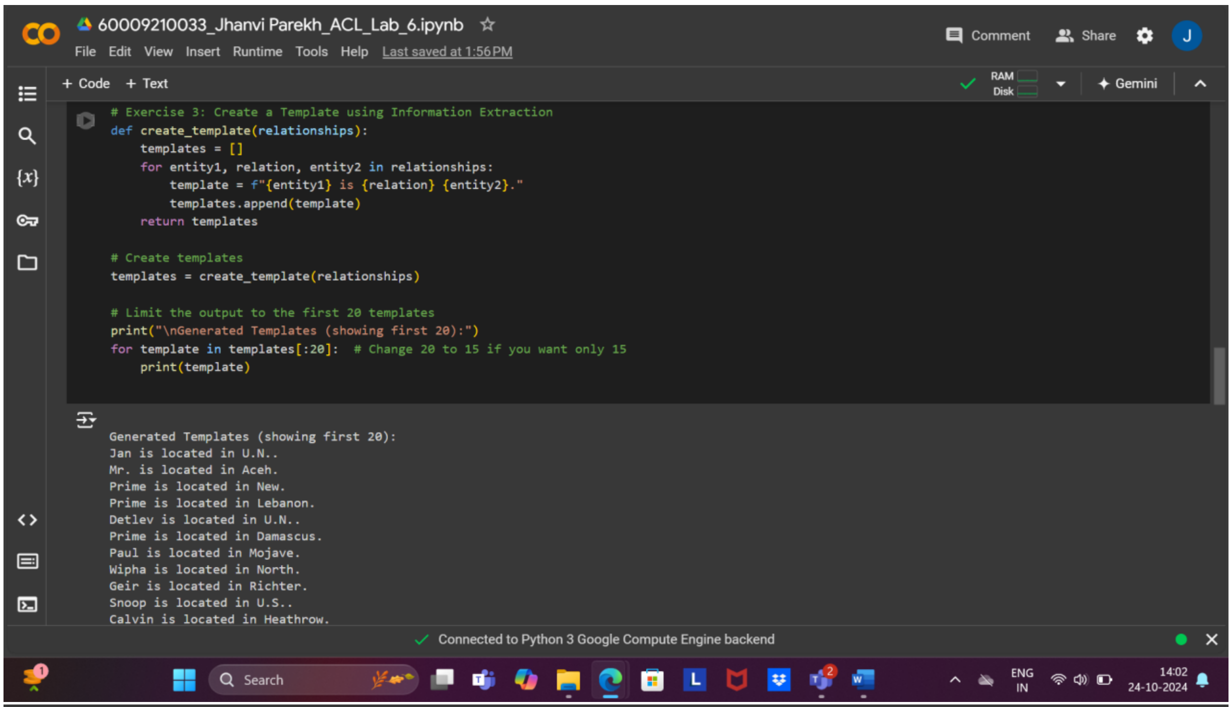
Connected to Python 3 Google Compute Engine backend



```
Extracted Relationships (showing first 20):
('Jan', 'located in', 'U.N.')
('Mr.', 'located in', 'Aceh')
('Prime', 'located in', 'New')
('Prime', 'located in', 'Lebanon')
('Detlev', 'located in', 'U.N.')
('Prime', 'located in', 'Damascus')
('Paul', 'located in', 'Mojave')
('Wipha', 'located in', 'North')
('Geir', 'located in', 'Richter')
('Snoop', 'located in', 'U.S.')
