Virtual Internship - 6.0

Milestone-2

Group-4

Vaibhavi Nimba Patil

Preetham Aditya

Vardhan

Ravi Kumar Rathlavat

- **Dataset collection and Preprocessing & NER -** Vaibhavi Patil
- **RE -** Ravi Kumar Rathlavat
- **Knowledge Graph -** Vardhan
- **Semantic Search -** Preetham Aditya
- Combined All these into a single working pipeline

NER

```python
import pandas as pd
import spacy
from tqdm import tqdm

data = pd.read_csv("KMapFinal.csv")
nlp = spacy.load("en_core_web_sm")
tqdm.pandas()
def extract_entities(text):
    doc = nlp(str(text))
    return [(ent.text, ent.label_) for ent in doc.ents]

data["entities"] = data["article"].progress_apply(extract_entities)

data.to_csv("entities_extracted.csv", index=False)

print("NER extraction complete — saved as 'entities_extracted.csv'")
```

Relation Extraction

```python
import pandas as pd
import spacy
from tqdm import tqdm

# Load dataset
data = pd.read_csv("KMapFinal.csv")
nlp = spacy.load("en_core_web_sm")
tqdm.pandas()
# Extract relations
def extract_relations(text):
    doc = nlp(str(text))
    triples = []
    for sent in doc.sents:
        subject, relation, object_ = None, None, None
        for token in sent:
            if "subj" in token.dep_:
                subject = token.text
            if token.pos_ == "VERB":
                relation = token.lemma_
            if "obj" in token.dep_:
                object_ = token.text
        if subject and relation and object_:
            triples.append((subject, relation, object_))
    return triples
data["relations"] = data["article"].progress_apply(lambda x: extract_relations(str(x)))
data.to_csv("relations_output.csv", index=False)
print("✅ Relation extraction complete — saved as 'relations_output.csv'")
```

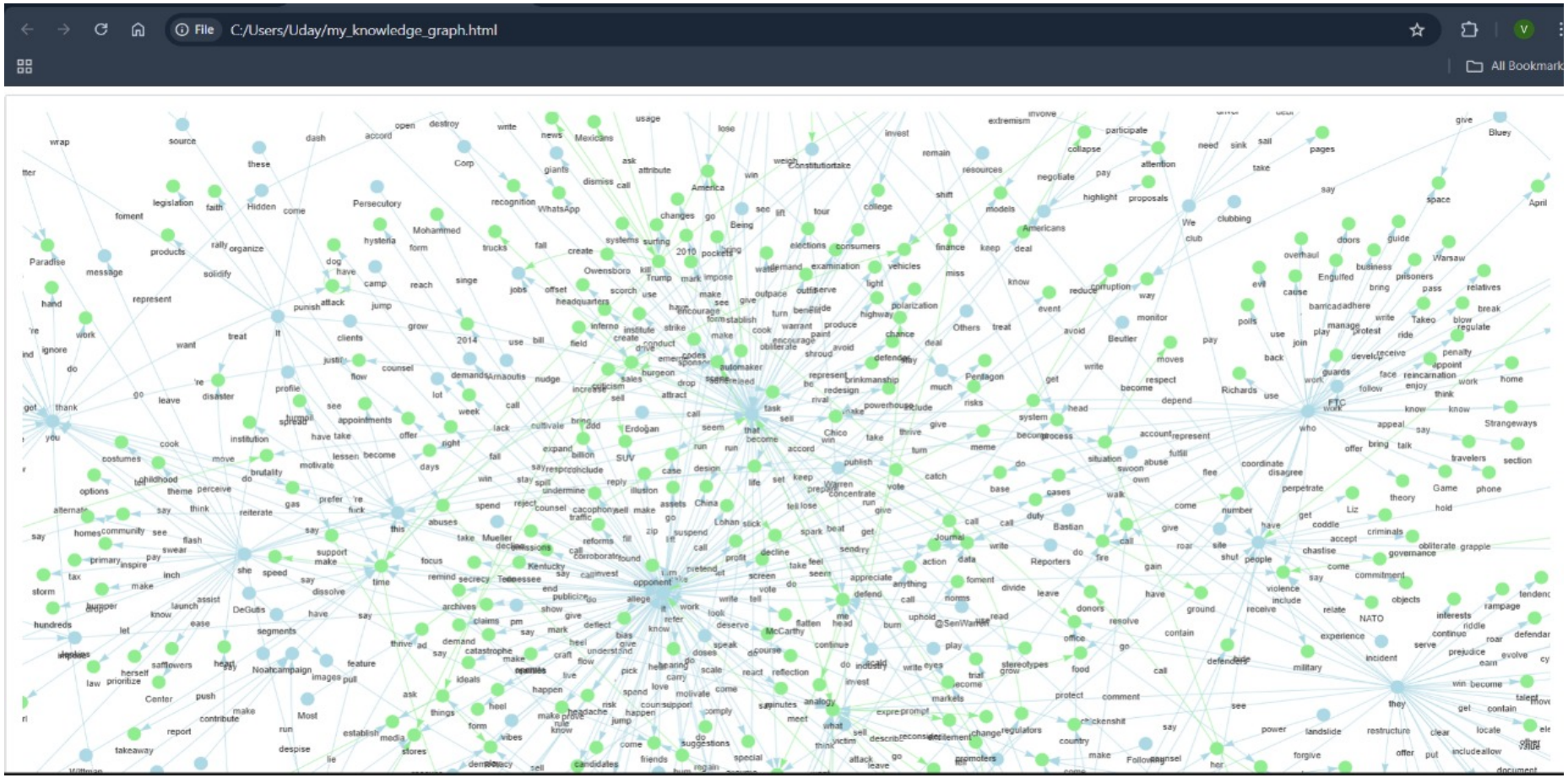
output

```
rgukt-basar@ravi-kumar-rathlavath:~/MileStone-Final-Task/new$ python3 entity_triplet_extractor.py
Loaded 5 sentences from dataset.
First few rows of data:
                                          text
0  Albert Einstein developed the theory of relati...
1        Marie Curie discovered radium and polonium.
2        Isaac Newton formulated the laws of motion.
3   Charles Darwin proposed the theory of evolution.
4            Galileo Galilei improved the telescope.

Entities saved to: /home/rgukt-basar/MileStone-Final-Task/new/entities_extracted.csv
Triplets saved to: /home/rgukt-basar/MileStone-Final-Task/new/triplets_extracted.csv

Sample Entities:
Sentence: Albert Einstein developed the theory of relativity.
Entity: Albert Einstein (PERSON)

Sentence: Marie Curie discovered radium and polonium.
Entity: Marie Curie (PERSON)

Sentence: Isaac Newton formulated the laws of motion.
Entity: Isaac Newton (PERSON)

Sentence: Charles Darwin proposed the theory of evolution.
Entity: Charles Darwin (PERSON)

Sentence: Galileo Galilei improved the telescope.
Entity: Galileo Galilei (PRODUCT)


Sample Triplets:
Einstein develop relativity
Curie discover radium
Newton formulate motion
Darwin propose evolution
Galilei improve telescope
```

