COSC2667 - Data Science Postgraduate Project

# Knowledge Graph Creation for the Legal Domain

Final Report

Dylan Pleiter (s3252987)
Siddarth Kumar (s3816208)
Yogesh Haresh Bojja (s3789918)
Mohamed Thaha Jiffry Ahamed Hibishy (s3851674)
Harvinder Singh Sethi (s3821903)

22 October 2021

## Contents

## 1 Introduction

Whether a high profile case being heard in the High Court or smaller matters involving disputes between two individuals, the time taken between applications and final resolution of a case can take many months, or even years, and many different hearings. Each hearing in itself contains many different participants including the applicants and defendants, their counsels, the judge, and sometimes even other parties who have no direct involvement in the case (interveners for example). With many of these participants also being involved in many other cases at the same time, it is easy to see how a complex web starts to develop where two participants can be directly involved in one case, but also indirectly through other cases at the same time. Likewise, sometimes precedents are important in a case as a judge may sometimes use the outcome of a previous, similar case to inform them how to adjudicate the case being heard at the time, adding further relationships between cases and entities. Keeping track of these relationships is important in law as, for example, if one lawyer has contact with one party then they can no longer represent an opposing party in the same case. With the sheer amount of data produced from all these hearings and the somewhat recent explosion in data analytics capabilities, there exists opportunities for the development of tools to assist in mapping relationships in the legal system.

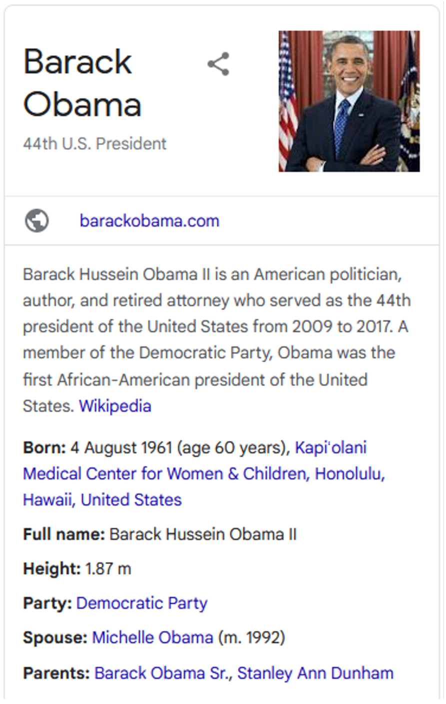
The client for this project is Loom Analytics - a data analytics firm based in Toronto, Canada, that mainly focusses on the legal domain. Currently, their main analytical product is Structura - a no-code data management platform, where non-technical users can build their own business workflow solutions without any IT support, using a drag-and-drop interface. Users can design their own data entry screens, bring in their data from spreadsheets and text documents of any type, connect to external data storage systems, and then search through all that data as well as predict future business events on it. They can then monitor their data on an ongoing basis and be warned of critical events before they occur, allowing them to take preventive action as needed.

The aim of this project was to create an application similar to Structura, but using a knowledge graph data structure, that allows non-technical end users to extract relevant information from a collection of court documents (such as participants or other documents cited), and then visualise the dataset in a graph format. The application should allow users to interact with the data such as adding or removing information, as well performing analysis using a set of pre-defined operations. The agreed deliverable was a fully functioning web application, as opposed to any primary analysis, written in a framework of the project team's choosing. The final deliverable to Loom was a Git repository uploaded to Loom's BitBucket account, and some additional documentation to be delivered after project completion.

## 2 Background

Already used across a range of domains, knowledge graphs are often utilised as the final step of many Natural Language Processing (NLP) tasks to store and visualise relationships between real-world entities. Knowledge Graphs are particularly useful for working with highly connected data, such as visualising social media networks, which can often be difficult to efficiently manage using traditional relational databases, especially when graph-like traversal of a network is desired. Knowledge Graphs have been applied in many large systems including Google search [1] (see Figure 1) and WolframAlpha question-answering service.

Graph databases are a useful tool in storing the information within a knowledge graph, as they



Barack Obama

44th U.S. President

barackobama.com

Barack Hussein Obama II is an American politician, author, and retired attorney who served as the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, Obama was the first African-American president of the United States. Wikipedia

**Born:** 4 August 1961 (age 60 years), Kapi'olani Medical Center for Women & Children, Honolulu, Hawaii, United States

**Full name:** Barack Hussein Obama II

**Height:** 1.87 m

**Party:** Democratic Party

**Spouse:** Michelle Obama (m. 1992)

**Parents:** Barack Obama Sr., Stanley Ann Dunham

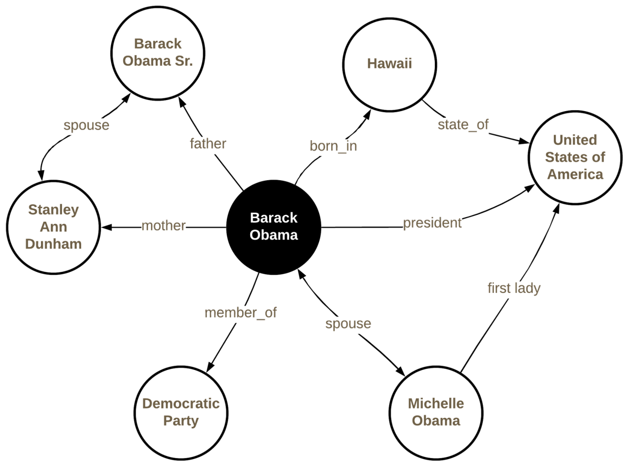


Figure 1: Example of Google Search summary and possible Knowledge Graph representation

allow the researcher to focus on the overall design of a knowledge graph system instead of lower level details. Several graph databases currently exist on the market, such as Neo4j and ArangoDB, and can often be used for similar purposes as more traditional SQL or NoSQL databases. However, the unique data structure and query languages of graph databases allow for efficient traversal of the data, meaning more thorough analysis is possible ‘out-of-the-box’. As of writing the most popular graph database currently in use is Neo4j [2].

