Generic Knowledge Graph Generator

Tong Wu **2734542**

Leonardo Kuffo **2722539**

Yunshan Wang **2721621**

Zhaolin Fang **2725173**

I. Introduction

Entities in publicly available knowledge bases are skewed to famous entities. It is difficult to find entities from a very specific domain. Hence, we propose a framework to interactively generate and visualize a Knowledge Graph from scratch with entities and relationships from a specific domain. On this work, we are going to focus on analyzing books that tell stories. Using information extraction techniques we implemented a pipeline for automatically constructing a knowledge graph given a text from a book. At the end of the pipeline, we build and show a visual representation of the resulting knowledge graph.

II. Methodology

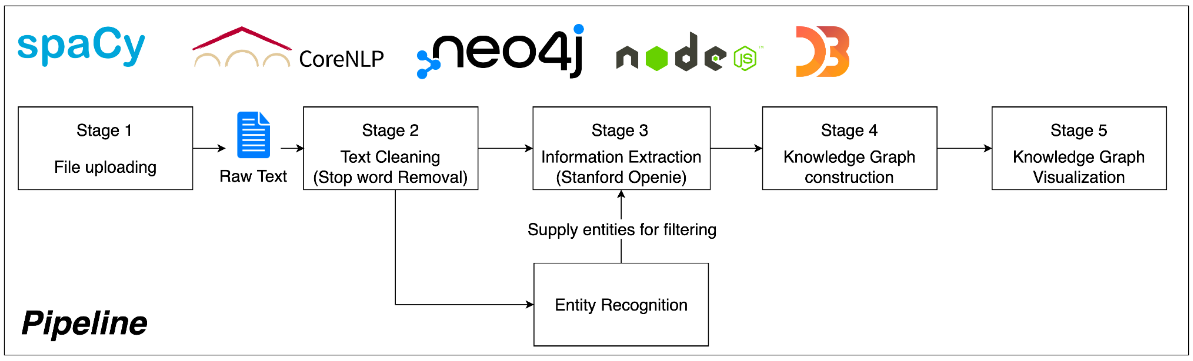

Figure 1. Generic Knowledge Graph Generator pipeline

Figure 1. shows the designed pipeline from the user input of information until the visualization of the generated Knowledge Graph. As a first step, the raw text of the book story must be entered in the pipeline. There are two ways of achieving this: The user uploads a file in our developed end-user app or a programmer executes the pipeline from their local environment.

Next, we apply an **Information Extraction pipeline** to the raw text. This pipeline starts with **removing the stop words** from the text to retrench the redundancy so that it makes the processing time as short as possible. When it comes to the **information extraction** stage, we choose to use the **Stanford Openie** since rule-based patterns are low recall and openie will improve efficiency. As the figure shows, we use openie combined with **entity recognition using Spacy**. The reason for this combination is that the result of stanford openie entity recognition contains too many types of entities and some of their relations are uninformative. However, we found that the relations between entities of "PERSON" type are more meaningful. Consequently we used spaCy to filter only the entities which were of "PERSON" type. Hecne, we constrained the result of the subject and object of the relation triples. Then we got the relation triples and transferred it to the next stage.

Finally, the refined triplets are stored in **a Neo4j database**. Neo4j[1] is a native graph database which uses the Cypher query language to add and obtain information from the database. The latter is perfect to build and query Knowledge Graphs. Our Neo4j database is part of the Neo4j free-tier sandbox environment. Using the Python driver of Neo4j we are able to execute Cypher queries to persist our triplets onto the database. For each entity we store the book to which it is related, the name of the entity and a unique identifier. All entities are linked to the same entity type (i.e. "Character"). Finally, directional relationships between entities are stored and our Knowledge Graph is complete.

III. End-User Application (Demo)

An end-user application was developed as a way for the user to interact with the existing knowledge graphs and upload new books to be analyzed. The application was built on Node.js + Express.js. The different endpoints on the back-end query to Neo4j to retrieve information from the Knowledge Graph. To execute the pipeline on uploaded books, the back-end spawns a Python child process that reads the raw text and performs the pipeline described in Figure 1. Once the pipeline is finished, the interface shows in an interactive way the entities and relationships from the uploaded book as a graph visualization built in D3.js[2]. In addition to this, the user has the possibility to search for specific entities in a book. The application is hosted freely on Heroku[3] and it is currently publicly available in: https://wdps-kg.herokuapp.com/. Unfortunately, due to storage limitations of our free-deployment environment and the huge size of some Stanford Openie dependencies, the feature that lets users upload their own books is not available in the live-demo.

IV. Results and Conclusions

We developed a pipeline which can generate and visualize a Knowledge Graph from scratch with entities and relationships from a specific text. Moreover, we developed a web interface to visualize the knowledge graph and the processed books. The largest book we processed using our pipeline was Harry Potter I which contained about 200K words. From this raw text data, we found 90 human characters and 353 relationships between the characters. Our pipeline took about 5 minutes to process this amount of words.

V. Limitations

Although we designed coreference resolution in our initial pipeline, we are not able to realize it due to technical incompatibility problems with spaCy, openie system and coreference libraries. Theoretically we could get more accurate results if we can realize it. In addition to this, we were not able to measure the correctness of our pipeline since we lack a baseline. Finally, due to time limitations, we were not able to compare Stanford openie results to other Information Extraction systems such as OpenNRE (Neural Relation Extraction toolkit) and DeepKE (Deep learning based knowledge extraction toolkit).

[1] https://neo4j.com/
[2] https://d3js.org/
[3] https://www.heroku.com/