

# **Irish judgement recommendation using a knowledge graph**

**Yu Xin, BAI**

## **A Dissertation**

Presented to the University of Dublin, Trinity College  
in partial fulfilment of the requirements for the degree of

**Master of Science in Computer Science (Intelligent Systems)**

Supervisor: Declan O'Sullivan

August 2022

# Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

---

Yu Xin

August 17, 2022

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

---

Yu Xin

August 17, 2022

# Irish judgement recommendation using a knowledge graph

Yu Xin, Master of Science in Computer Science  
University of Dublin, Trinity College, 2022

Supervisor: Declan O’Sullivan

Nowadays, one-third of the world’s population lives in Common Law systems mixed with Civil Law, including Ireland. Common Law is characterised by repeated references to decisional precedents that eventually produce universal, agreed-upon laws. This research aims to apply the knowledge graph to the Ireland legal field to optimise the current case search experience of public platforms. According to the neural citation, a system that recommends highly relevant precedents combines knowledge extraction, knowledge integration and a recommendation algorithm to realise the vision. The project uses the official Irish service website courts.ie as the data source, collects and processes semi-structured and unstructured data through crawlers and natural language techniques, and enriches the data content with third-party databases to achieve ontology-based design for knowledge graph construction. MetaGraph2Vec performs graph embedding of the knowledge graph for use in downstream recommendation tasks. The Latent Dirichlet Allocation (LDA) model performs the topic modelling task to claim the labels required for MetaGraph2Vec training. The system is evaluated to achieve good recommendation results, with minor ontological defects, 72% macro-f1 scores and 48.15% 48.15% system validity ( $\gg$  probability of random recommendation: 5.29%). The contribution of the work can be concluded as follows: first, a small Irish Court Case Dataset about ‘Justice’. Second, An ontology design and a knowledge graph compatible with the dataset. Third, a system that recommends relevant precedents/statutes. The whole Github repository can be found at <https://github.com/kongkongYuki/YuXin>, including video demonstrations of the system.

# Acknowledgments

First of all, I would like to thank my family. They have provided me with the financial support to enable me to achieve my dream of studying at Trinity College Dublin. Then I would like to thank my supervisor Professor Declan O' Sullivan for his guidance throughout the year. His rigorous style helped me to advance my dissertation regularly, his clear thinking helped me to clarify the direction of each stage of my work, and his serious attitude helped me to constantly correct and improve myself and my dissertation. I would also like to thank my landlords who showed me the friendly and charming country of Ireland and took care of me in my daily life. The peers I met during the year added to the beauty of my life. I am grateful for the study experience at Trinity College Dublin.

YU XIN

*University of Dublin, Trinity College  
August 2022*

# Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.1.1 Law in Ireland . . . . .	1
1.1.2 Mainstream search platform . . . . .	2
1.2 Motivation . . . . .	3
1.3 Research Objectives . . . . .	5
1.4 Research Methodology . . . . .	5
1.5 Expected Contribution . . . . .	6
1.6 Outline of Dissertation . . . . .	6
<b>Chapter 2 State of the Art</b>	<b>7</b>
2.1 Background . . . . .	7
2.1.1 Web crawler . . . . .	7
2.1.2 Natural Language Processing . . . . .	8
2.1.3 Knowledge Graph . . . . .	10
2.1.4 Graph Embedding . . . . .	11
2.2 Closely-Related Projects . . . . .	12
2.2.1 Ontology Design . . . . .	13
2.2.2 Recommendation . . . . .	14
2.3 Summary . . . . .	16
<b>Chapter 3 Design</b>	<b>18</b>
3.1 System Design . . . . .	18
3.1.1 Overview of the Pipeline . . . . .	18
3.1.2 Knowledge Extraction . . . . .	19