('Calvin', 'located in', 'Heathrow')
('Snoop', 'located in', 'South')
('President', 'located in', 'Southern')
('Akwei', 'located in', 'Africa')
('President', 'located in', 'Burma')
('Clinton', 'located in', 'New')
('Prime', 'located in', 'Israel')
('Jill', 'located in', 'Subcomandante')
('Carroll', 'located in', 'Islamic')
('Islam', 'located in', 'India')
```

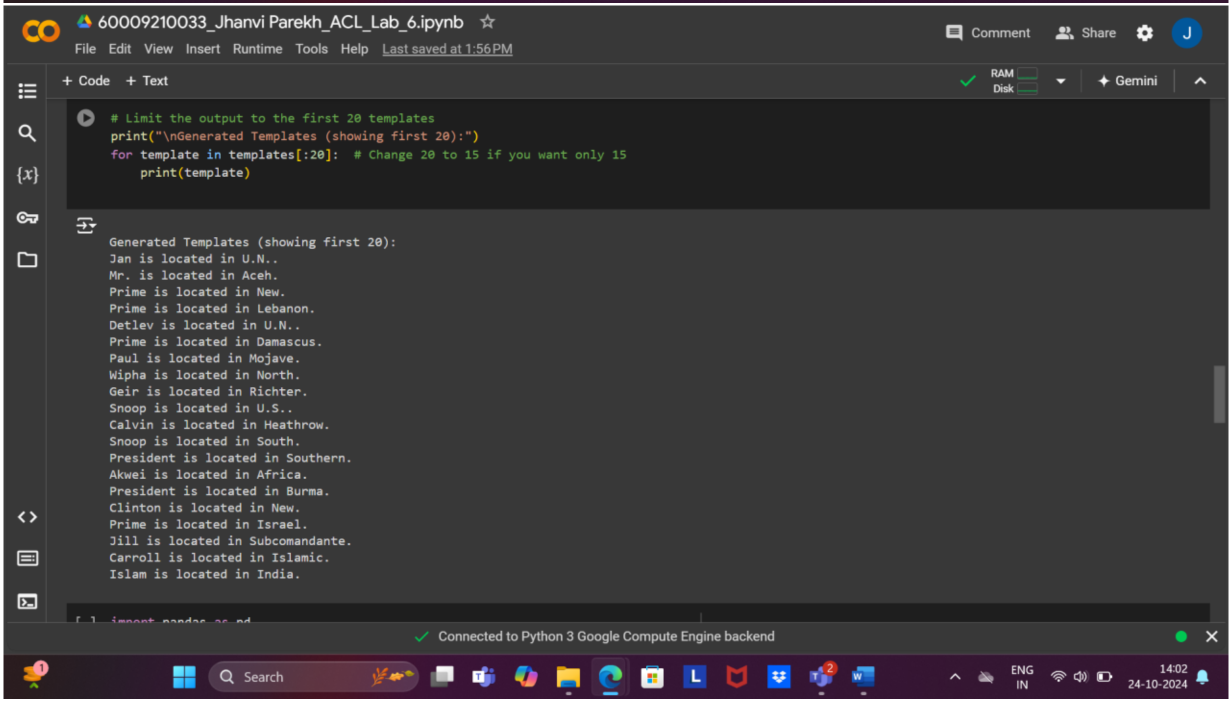
```python
# Exercise 3: Create a Template using Information Extraction
def create_template(relationships):
```

Connected to Python 3 Google Compute Engine backend



```
# Exercise 3: Create a Template using Information Extraction
def create_template(relationships):
    templates = []
    for entity1, relation, entity2 in relationships:
        template = f"{entity1} is {relation} {entity2}."