Building a knowledge graph is often a complex process that requires extensive planning and collaboration with subject matter experts (SMEs) and researchers specialising in a range of areas. A typical development process may include the following tasks [3]:

1. Constructing a taxonomy in order to identify all different entity types within the given domain. Taxonomies usually take a hierarchical structure, mapping specific entity types to higher level concepts, such as a Golden Retriever belonging to the Canine family.

2. Identifying all the possible ways these entities can be related to each other in order to build a comprehensive ontology. This design is important to get right, as the ontology directly impacts most downstream tasks.

3. Named Entity Recognition that extracts all proper nouns within a given text, and then performs entity resolution for when a given real-world entity can be represented textually in multiple different ways. Initially the entities will not have a type, so often a model needs to be built to classify them according to the classes in the taxonomy.

4. Identify relationships between pairs of entities. In NLP, this requires extracting the verbs within sentences, and often requires further processing of the sentence or body of text as

a whole in order to understand exact meanings. These relationships will need to classified according to the entity types and the ontology mentioned in step 2.

5. With the process of creating a knowledge graph complete, an efficient querying system usually needs to be built in order to handle tasks, such as semantic search and prioritising certain entity or relation types (particularly if the knowledge graph is to be utilised as a search engine).

While these tasks serve as a general guide to building a domain-specific knowledge graph, not all tasks are relevant to this report - mainly due to the scope of the project.

## 3 Methodology

### 3.1 Data

In Ontario, as with much of the developed world, some time after a hearing a PDF document is released detailing the participants, facts, reasoning, and outcomes as interpreted by the judge. These documents serve as an official record of the hearing, and are often used in future cases for appeals or to establish precedence. As these PDFs are usually released to the public, Loom has collected a large sample and allowed users to search through them using Structura - their online analytics platform. For the purposes of this project Loom has given access to the project team of their collection of PDFs stored on AWS S3, which in turn has been used to inform the development of the web application.

The structure of these documents are usually similar across different jurisdictions, and are structured in such a way as to allow easy referencing of participants and individual paragraphs. As informed by Loom, within each PDF supplied as training data there are three distinct 'zones' which contain the following information:

- **Zone 1** contains the data about the circumstances of the hearing, including the participants, location, date, filing number etc.
- **Zone 2** contains the facts and outcomes of the case as interpreted by the judge. This section is usually split into paragraphs, each of which are numbered which allows for easy referencing.
- **Zone 3** is reserved for the judges signature confirming the contents of the document.

Explained further in the next section, the main focus of this project is on Zone 1 for court participants, and Zone 2 for establishing facts and outcomes as well as detecting references to other documents. Additionally, this project only deals with documents from the Superior Court of Justice in Ontario, Canada, as this is the focus of Loom's operations. Using only documents from one court allowed the project team to develop rule-based algorithms to extract relevant information, as different jurisdictions usually have unique formats for Zone 1, often complicating the automated extraction of participants.

### 3.2 Natural Language Processing

In any knowledge graph construction, a majority of time and effort needs to be spent on information extraction (IE), and this usually requires a major focus on natural language processing (NLP). This work stream was able to begin relatively early in the project, as there are several tasks within

NLP that are consistent across all knowledge graph construction systems. Other tasks needed to wait until sufficient planning had gone into the design of the application.

Named Entity Recognition (NER), as the name suggests, is a process of mining through a text corpus and identifying and extracting real-world entities that will form the nodes of the knowledge graph. Examples of named entities could be instances of people or companies. This process is not as simple as tagging all proper nouns within a text, as names (especially for organisations) can consist of many other types of words. For example consider the name 'Bank of Melbourne' - any simple part-of-speech tagger will only tag Melbourne as a proper noun and treat 'bank' as a noun and 'of' as a preposition. Obviously in any proper NLP pipeline the entity 'Bank of Melbourne' should be a very different entity to the city of Melbourne. Thankfully, as this kind of task is relatively common in NLP, there exists several online toolkits that can assist with NER.

Relationship extraction can only begin once NER is functional, and also relies on the ontology having been completed. This is because different entity classes can only be related to each other in a specific way. Early in this stage of the NLP a system had been set up that extracted the raw relationships between Zone 2 entities, but upon discussions with Loom the decision was made to forgo this part the IE process and instead focus on Zone 1. From zone 1 we extracted citation, court file no, date, and province using rule-based algorithm. Court participants - applicant, plaintiff, judge, counsels involved, place of hearing - were recognized and extracted using the Spacy model. Spacy model is trained on these case laws and is able to extract court participants efficiently.

The final step of the IE process undertaken was to extract references to other hearings and cases within Zone 2 of the documents. As mentioned in the introduction of this report, often judges will use the findings of a previous case to inform their judgements in the current case. Consider the following example [4]...

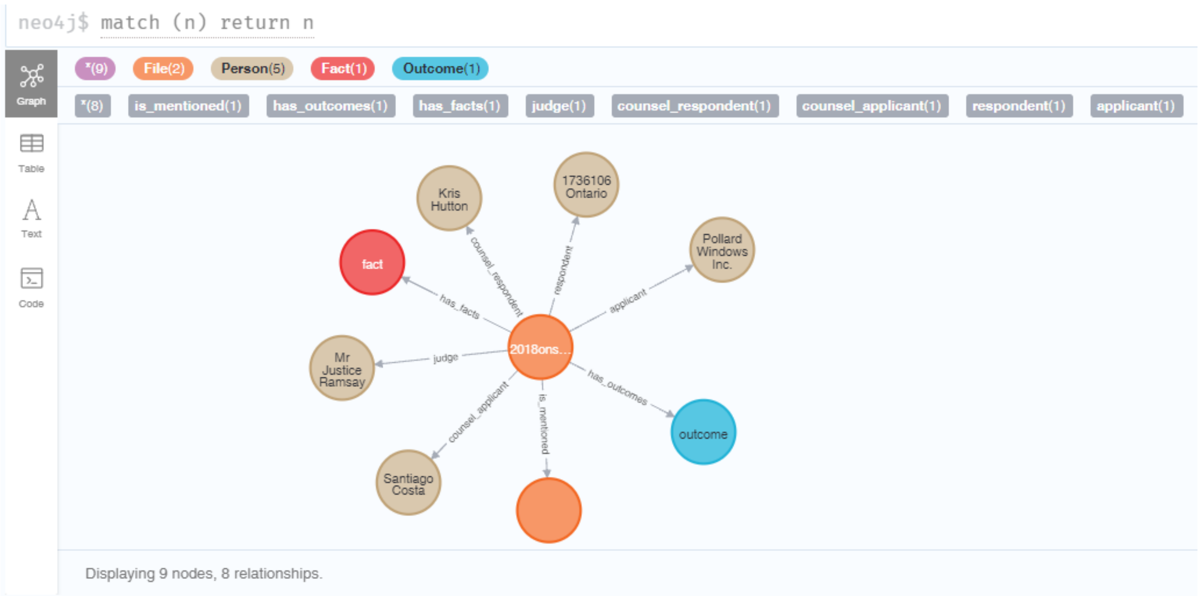
> "Much of the history of the proceedings is set out in my endorsement of March 21, 2018: *2018 ONSC 1924*. The plaintiff seeks full indemnity of $160,422.70, which does not include previous costs awarded or costs of proceedings in five interlocutory appeals to the Divisional Court and a motion to stay the finding of contempt of March 21, 2018."