Knowledge Graph

```python
import pandas as pd
import networkx as nx
from pyvis.network import Network
import webbrowser
# Load relation data
data = pd.read_csv("relations_output.csv").head(50)
# Extract triples
triples_list = []
for cell in data.iloc[:, -1]:
    try:
        tuples = eval(cell)
        triples_list.extend(tuples)
    except:
        pass
print(f"✅ Extracted {len(triples_list)} triples from relations_output.csv")
# Build directed graph
G = nx.DiGraph()
for subj, rel, obj in triples_list:
    G.add_node(subj, color="lightblue")
    G.add_node(obj, color="lightgreen")
    G.add_edge(subj, obj, label=rel)
print(f"✅ Graph built with {len(G.nodes())} nodes and {len(G.edges())} edges.\n")
# Visualize
net = Network(height="750px", width="100%", directed=True, notebook=False)
net.from_nx(G)
net.toggle_physics(True)
net.write_html("my_knowledge_graph.html")

print("🎉 Interactive Knowledge Graph saved as 'my_knowledge_graph.html'")
webbrowser.open("my_knowledge_graph.html")
```

output



Semantic Search

```python
import pandas as pd
from sentence_transformers import SentenceTransformer, util

# Load dataset
data = pd.read_csv("KMapFinal.csv")
# Load model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Encode sentences
sentences = data["article"].dropna().tolist()
embeddings = model.encode(sentences, convert_to_tensor=True)

# Query input
query = input("\n💬 Enter a search query (e.g., 'AI technology'): ")
query_embedding = model.encode(query, convert_to_tensor=True)

# Compute similarity
cosine_scores = util.cos_sim(query_embedding, embeddings)
top_results = cosine_scores[0].argsort(descending=True)[:5]

# Display results
print("\n🔎 Top 5 Semantic Matches:")
for idx in top_results:
    print(f"➞ {sentences[idx]}  |  Score: {float(cosine_scores[0][idx]):.3f}")

print("\n✅ Semantic search completed successfully!")
```