Summary of the Project “Semantic, Context Analysis and Visualization of McGill theses data using word2vec”

Data Sources:

The theses data was present across multiple xml and html files, with the metadata present in nearly 49,000 files and the actual text of the theses present in 39,000 files.

Parsing the Data Sources:

The XML files contained multiple embedded CDATA xml files within each XML file, one of which contained all the metadata for the theses such as author, title, year of publication, etc. The metadata was parsed by Element Tree and Beautiful Soup using Python.

The HTML files were parsed using Beautiful Soup. There was no pattern in the HTML files in terms of content so the extra information like Table of Contents, etc. could not be removed, but the text was converted to lowercase and cleaned using mostly regular expressions.

A separate HTML XML pid linkage csv file was used to link the HTML files with their corresponding content metadata in XML. The linkage file, in turn helped to find out which of the multiple CDATA file needs to be parsed.

As the data was being parsed and cleaned, all the reproduced datasets were being stored in an AWS Postgres database.

Word2Vec Modelling:

The cleaned and parsed HTML data once available with its corresponding metadata, was used to create Word2Vec models which will be used to extract similar words to a word entered by the user.

Initially, the data was aggregated by year and department of the theses, however the resulting corpora didn’t have a rich vocabulary and hence could not be used further. The data was finally split by year, with just one split on year 2000, generating 2 corpuses, one containing all the theses before and including 2000, and another containing the later years.

Please note that the corpora can be split as you wish based on the kind of analysis you would do. For a semantic analysis using word2vec, doing a decade split would be recommended.

Once aggregated accordingly, the word2vec models can be generated using the Gensim library in Python.

Express JS:

To create the front end of the application, Express, a popular JS framework was used to create and configure the server side. A front end in HTML and CSS was created and linked to the Express server.

The modus operandi of the project is:

- An HTML CSS interface that asks the user to enter a word.
- The word is sent as a request to the Express Server.
- The Express Server sends the word to a python script linked in the code.
- The script generates and returns most similar words to the word entered for all the corresponding word2vec models.
- The server sends back the resulting dataset using csv (or JSON).
- The resulting csv is fetched and visualized using D3.

Visualization using D3 and D3 Force Layout:

Once the Express server sends back the response, it is served and visualized to the user using D3. For visualization, D3 Force Layout was used since it represents data elements (nodes) by linking them and creating a physical world simulation of how object interact in the real world.

The data elements in the project were represented using a bubble chart, but due to the scalable and adaptable nature of D3, any specific kind of visualization can be produced.