As can be seen, the judge has cited a previous hearing (2018 ONSC 1924) of this same case in their reasoning. This citation number follows a consistent pattern across all hearings in Canada, with each court having a different combination of letters identifying it. With this is mind, some regular expressions were able to be written that could be applied to each paragraph within Zone 2 of a document and extract all citation numbers. The handling of this information is explained later in the Database subsection.

## 3.3 Database

There are several factors that could go into the selection of a database to support this application, especially considering the sheer number of choices and the different capabilities, but after extensive research the project team decided to proceed with Neo4j due primarily due to being the most popular graph database currently available. The thinking behind choosing the most popular graph database is that there was likely to be a greater amount documentation and learning resources available compared to smaller providers - an important point considering prior to this project none of the project team had any direct experience with graph databases. Further confirming the decision to continue with Neo4j is its generous free tier on their fully managed cloud service - Neo4j Aura, meaning several databases were able to be set up in order to build the application while not incurring a cost for Loom.

Having selected the database, the next step was to design how the database would operate. The very first step required when processing a document was to create a File node in the database on which information is stored such as the date, hearing type and citation number. As other entities are detected as a result of the NLP, an Entity node is created and connected to its corresponding File node using the relevant relationship name (i.e. 'applicant' for each applicant detected). Additional File nodes would need to be created if a case is directly referenced in Zone 2 and it hasn't already been processed itself, with the associated information mentioned earlier remaining empty until (or if) it is finally processed. Figure 2 shows how the database may look after one document has been processed. As further documents are processed, sub-graphs will begin to be connected as entities are detected in multiple documents.



```
neo4j$ match (n) return n
```

```
Displaying 9 nodes, 8 relationships.
```

Figure 2: Example database after a single document has been processed

Neo4j's inbuilt query language is known as Cypher. With many similar operations to the more traditional SQL, such as transactions and the ability to add and query data, Cypher has been optimised for graph databases. Example queries may be to return all nodes of a specific type or to change properties on a relationship. Additionally Cypher provides an additional library called Awesome Procedures On Cypher (APOC) [5], that provides functions to assist with simple queries as well as performing complicated procedures. This package has been used extensively throughout the application.

In the application, the database needs to interact with both the NLP and the user interface. As some of the operations between these two streams of the application are similar, such as adding nodes and relationships, several queries were able to be reused throughout the application. In addition to graph manipulation queries, analytical queries, such as determining whether two nodes were linked through any series of relationships or what the shortest path is between two nodes, were integrated within the application. The application of these queries are discussed further in the Results section.

## 3.4 BERT Integration

Up until this stage the pipeline has only extracted the entities and relationships contained within Zone 1 and 2 of documents, but there is much more information contained within these documents (especially Zone 2) that may be beneficial to incorporate into the knowledge graph. While an application based on the current NLP process would have been sufficient to meet Loom's requirements, an attempt was made to add a module that incorporated the summary of facts and outcomes within Zone 2. To implement this functionality, contact was made with one of Loom's other RMIT project teams to obtain their completed BERT model, which could then be integrated into this application. To understand how this model was trained, please read the report of the BERT NLP 1 project team.

The BERT model received works by passing the unfiltered extracted text from all zones of uploaded PDF, which then returns all the sentences contained within classified into one of several categories. For the purposes of this application, only those sentences classified as either 'Fact' or 'Outcome' were kept for further use in the application. With many sentences that may be classified into each of the two categories, the decision was made to only store the five most likely facts and the two most likely outcomes. While likelihood of a given sentence belonging to a particular category is clearly not a substitute for determining relevance in summarising a case, the model was not designed for this project's specific use case, and as such should be viewed as a proof-of-concept. How this functionality could be improved on is further discussed in the Results section of this report.

Once the most likely facts and outcomes had been identified,we are running a reverse search on the formatted version of extracted text to extract the complete clause to which a classified sentence belongs to. The clauses matched with this reverse search is then stored in the 'fact' and 'outcome' node of the graph respectively and connected to the containing document. The decision was made to store this information on their own nodes instead of properties on the File node as it allows for future development where two files could share similar facts or outcomes, which could then be used as a way to connect the two files. As explained in the Results section of this report, these nodes are not displayed on the knowledge graph in the UI, but rather as additional information to the side of the graph - serving as complementary information giving the user more context on the selected file.

## 3.5 Application Framework

As the main deliverable of this project was an application, an important decision early on in the project was to settle on a framework to use. As Python offers the best support for NLP, a Flask or Django framework was favoured. The project team, in consultation with Loom, eventually settled on Flask due to it's documentation and relative ease of setting up.

The structure of the application source code relied heavily on modules, where a particular set of tasks could be grouped together allowing for easier understanding of code and to promote code reuse wherever possible. Each module is further split into files to further separate different tasks, such as providing API endpoint functionality compared to handling the actual processing.