        templates.append(template)
    return templates

# Create templates
templates = create_template(relationships)

# Limit the output to the first 20 templates
print("\nGenerated Templates (showing first 20):")
for template in templates[:20]:  # Change 20 to 15 if you want only 15
    print(template)
```

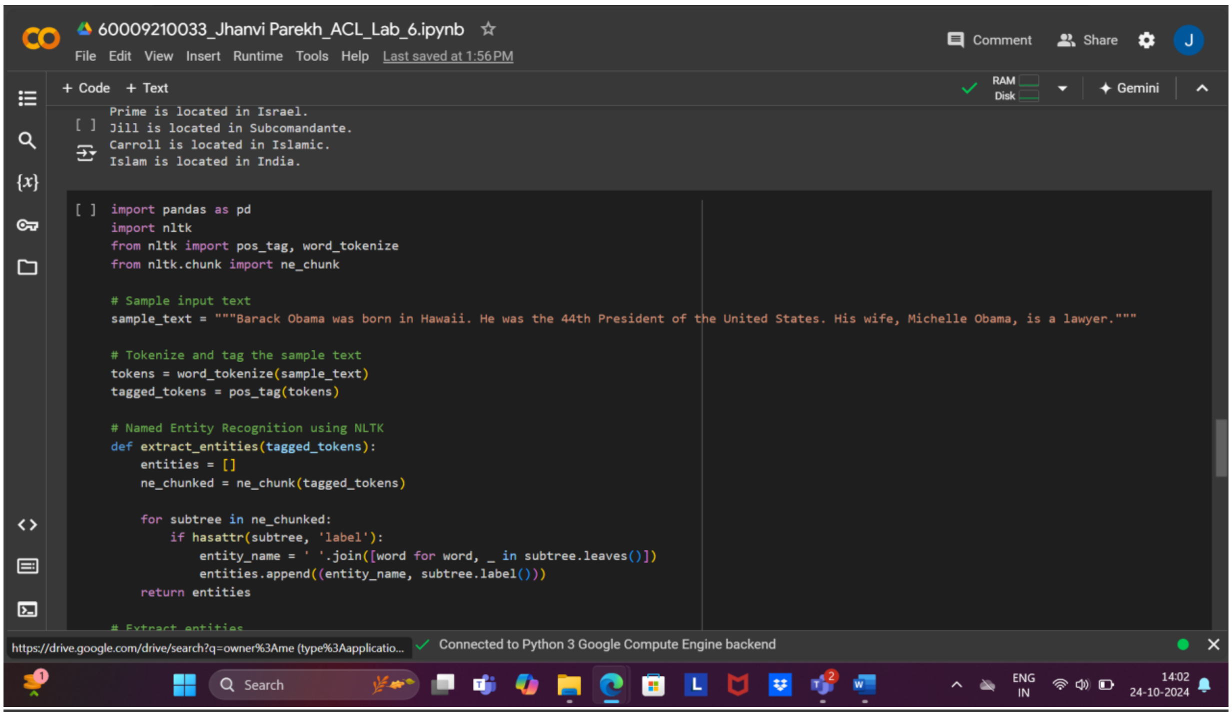
```
Generated Templates (showing first 20):
Jan is located in U.N..
Mr. is located in Aceh.
Prime is located in New.
Prime is located in Lebanon.
Detlev is located in U.N..
Prime is located in Damascus.
Paul is located in Mojave.
Wipha is located in North.
Geir is located in Richter.
Snoop is located in U.S..
Calvin is located in Heathrow.
```



```
# Limit the output to the first 20 templates
print("\nGenerated Templates (showing first 20):")
for template in templates[:20]:  # Change 20 to 15 if you want only 15
    print(template)
```

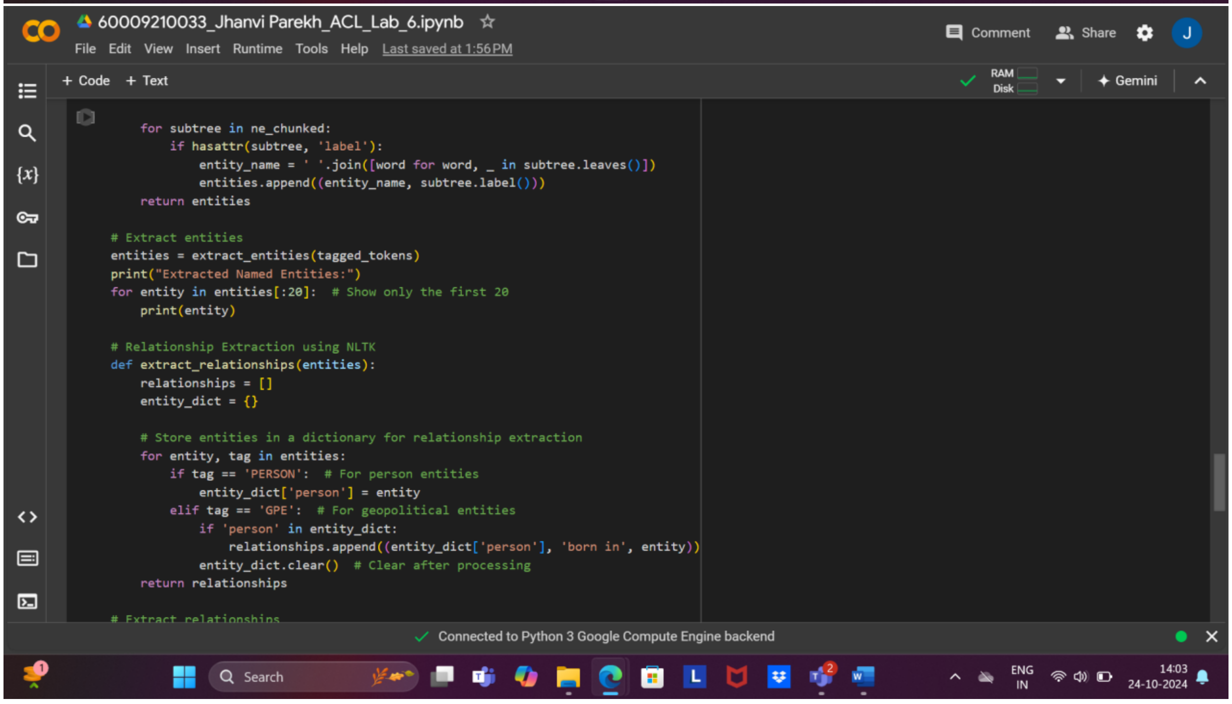
```
Generated Templates (showing first 20):
Jan is located in U.N..
Mr. is located in Aceh.
Prime is located in New.
Prime is located in Lebanon.
Detlev is located in U.N..
Prime is located in Damascus.
Paul is located in Mojave.
Wipha is located in North.
Geir is located in Richter.
Snoop is located in U.S..
Calvin is located in Heathrow.
Snoop is located in South.
President is located in Southern.
