SPO Knowledge Graph for Information Extraction

Asmita Guha

BTech IT 3rd Yr, MIT Manipal, Karnataka, India

Asmita1.mitmpl2023@learner.manipal.edu

BACKGROUND

This project outlines a simplified approach to automatically building a subject–predicate–object (SPO) knowledge graph from natural language text. Initially, the model scrapes unstructured textual data from web sources using requests and BeautifulSoup. The retrieved content is preprocessed into sentence fragments by splitting at conjunctions to remove complex or compound sentences. Each fragment is analyzed with spaCy's dependency parser to isolate grammatical relations, extracting subject, verb (predicate), and object triples that convey facts. These triples are converted into structured form and can be visually represented as directed graphs using networkx and matplotlib which highlight nodes as entities and edges as semantic relations. The produced knowledge graph serves as a compressed semantic depiction of the unstructured text, which enhances information retrieval, natural language understanding, and knowledge inference for further applications. The entire system is built in Python and can easily be extended to incorporate coreference resolution, named entity linking, or real-time text streams from online pages.

PYTHON PACKAGES REQUIRED

Package	Purpose
spaCy	NLP engine for parsing, POS tagging, dependency analysis
en_core_web_sm	SpaCy's small English language model for syntactic analysis
requests	Download HTML content from web URLs
beautifulsoup4	Parse and extract readable text from web pages
re (built-in)	Regular expressions for sentence splitting
networkx	Create and manage the structure of the knowledge graph (nodes/edges)
matplotlib	Visualize the knowledge graph

DEVELOPMENTAL TOOLS AND ENVIRONMENT

Tool/Platform	Purpose

Python 3.8+	Core programming language
Spyder (Anaconda)	IDE for Python scripting with data science tooling support
Anaconda Navigator	GUI for managing environments and packages
Jupyter Notebook / IPython (optional)	For modular and interactive testing/debugging

OPTIONAL EXTENSIONS

Optional Tool / Lib	Purpose
Flask / Streamlit	Build a simple web app interface to enter URLs or view graphs live
PyGraphviz	For more advanced and styled graph visualizations (if needed)
scikit-learn / transformers	For later additions like semantic similarity or classification

OBJECTIVE

- Extract structured knowledge in the form of Subject–Predicate–Object (SPO) triples from unstructured natural language text.
- Input consists of sentences from diverse sources such as web articles, novels, or Wikipedia pages.
- The goal is to convert each sentence into one or more semantically meaningful SPO triples, for example:

Example:

Input: "The Earth revolves around the Sun and is home to many species."
Output:

Subject: Earth, Predicate: revolves around, Object: Sun
Subject: Earth, Predicate: is home to, Object: many species

- These triples serve as atomic facts that can be visualized as a Knowledge Graph, where:
- Nodes represent entities (subjects and objects)
- Edges represent predicates (relationships)

METHODOLOGY

1. Text Preprocessing

- Convert the input text to lowercase for uniform processing.
- Split compound and complex sentences into simple ones using punctuation marks and coordinating conjunctions (e.g., and, but, or).
- Remove or ignore non-informative punctuation (commas, quotation marks, etc.).
- Normalize whitespace and filter out sentences that are too short to be meaningful.

2. Syntactic Parsing

- Use an NLP engine (e.g., spaCy) to:
- Tokenize the text into words and punctuation.
- Perform Part-of-Speech (POS) tagging to identify nouns, verbs, prepositions, etc.
- Apply Dependency Parsing to identify grammatical roles such as subject (nsubj), object (dobj or pobj), auxiliary (aux), and root verb (ROOT).

3. SPO Triple Extraction

For each sentence:

- Identify the subject as the noun or pronoun with nsubj or nsubjpass dependency relation.
- Identify the predicate as the main verb (ROOT) and related auxiliary/modifier words (aux, xcomp, prep, etc.).
- Identify the object as the direct object (dobj), prepositional object (pobj), or attribute (attr).
- Construct the triple in the form: (Subject, Predicate, Object)

4. Post-processing (Optional but Recommended)

- Lemmatize words to their root form to standardize vocabulary (e.g., "revolves" → "revolve").
- Remove or exclude stopwords from predicates or objects for cleaner triples.
- Format triples into a clean, printable structure or export for use in graph visualization tools.
- Deduplicate redundant triples or merge semantically identical ones if necessary.