A summary of each module in the application:

**Main**
A simple module containing the endpoints for the non-graph related operations of the application. The main purpose of this module is to render the upload page.

**Database**
This module contains all the raw Cypher queries and the code to run them. There are several methods available that allow any other part of the application to interact with the database.

**Graph**
The primary function of the Graph module is to serve as an endpoint accessible to the user interface via an API. This module is almost completely dependent on the Database module in order to operate, with only decoding and validating requests and sending responses happening here.

**Model**
This module handles the integration of the BERT model from the BERT NLP 1 project team. Using the same output file generated from the PDF module, this module handles all the functionality as described above in BERT Integration.

**PDF**
The largest module in the application, here is where initial uploaded PDF file is handled by coordinating with Adobe's PDFservices endpoint and processing the response. The scraping of the returned JSON files results in the relevant database operations being called in order to create the initial knowledge graph.

## 3.6 User Interface

Development of the user interface (UI) commenced relatively late in the project, due to the reliance on implementing the logic sitting behind the knowledge graph, as well as having enough of the server developed to be able to connect to the database in the browser. Once these milestones had been achieved, and a suitable design had been agreed upon in consultation with Loom, development could begin in earnest.

One of the most important decisions to be made for the UI was how to render the knowledge graph within a browser while still maintaining maximum control over display and responding to user click events. For this reason and due to the abundance of documentation and examples online the project team opted to proceed with Vis.js [6]. Vis.js is a library built on D3.js that is optimised for both speeds of rendering and level of user interactivity. While also providing support for other types of visualisations, Vis.js is mainly used for complex graph visualisations, including hierarchies, networks, or timelines.

For styling the UI the project team extensively used Bootstrap [7] - a MIT licenced, free and open source CSS framework focussed on responsive web development, as well as custom CSS written by the project team. User events were handled primarily with jQuery [8] - a javascript plugin that greatly simplifies the process of making requests to external servers and modifying elements on a web page. The decision to proceed with these options were made purely based on members of the team's previous experience with these libraries.

The first task was to design the PDF upload page where users could upload multiple PDF files from their PC to be processed by the server. Once a response has been received by the browser, the user is then redirected to the main visualisation page.

The UI was designed as an interactive dashboard, with functionality to allow non-technical users to interact with the data via a collapsible sidebar. Each action on the sidebar was implemented and refined alongside writing of the Cypher queries on the server. These actions, as explained further in the Results section, includes adding or changing nodes and relationships, filtering the graph, and performing analysis.

# 4 Results

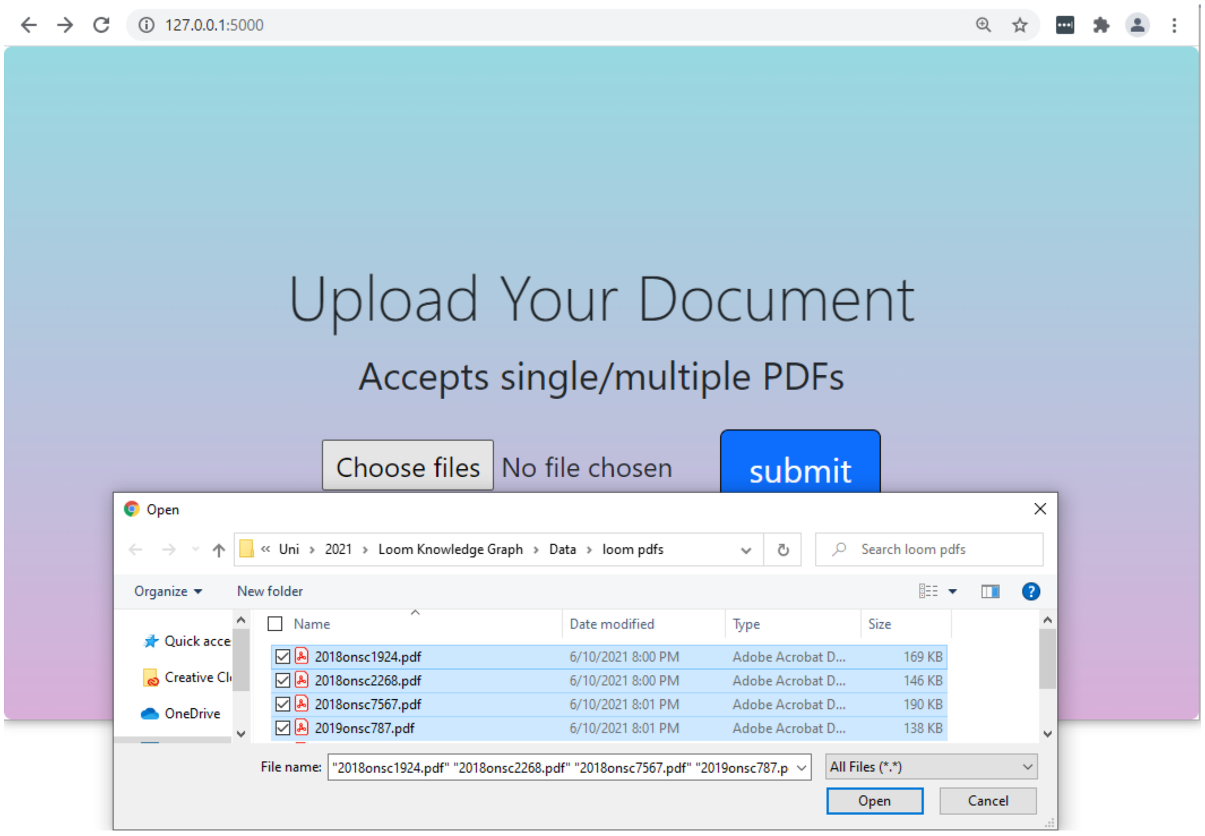
## 4.1 Application

The product delivered to Loom was a web application consisting of the parts outlined in the previous section. The two web pages that are shown to the user consist of an index page that handles document uploading, and a UI that displays the data and allows the user to interact with the graph output.