Akwei is located in Africa.
President is located in Burma.
Clinton is located in New.
Prime is located in Israel.
Jill is located in Subcomandante.
Carroll is located in Islamic.
Islam is located in India.
```

```
[ ] import pandas as pd
```

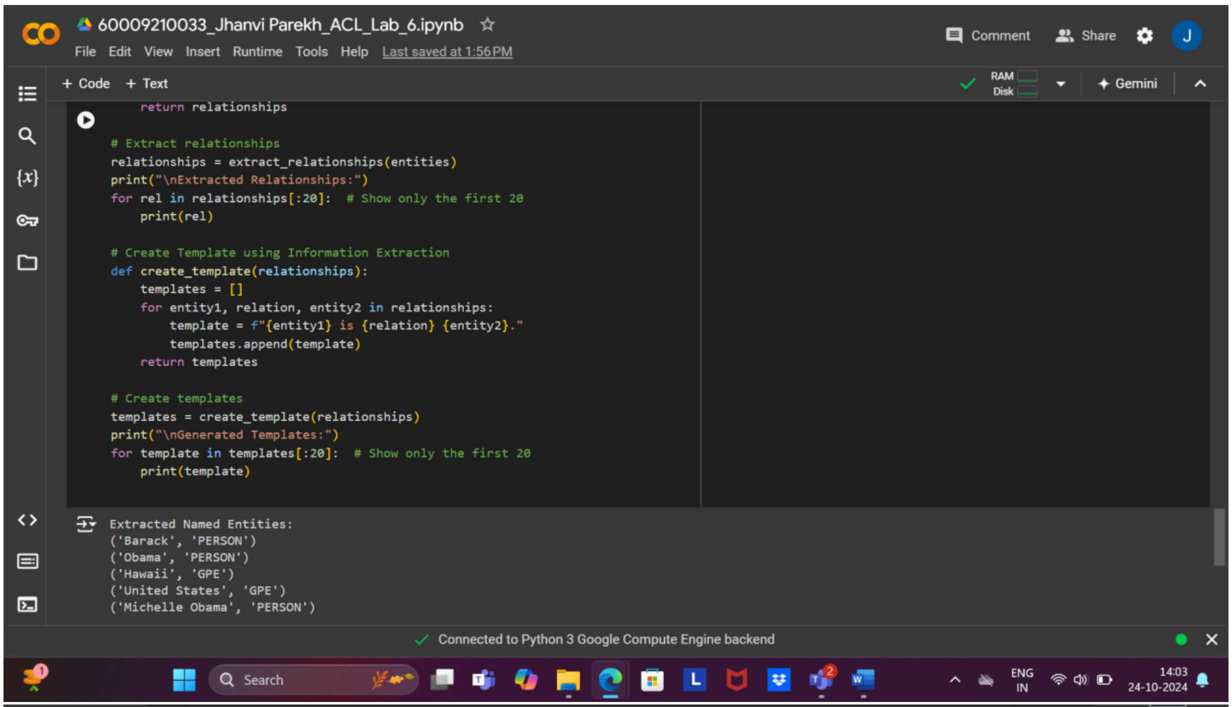

60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini
Connected to Python 3 Google Compute Engine backend
14:02 24-10-2024

```
Prime is located in Israel.
Jill is located in Subcomandante.
Carroll is located in Islamic.
Islam is located in India.
```

```python
import pandas as pd
import nltk
from nltk import pos_tag, word_tokenize
from nltk.chunk import ne_chunk

# Sample input text
sample_text = """Barack Obama was born in Hawaii. He was the 44th President of the United States. His wife, Michelle Obama, is a lawyer."""

# Tokenize and tag the sample text
tokens = word_tokenize(sample_text)
tagged_tokens = pos_tag(tokens)

# Named Entity Recognition using NLTK
def extract_entities(tagged_tokens):
    entities = []
    ne_chunked = ne_chunk(tagged_tokens)

    for subtree in ne_chunked:
        if hasattr(subtree, 'label'):
            entity_name = ' '.join([word for word, _ in subtree.leaves()])
            entities.append((entity_name, subtree.label()))
    return entities
```

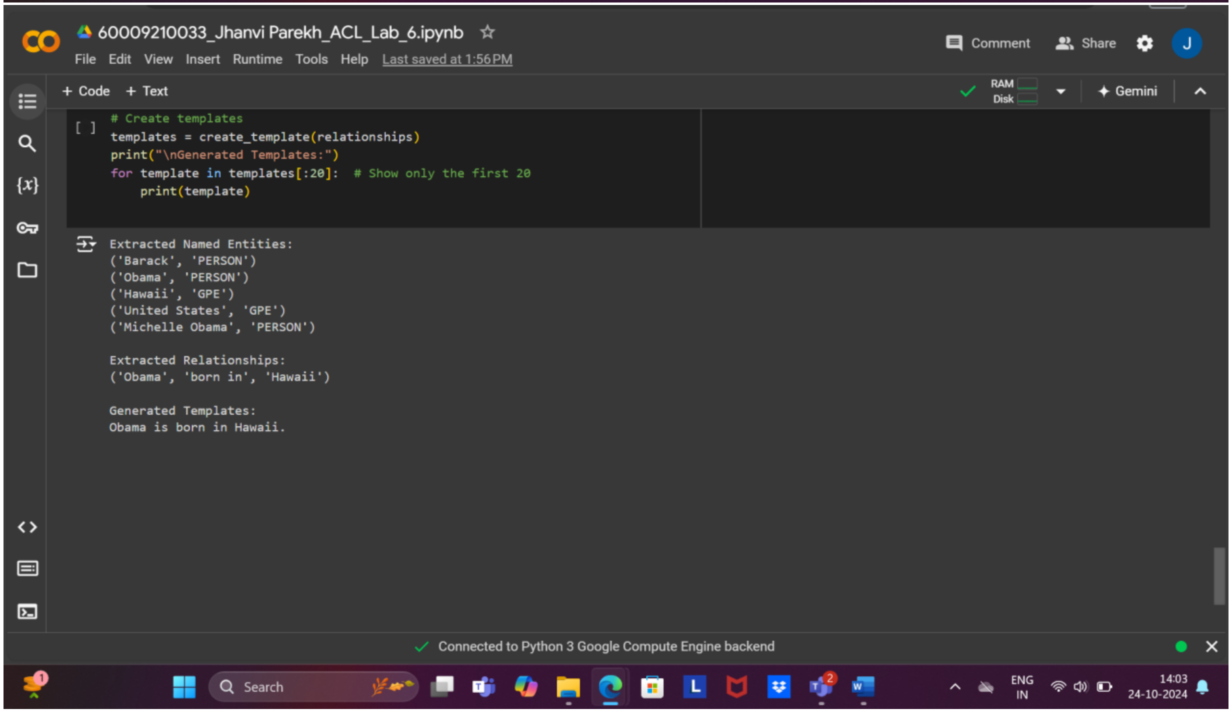