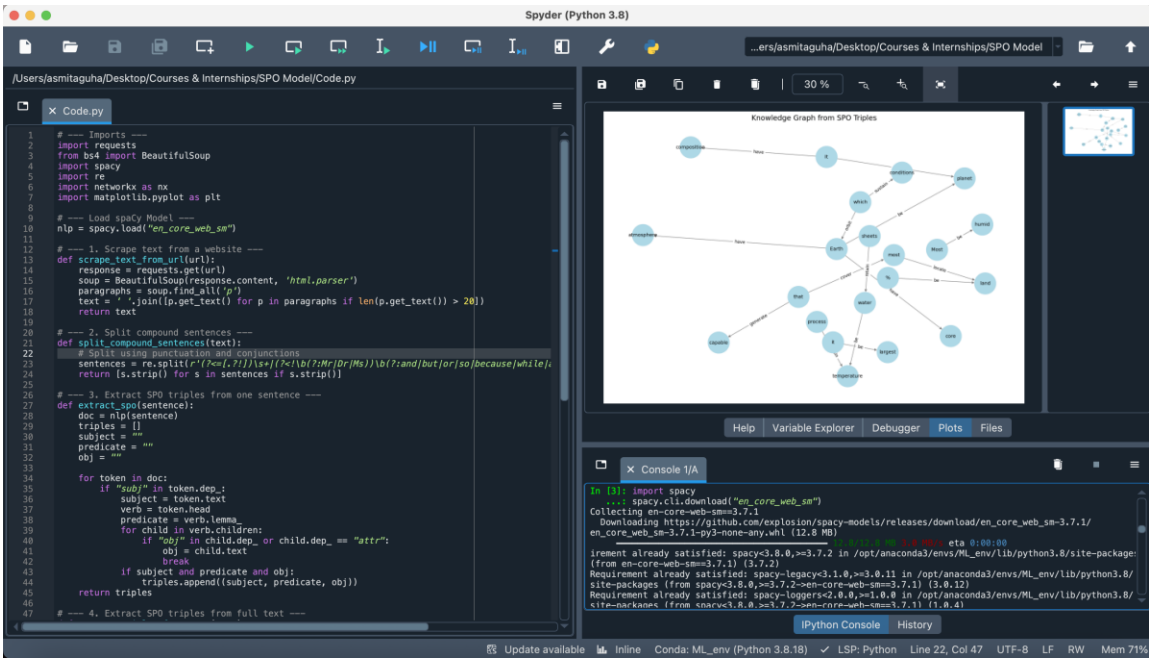
RESULTS OF CODE-1: TEXT EXTRACTION FROM WEBSITE

It is important to run the following in the iPython console before running the actual code

```
In [3]: import spacy
   ...: spacy.cli.download("en_core_web_sm")
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/
en_core_web_sm-3.7.1-py3-none-any.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 3.0 MB/s eta 0:00:00
irement already satisfied: spacy<3.8.0,>=3.7.2 in /opt/anaconda3/envs/ML_env/lib/python3.8/site-package
(from en-core-web-sm==3.7.1) (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /opt/anaconda3/envs/ML_env/lib/python3.8/
site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /opt/anaconda3/envs/ML_env/lib/python3.8/
site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.4)
```

IPython Console | History

Final Output with the knowledge graph



SPO Triples

```
In [4]: %runfile '/Users/asmitaguha/Desktop/Courses & Internships/SPO Model/Code.py' --wdir
Scraping content from: https://en.wikipedia.org/wiki/Earth

Scraped text length: 58353

Extracted SPO Triples:
Subject: Earth, Predicate: be, Object: planet
Subject: %, Predicate: be, Object: land
Subject: most, Predicate: locate, Object: land
Subject: Most, Predicate: be, Object: humid
Subject: sheets, Predicate: retain, Object: water
Subject: Earth, Predicate: have, Object: core
```

IPython Console | History

RESULTS OF CODE-2: TEXT EXTRACTION FROM NOVEL

Novel: 'A Christmas Carol' by Charles Dickens (we're using a free .txt file downloaded from Project Gutenberg)

The code has been designed keeping the following in mind:

- Skip Project Gutenberg metadata
- Filter meaningful sentences
- Extract more accurate SPO triples
- Print stats on how many triples were found
- Optional) Visualize the knowledge graph

The code takes longer to compile than the previous example because of a steep increase in the amount of text

Initially, the SPO triples are nonsensical
Hence, we update the code to keep only named entities as subjects/objects, remove personal pronouns and ignore sentences without named entities

But now another problem arises, very less valid SPO triples are extracted (33/1279 processed)

Improvements Added

1. Extracts multiple SPO triples per sentence
2. Allows both named entities and general nouns as subjects/objects
3. Optionally includes pronouns (configurable)
4. Still uses spaCy and visualizes a knowledge graph