The first thing a user will do when interacting with the application is to upload PDFs to be processed via the methods described earlier (see Figure 3). Complex processing is performed by the server, and once inserting the relevant data into the database has been completed, the user is redirected to the UI that presents the raw knowledge graph to the user (Figure 4). The user should instantly be able to see the connectivity from their input documents and begin exploring the different entities and relationships.

On the sidebar on the right hand side of the visualisation the user can select from a set of predefined actions to interact with the graph. These actions are:

- **Create Node** - Add a node to graph with a user-defined label
- **Rename Node** - Rename an existing node
- **Set Node Property** - Set a property on an existing node with a user defined key-value pair
- **Set Relationship Property** - Set a property on an existing relationship with a user defined key-value pair
- **Remove Node** - Removes a node from the graph
- **Merge Nodes** - Merge two nodes (and their relationships) that the user deems are equivalent. This is useful as the document processing may not pick up that two very similar names refer to the same entity (possibly due to differences in wording or the addition or removal of characters)
- **Create Relationship** - Creates a relationship between nodes. For example the user may determine that two participants in a case are spouses, and as this will not be picked up in document processing they can manually add this information.
- **Remove Relationship** - Removes an existing relationship



| Name | Date modified | Type | Size |
|---|---|---|---|
| 2018onsc1924.pdf | 6/10/2021 8:00 PM | Adobe Acrobat D... | 169 KB |
| 2018onsc2268.pdf | 6/10/2021 8:00 PM | Adobe Acrobat D... | 146 KB |
| 2018onsc7567.pdf | 6/10/2021 8:01 PM | Adobe Acrobat D... | 190 KB |
| 2019onsc787.pdf | 6/10/2021 8:01 PM | Adobe Acrobat D... | 138 KB |

File name: "2018onsc1924.pdf" "2018onsc2268.pdf" "2018onsc7567.pdf" "2019onsc787.p

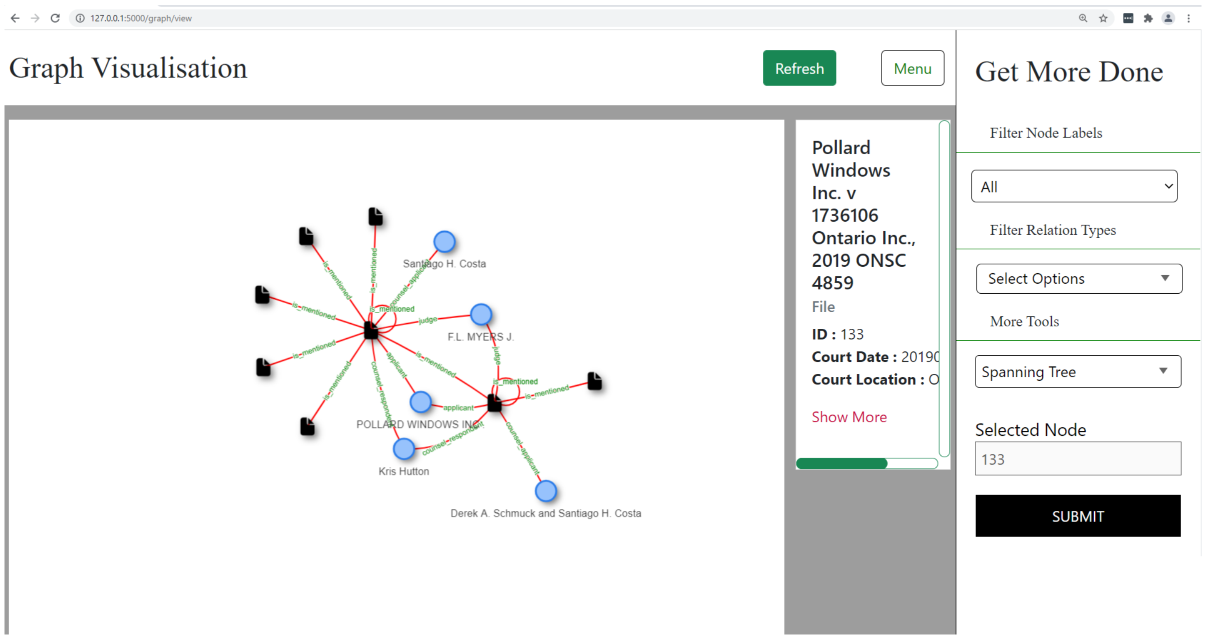
All Files (*.*)

Figure 3: Upload PDF page

- **Get File Details** - Retrieves the summary of facts and outcomes from the database and displays this in the information panel.
- **Spanning Tree** - Given an input node, this tool allows the user to filter the graph to display only other nodes and relationships that are reachable. This is a useful tool for when there may be multiple sub-graphs displayed in the visualisation and they would like to explore one of them in more detail.
- **Shortest Path** - Highlights the shortest path between two nodes. This function is useful if the user wishes to know how two people may be related, if they are at all.

These implemented actions should be seen as non-exhaustive, with many more available to be introduced without too much difficulty (explored in more detail later).

Allowing the user to have a generous amount of control over the structure of the graph is motivated by the fact that sometimes the scraping may be imperfect, such as small differences in names yielding two separate nodes when they should be one. Additionally, the user could be in possession of extra data that cannot be ascertained from the PDF alone that they may wish to have represented. The analytical tools allows the user to explore different regions of the graph instead of everything at once, so they can really explore what they may find interesting.



Pollard Windows Inc. v 1736106 Ontario Inc., 2019 ONSC 4859
File
**ID :** 133
**Court Date :** 2019C
**Court Location :** O

Filter Node Labels
Filter Relation Types
More Tools
Selected Node 133

Figure 4: User Interface

## 4.2 Further Improvements

While the delivered application serves well as a proof of concept, there still requires some work to get it production-ready, a lot of which falls outside the scope of this project (such as authentication and moving to a more scalable web framework such as Django). In terms of the core functionality of the application the project team have some suggestions as to how functionality could be improved in order to maximise the benefit to the end user.

