Treks To Trips

By: Manan Rajdev and Posha Dave

Problem Domain and Goals

This project was focused on planning, building and deploying a knowledge graph for all the trails in the national parks of the United States. The knowledge graph includes information about each trail including the length, duration, elevation, route type, difficulty level and activities that can be performed there. It combines the trail properties with information about each national park such as inception point, scenic drive, best accessible time in the year, highest peak, etc. Besides this, it will also give information on whether the trail is dog or kid friendly for families. To build a trip centered around a trek, we have also collected and combined information about the species and nearby hotels about each national park. Currently, there is no knowledge graph that provides all this information on a single source. Through our knowledge graph, our goal is to provide a one stop experience where users can access all the necessary information needed to plan a trip. To further enhance the experience for the users, we have built a recommendation system that suggests trails that they would like based on their previous experiences. We have also built several dashboards where the users can interact with the data and gain some analytical insights about various trails, national parks, species and hotels.

Challenges Faced

Challenge 1: Dynamic Sites

We encountered our first challenge while scraping the Alltrails website. It is a dynamic site that has a restriction on their API calls and capped the number of pages that can be loaded at once. This forced us to obtain only 1000 page urls during one iteration. We navigated this situation by gathering urls for top 1000 trails from each state in the United States of America. This was done using Selenium and Beautiful Soup. We switched to Scrapy for extracting specific information for each trail.

Challenge 2: Blocking & String Similarities

The common attributes among all the websites like the State attribute, could not be used for blocking as a national park can be present in various states together. Therefore, first, we decided to separate out national parks, monuments and forests from each other and then create blocks based on the first names within these categories. Furthermore, due to typing errors, misspelled words and different representations of the same word, we had to use Jaro-Winkler similarity within the blocks. However, even after blocking and integration, there were many entities which were unliked. This was because there were different definitions of the same entity (for example, some national parks were labeled as national monuments). So, we had to perform entity linking on the entire batch without blocking just for the unmatched records.

Challenge 3: Integration

While conducting entity linking for the national parks, we had overlapping resources. Firstly, the hotel information was scraped from two different sources. This was managed by merging the hotels per national parks by performing string similarity and using the rent from the most updated website. Secondly, to provide a rating for each trail, we generated a weighted score based on favourite trails in each national park scraped from the parks expert website and the ratings of individual trails from the Alltrails website. This allowed us to give a more precise rating score.

Challenge 4: Creation of RDF Triples
The data we collected from the structured and unstructured sources contained a lot of attributes in the form of a list with non-uniform number of parameters within it. The Alltrails website has a lot of tags about activities, restrictions, water and snow presence which were scraped as list. For most of the tags, we clustered similar attributes and grouped them as individual properties. Furthermore, the common names in the structured source and the data scraped from the parks expert website contained parameters such as hotels, favorite trail and scenic drives were in the form of comma separated values. So, we had to experiment with Neo4J settings to handle multiple values.

Challenge 5: Dashboard Creation
We built an interactive dashboard on ElasticSearch and Kibana using our generated RDF triples files. One of the major challenges while working on the dashboards was the required input format. Kibana required the RDF triples to be converted into a newline delimited json file. This required the triple ttl format to be converted into first JSON-LD format and then compacted JSON-LD, which was then converted to a newline delimited JSON file.

Lessons Learnt

Every challenge we faced was integral to the success of the project and we learnt a lot of lessons while working on the project. Not only did we learn building a knowledge graph from real-world data on an underdeveloped topic, but also, we learnt how to navigate working on various tools and software systems to showcase our results.

Firstly, while scraping the data, we learnt that when pages are not dynamic and if loading the entire page is not necessary, we could just use Scrapy to reduce the processing time and increase the speed. Secondly, in the homeworks, we designed RDF triples just for one node entity and did not connect them with any relationships. In this project, we were able to enhance our knowledge about node entities, their relationships, and their attributes to correctly identify classes and properties. This further helped us understand different semantic types and differentiate between building relationships and attributes. Moreover, we also followed Wikidata ontology for list generation and used predefined schema classes to design our ontology efficiently. Thirdly, we learnt working on different data types such as date-time, floats, etc and Neo4J settings while querying for results from them. We learnt the impact of multival setting in Neo4J settings and how it could impact the datatype of our attributes and hinder processing various queries on objects having multiple values for the same class and properties. Finally, we learnt to conduct data preprocessing using NLP libraries and devise our own blocking technique.

Lastly, we understood the importance of building a knowledge graph and having consolidated information that could be accessed and queried on one platform instead of searching for it on various sources. This project gave us the opportunity to apply all the skills and topics we studied in class and practiced in homeworks to a topic of our preference. The link to our demo is https://youtu.be/wU1QVkobC38.