60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini
Connected to Python 3 Google Compute Engine backend
14:03 24-10-2024

```python
    for subtree in ne_chunked:
        if hasattr(subtree, 'label'):
            entity_name = ' '.join([word for word, _ in subtree.leaves()])
            entities.append((entity_name, subtree.label()))
    return entities

# Extract entities
entities = extract_entities(tagged_tokens)
print("Extracted Named Entities:")
for entity in entities[:20]:  # Show only the first 20
    print(entity)

# Relationship Extraction using NLTK
def extract_relationships(entities):
    relationships = []
    entity_dict = {}

    # Store entities in a dictionary for relationship extraction
    for entity, tag in entities:
        if tag == 'PERSON':  # For person entities
            entity_dict['person'] = entity
        elif tag == 'GPE':  # For geopolitical entities
            if 'person' in entity_dict:
                relationships.append((entity_dict['person'], 'born in', entity))
            entity_dict.clear()  # Clear after processing
    return relationships
```


60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini
Connected to Python 3 Google Compute Engine backend
14:03 24-10-2024

```
    return relationships

# Extract relationships
relationships = extract_relationships(entities)
print("\nExtracted Relationships:")
for rel in relationships[:20]:  # Show only the first 20
    print(rel)

# Create Template using Information Extraction
def create_template(relationships):
    templates = []
    for entity1, relation, entity2 in relationships:
        template = f"{entity1} is {relation} {entity2}."
        templates.append(template)
    return templates

# Create templates
templates = create_template(relationships)
print("\nGenerated Templates:")
for template in templates[:20]:  # Show only the first 20
    print(template)
```

```
Extracted Named Entities:
('Barack', 'PERSON')
('Obama', 'PERSON')
('Hawaii', 'GPE')
('United States', 'GPE')
('Michelle Obama', 'PERSON')
```


60009210033_Jhanvi Parekh_ACL_Lab_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:56 PM
Comment Share
+ Code + Text
RAM Disk
Gemini
Connected to Python 3 Google Compute Engine backend
14:03 24-10-2024

```
# Create templates
templates = create_template(relationships)
print("\nGenerated Templates:")
for template in templates[:20]:  # Show only the first 20
    print(template)
```

```
Extracted Named Entities:
('Barack', 'PERSON')
('Obama', 'PERSON')
('Hawaii', 'GPE')
('United States', 'GPE')
('Michelle Obama', 'PERSON')

Extracted Relationships:
('Obama', 'born in', 'Hawaii')

Generated Templates:
Obama is born in Hawaii.
```