### PDF Scraping

The dataset that was used to inform the creation of the application consisted of PDF documents that followed one of a few formats based on the Ontario court system, for which the scraping algorithm works reasonably well. If this application was to be rolled out to other jurisdictions, or even countries, it is likely the format of these documents would change. These different formats, especially for Zone 1, would mean that the implemented data scraping algorithm may struggle to identify the nature of the participants (i.e. determining if a name refers to the judge or the plaintiff). If this process could be improved then it would mean that the end-user would be able to spend less time cleaning up the data output and start performing analysis sooner.

### Utilisation of Other Data Sources

This application deals with only the contents contained within the uploaded PDFs, meaning that some information relevant to the case may be missed. For example, most documents don't list which firm the lawyers involved in a case are representing, which may be useful information to know. If there existed a database with such information it may be beneficial to link it to this application to either supply supplementary information, or even to add its own information into the knowledge graph.

### BERT Model Integration

The BERT model that was integrated into this application wasn't designed specifically for this application, and will likely have other uses within Loom's setup. If a similar model could be created to further refine the data within Zone 2 of the PDFs, the data scraping could begin to extract more relevant information and insert it into the knowledge graph. For example, currently the BERT model only classifies sentences as facts or outcomes of a case - of which only a selection of the most relevant are utilised, but it should be possible to build a model that starts to interpret the meaning behind these sentences such as *who* won the case, what evidence was submitted, or how the participants relate to each other beyond just being involved in the same case.

## 5 Conclusion

This project was completed in three parallel, yet dependent, streams, each of which highlighted a different aspect of data science as a skill set. The first stream - Natural Language Processing, required us to be able to extract meaningful information from a document, while understanding the specific domain, being the court system in the case of this project. The second stream, knowledge graph representation and database management, required us to think about how to store information and query it efficiently. The last stream, being the UI development, required us to think about how to best display vast amounts of data to an end-user in order for them to be able to extract insights and perform custom analysis. Additionally, as the end product was a fully functioning application that combines all three streams, we were able to understand how these different aspects, each of which is a separate skill in data science, could be brought together to form a pipeline that performs much more comprehensive task than they could ever hope to separately.

As mentioned previously in this report, while there is undoubtedly further work required to get the application into a production-ready state, we believe that the application as-is serves as a promising proof of concept for a product that could complement their existing Structura product. If implemented in the future, users will be able to see what was previously standalone PDF documents in a much more interesting light that truly reflects the real-world relationships that make up much of the legal system, and indeed society as a whole.

## 6 Appendix 1: Roles and Responsibilities

Due to the size of the team, the communication methods and separation of tasks was vital in order for the project to be completed as smoothly as possible and in a timely manner. Early in the project a Microsoft Teams channel was set up to allow easier sharing of documents, as well as a WhatsApp messaging group to facilitate more casual communication. In order to allow for the easiest method of collaborating we created a GitHub organisation where we could create as many repositories as we needed and have them accessible to everyone with no further setting up.

For the lifespan of the project the team maintained roughly three days of contact per week, including a Tuesday morning meeting with Loom to discuss design issues and update progress; a one hour discussion on Friday afternoons with our mentor, Estrid He; and a less formal catch up usually on Sunday or Monday evening to discuss smaller issues and to plan the week ahead. Additionally, smaller meetings to achieve an particular work outcome were held as required, usually on a ad-hoc basis. This arrangement seemed to work well for the team with most team members attending each meeting, although with the number of contact hours it was never an issue when a team member couldn't make a specific meeting.

As mentioned in this report and the interim report submitted earlier in the semester, the work was divided into 3 main work streams, of which each of the five team members were to focus on. Specifically, the roles and responsibilities that each of the team members focussed on were...

**Dylan Pleiter** took on a project management role with a focus on planning the work and meeting deadlines. He was also primarily responsible for setting up the web framework (Flask), designing the database 'schema', writing the Cypher queries, and report writing.

**Siddarth Kumar** also took on a project management role focussing on coordinating with each of the work streams to ensure that each person's relevant work fit together into the whole. In addition, he assisted with the NLP, UI, and was responsible for integrating the BERT model from the other RMIT project team.

**Yogesh Haresh Bojja** was primarily responsible for the NLP. As mentioned in the report, this involved implementing the data scraping and entity recognition.

**Mohamed Thaha Jiffry Ahamed Hibishy** was responsible for implementing the functionality in the UI, and developing the knowledge graph visualisation.

**Harvinder Singh Sethi** was closely involved with Thaha for the graph visualisation and the javascript functionality. He was also responsible for the look and the feel of the upload page and the graph dashboard.

# 7 Appendix 2: Individual Self-reflection

## 7.1 Dylan Pleiter

This project was quite interesting to be involved with, in that I was able to be directly involved with the design and implementation of the product. What I found challenging was trying to envision the finished product in the early stages of the project, as with creating a knowledge graphs there are no shortage of possibilities - they can range from simple web applications to computationally intensive pipelines. Reading some academic papers and web articles got me excited for the potential of this project - until it came time to actually implement the features. An example of this is when, early in the design phase, I created a fairly complicated ontology of how different entity types could be related to each other (such as a person being employed by a company), but it soon became clear that actually implementing this requires a great deal of expertise and effort. However, I don't really see the process of creating this complex ontology as wasted time as it helped me gain perspective on how knowledge graphs are typically built.

In terms of planning, setting timelines required a lot of flexibility as what I thought would be minor tasks actually were a lot more complex that initially considered. In discussions with our mentor it soon became clear that usually academic or industrial-level knowledge graph construction processes requires one researcher focussing small aspects (in the grand scheme of the project) for extended periods of time, such as entity resolution or relationship detection - time that we couldn't really afford. Luckily, in consultation with Loom, their vision wasn't quite so large, meaning that we could just detect the participants and citations in a document and connect entities that way.

Probably the most challenging aspect of the entire project were the relatively few contact hours with the client. The time difference between Toronto and Melbourne was 10 hours, moving to 8 hours when daylight savings changed for both cities, meaning our weekly calls were the only time we could discuss progress and clarify issues. We did try to utilise emails and WebEx messaging when we had specific queries, but again the time difference impacted our ability to have any sustained dialogue. In the end we developed a routine where we would do extensive planning and work during our workday, and then discuss many things at once, whether by message or call.

In terms of the project management, I found that spending some time before undertaking a piece of work actually planning the process saved a lot of time and stress. This is true for individual pieces of work and the entire project as a whole, where setting early goals and timelines made reaching milestones much easier. It also meant that we did not need to panic when we still hadn't properly started our application until about the mid-semester break, as we had already set aside time for everyone to do their own personal research in the form of reading and experimentation on the topic of knowledge graphs. This time was also important to discuss with Loom what the overall design of the project would be, as compared to other projects, this one had a very loose set of goals due to the nature of the topic and the deliverable. While there were plenty of times we had to readjust goals or timelines, I think we all found having early discussions about what we were aiming for and when it would be completed to be beneficial.

## 7.2 Siddarth Kumar

The requirements for building an end-to-end knowledge graph-based analytical solution were quite challenging from the get-go, which made this project exceptionally interesting. Working on this project helped me learn the concepts of knowledge graphs and ontology and made me realize how machine learning integrated with knowledge graphs can help develop powerful analytical tools. While I did get to implement these concepts in an industry project, I still have a long way to go to develop a comprehensive domain-specific KG based machine learning solution.

'Knowledge-Graph' being a new concept, the initial couple of weeks were spent on gaining subject matter; we did quite a bit of reading on existing knowledge graph-based solutions. We divided the research, design, and implementation, based on our experience and expertise, thus creating 3 different workstreams. As part of NLP and App design workstream, one of my initial responsibilities was research into various NLP models - to decide upon the best fit for our use case of Named Entity Recognition. After the research phase, the next task I tackled was the infrastructure setup and the application design. In application development phase, I got an opportunity to work with various frameworks and technologies, ranging from 'Adobe PDF scrapping' (for text extraction), 'Neo4j' (Graph database) and Cypher (graph query language), to 'Flask'. Based on my expertise as a software engineer, I suggested a 3-tier application architecture, allowing all the three workstreams to work together on the application, independent of each other.

One of the drawbacks was the underestimation of design implementation. The initial product design we proposed, aiming to be market-ready, had to be scaled back when we started implementing the design. We quickly realized that it wouldn't be possible in a relatively short period of 12 weeks. With keeping fundamental requirements from Loom Analytics, we scaled back the product design for a Minimal Viable Product. The key factor for our group functioning so well was the extensive planning and forethought we put into the work. Although we commenced building the deliverable during the mid-semester break, early planning and time invested in the research saved us significant amount of time during the implementation.

Speaking of team members, I was blessed to have an amazing team. Dylan was always on top of the project management, from coordinating with other teams to setting up a timeline. Yogesh was phenomenal with development of a quick proof-of-concept and the Named Entity Recognition model. Thaha and Harvinder were amazing in setting up a user-friendly UI and thus bringing together the complete solution.

This project has been an incredible learning experience for me. I got the opportunity to learn how to build an industry-ready solution by working with a team, while directly coordinating with the clients. This project has far-reaching learnings for me that will boost me in my career prospects.

## 7.3 Yogesh Haresh Bojja

I had an exciting experience studying data science at RMIT and was very keen on applying my knowledge gained during the course at industry level. Industry work at Loom Analytics was just the right opportunity I was looking for. Though working on NLP techniques and building models made me excited, 'Knowledge graph' was something I had lately heard of and was looking forward to try hands on.

Initially I felt a bit worried because of the reason that our project had the inputs from other 2 teams, and we could not get those inputs as they were working parallelly along with us. All credit goes to our team leader i.e., Dylan who managed to streamline the workflow so well. It was educating to read papers and information about new technologies to me like Neo4j, Cypher, and Spacy. At the start I was able to get proof-of-concept up and running which, I feel kind of gave an idea about our project to all the teammates and check whether our understanding was on same page with that of Loom Analytics. I was assigned to work on Natural language processing stuff. After research I finalized to work with Spacy for Named entity recognition. I had worked with the NLP models before, but these models were pre-trained and existing on the internet. I always wanted to build the model which is domain-specific and get to know what steps and challenges are involved with it. This project surely made me do what I wished for. I faced many challenges during the model creation step, and I liked resolving those, this procedure enabled me to learn various terminologies out of the box. During the project implementation phase, I faced couple of problems regarding the Canadian law terms and I would credit Siddharth for arranging calls personally with Loom, understanding those terms and getting back to me. I feel accomplished learning about different phases of NLP model building and being successful in completing those. Once the backend stuff was on the verge of getting ready I was worried about the front part as we had minimum time to complete it. Front end part was created beautifully within short span by Thaha and Harvinder which is worth appreciating.

I feel we as a team pulled off a major project pretty well in such a short span as the type of deliverable expected by Loom was not in the scope of being completed within 4 to 5 months. Overall, it was an excellent experience working with an amazing team and under the guidance of a best mentor.

## 7.4 Mohamed Thaha Jiffry Ahamed Hibishy

Reflecting upon past experience is an important capability for all professionals in order to develop better communications skills, conflict resolution and enhance future performance. Looking back at the project initiation, knowledge were new for every member in the team. None of had any technical knowledge on them. Apart from that, the legal domain was a big challenge for us too. The very few weeks were spent doing research on Knowledge graphs and the Legal Domain. Every member of the team was trying the best to figure out where should we start. We tried playing around developing several applications to gain some experience. Mainly the team was developed into front end including graph visualization and the backend to work with the NLP and Extraction. I was given the responsibility of visualization. Getting deeper into the visualization, as per our research we agreed upon using Neovis as the language and technology to be used for the visualization. Mainly because we found it on Neo4jd official channels and claimed to be directly integrated with the Neo4j database.

While the backend team worked on developing a flask application on the extraction, the UI team worked on developing a web page that can visualize the graphs. We were successfully able to visualize the graph in a webpage. It was quite easy. It was a big achievement for the whole team as we figured that the pieces are starting to fix. But the easiness came with a price. It gave us less control over the visualization. It was a huge challenge and conflict on the graph manipulation. Even though Neovis was able to directly pull the graphs and visualize, it gave us less control over the visualization. We did not have control over how the graphs are visualized, filter nodes or relationships. Upon further research, the team suggested using Vis Js for visualization. The visualization was made from the scratch again. It was totally worth it. Vid Js had more control over the graph and was able to work with JavaScript functions. We integrated the webpage with the flask application that had been developed to upload and extract the entities of documents. The final visualization had more features including graph manipulation without cypher queries.

Overall I really enjoyed the project. Initially it was frustrating as we had no knowledge on anything and we had no idea on where to start. But with time the pieces starting to fit together and everything was making sense. With no previous experience on working on technical projects, i learnt a lot on JavaScript, Cypher, knowledge graphs, python and working with API's. The project was a huge achievement for me as it's the first industry related technical project I worked on. I really feel I have learned a lot. Every member of the team was highly engaged in the project. There were no conflicts within the team on anything from the beginning to the moment this reflection is written. Everyone worked on their responsibilities and assigned tasks thanks to our team leaders communication and leadership skills.

## 7.5 Harvinder Singh Sethi

Self-reflection in learning implies analysing the way an individual learns. It infers that without contemplating how we learn; we can never acquire the knowledge important to address helpless propensities and certify great ones. This intellectual course of self-reflection accordingly assists understudies with further developing learning results, however, cultivates self-directed learning, a repeating interaction that includes intending to wrap up a scholastic responsibility; utilizing methodologies to screen progress; assessing the result; and utilizing that information to direct future assignments. Coming to our project Knowledge Graph, firstly we have provided Client-Loom, with expertise and experience of each member of our team based on which I then nominated Dylan as our team leader. In the early stages of the project, we did not have much idea on what is required and to be produced, we then got enough insights in roughly first 3 weeks to produce a plan and design architecture for the project, we started with our initial submission for the project scope and deliverables which we all developed with the help of weekly team meetings and discussions we had.

In the second submission for video presentation of project goals and plans I was involved in creating slides and presenting/recording them for our submission. We, then divided the team into 3 workstreams NLP, UI and Visualisation, and the database management workstream. I was basically into UI and visualisation workstream along with Thaha. I was responsible for the complete UI/Frontend of our final Flask application, including some of the visualisation tools and JavaScripting features. According to our initial plan and meetings we used Neovis.js for the graph visualisation, in parallel I was designing custom and bootstrap based components for our app which are available on https://codepen.io/hector-harvinder . I then create the homepage UI for uploading multiple pdf files, also the Graph loading page with a search bar, submit and refresh buttons, loading the graph and adding UI to nodes, relationships etc. all this was being made responsive and scalable for future extensions. Due to limited functionalities and access to the canvas and JavaScript of the Neovis.js and according to clients' requirements it was getting tough to fulfill some of the important graph manipulation functions like create, delete, add node/relationships etc.

After further research and suggestions with team we moved to vis.js where we have good examples and resources available to refer to and it was easy to proceed. As we now have graph loaded with vis.js I must change the UI, and the team suggested to keep it dashboard styled, I used bootstrap 5 and jQuery to create UI and handling events, I also created a responsive sidebar having all the tools and graph manipulation functions Static UI implemented, which was then made dynamic by thaha. For the final submission me and Thaha together recorded/presented the complete running of the application and all the functionalities (as we both were the UI/Functional workstream) also added bits for the report For UI/Visualisation section. Apart from this, I also learned how the NLP was working and integrated in our app also the adobe pdf scrapper and extractor from Our other team members was insightful for me. Special thanks to Dylan for always being a good Team Lead. To conclude I would say I literally learned a lot not just technically but also teamwork and work synchronisation as every member of the team was complementing each other's work and equally motivated towards the project.

## References

[1] Amit Singhal. *Introducing the Knowledge Graph: things, not strings*. 2012. URL: `https://blog.google/products/search/introducing-knowledge-graph-things-not/`.

[2] URL: `https://db-engines.com/en/ranking` (visited on 15/10/2021).

[3] Mayank Kejriwal. *Domain-Specific Knowledge Graph Construction*. Springer, 2019.

[4] *Pollard Windows Inc. v. 1736106 Ontario Inc.* 2019.

[5] URL: `https://neo4j.com/labs/apoc/`.

[6] URL: `https://visjs.org/`.

[7] URL: `https://getbootstrap.com/`.

[8] URL: `https://jquery.com/`.

**RMIT University**
**School of**
**Computing Technology**
**COSC2667 Data Science Postgrad Project**

## Statement of Contributions

1. The members of the team must agree on the approx. percentage contributions and list the major contributions for each team member (e.g., "researched ML tools"; "pre-processed data"; "performed error analysis on results", etc.)
2. Contribution percentages will be used to adjust individual grades from overall team grade.
3. Each member of the team needs to sign this form. Contributions must add up to 100%.
4. A project will not be marked until we have the signed copy of this form.

Project Title: **Understanding and querying relationships in a dataset using Knowledge Graphs**

Team Name: N/A________________________Partner: Loom Analytics__________________

| Name | Student Number | Percentage | Signature | Date |
|---|---|---|---|---|
| Dylan Pleiter | s3252987 | 20% | [illegible] | 21/10/2021 |
| Major contributions: Project management, setting up web framework, Cypher queries, report writing | | | | |
| Siddarth Kumar | s3816208 | 20% | [illegible] | 21/10/2021 |
| Major contributions: Project management, BERT model integration | | | | |
| Yogesh Haresh Bojja | s3789918 | 20% | [illegible] | 21/10/2021 |
| Major contributions: Natural Language Processing, data scraping | | | | |
| Mohamed Thaha Jiffry Ahamed Hibishy | s3851674 | 20% | [illegible] | 21/10/2021 |
| Major contributions: UI functionality, graph visualisation | | | | |
| Harvinder Singh Sethi | s3821903 | 20% | harvinder | 21/10/2021 |
| Major contributions: UI design, graph visualisation | | | | |

- You should include names and contributions of team members not in this course (e.g., from Cybersecurity or Analytics, or from another university).
- The completed form should be included in the Final Report or email directly to Lawrence.
- If you have any questions or if there is any dispute amongst the team about contributions, please contact Lawrence at lawrence.cavedon@rmit.edu.au