3.1.3	Knowledge Integration . . . . .	21
3.1.4	Recommendation algorithm . . . . .	21
3.2	Ontology Design . . . . .	22
3.2.1	Design Scope and Principles . . . . .	22
3.2.2	Competency Questions . . . . .	23
3.2.3	Identification of Ontology . . . . .	24
3.2.4	Creation of OWL Files in Protégé . . . . .	26
<b>Chapter 4 Implementation</b>		<b>27</b>
4.1	Project Structure . . . . .	27
4.2	Dataset . . . . .	28
4.2.1	Dataset for Mapping . . . . .	28
4.2.2	Dataset for Training . . . . .	31
4.3	Ontology . . . . .	32
4.3.1	Mapping Tools and Files . . . . .	32
4.3.2	Database Visualization . . . . .	35
4.3.3	Competency Questions Query . . . . .	35
4.4	Knowledge Extraction . . . . .	38
4.4.1	create repository crawl . . . . .	39
4.4.2	create repository nlp . . . . .	40
4.4.3	create repository topic . . . . .	40
4.5	Knowledge Integration . . . . .	41
4.5.1	create repository link . . . . .	42
4.5.2	create repository dic . . . . .	43
4.6	Recommendation algorithm . . . . .	44
4.6.1	OpenHINE . . . . .	45
4.6.2	recommend . . . . .	46
4.6.3	process input case id . . . . .	47
4.6.4	execute graph query . . . . .	47
4.7	Summary . . . . .	48
<b>Chapter 5 Evaluation</b>		<b>50</b>
5.1	Ontology Evaluation . . . . .	50
5.2	Model Evaluation . . . . .	51
5.2.1	Hyperparameter Experiment Results . . . . .	52
5.2.2	Experiment Conclusion . . . . .	53
5.3	System Evaluation . . . . .	54

5.3.1	Topic speculation . . . . .	55
5.3.2	Analysis of the reasons for recommendation . . . . .	56
5.3.3	Recommendation efficiency . . . . .	56
5.4	Overall Evaluation . . . . .	58
5.4.1	Questionnaire . . . . .	58
5.4.2	Feedback . . . . .	59
<b>Chapter 6 Conclusions</b>		<b>60</b>
6.1	Primary Contributions . . . . .	60
6.2	Future Work . . . . .	60
6.3	Final Remarks . . . . .	61
<b>Bibliography</b>		<b>63</b>
<b>Appendices</b>		<b>66</b>
.1	OOPS report . . . . .	67
.2	Query Result . . . . .	68
.3	Questionnaire . . . . .	69
.3.1	Respondent 1: . . . . .	69
.3.2	Respondent 2: . . . . .	70
.3.3	Respondent 3: . . . . .	71
.4	System Interface . . . . .	72



# List of Tables

2.1	Blackstone NER: Entity Types and Examples[13]	9
4.1	case_all_with_topic_id.csv Format and Sample	29
4.2	citation.csv Format and Sample	30
4.3	instrument_link.csv Format and Sample	30
4.4	judge_link.csv Format and Sample	31
4.5	Model Input Format and Sample	31
4.6	Mapping engines comparison	32
4.7	HetG embedding model comparison on ACM dataset[4]	44
5.1	Experiment 1: num_walks=50, walk_length=100, epoch =25	52
5.2	Experiment 2: num_walks=70, walk_length=100, epoch=15	52
5.3	Experiment 3: num_walks=100, walk_length=70, epoch=10	52
5.4	Experiment 4: num_walks=100, walk_length=50, epoch=10	53
5.5	Experiment 5: num_walks=100, walk_length=100, epoch=10	53
5.6	Topic speculation	55

# List of Figures

1.1	Court Service . . . . .	2
1.2	IRLII . . . . .	3
1.3	BAILII . . . . .	4
2.1	General Information Extraction Architecture[7] . . . . .	9
2.2	MetaGraph2Vec Model . . . . .	12
2.3	Related paper 1: Semantic model for legal documents . . . . .	13
2.4	Related paper 2: Ontological representation of legal cases . . . . .	13
3.1	System Automation Pipeline Overview . . . . .	18
3.2	Semi-structured Date Example . . . . .	19
3.3	Unstructured Date Example . . . . .	20
3.4	Third Party Database Example . . . . .	21
3.5	Ontology Design Edition 1 . . . . .	24
3.6	Ontology Design Edition 2 . . . . .	25
3.7	Ontology Visualization Generated by WebVowl[22] . . . . .	26
4.1	Topic Distribution . . . . .	32
4.2	Class relationships in GraphDB . . . . .	35
5.1	Experiment 3 Statistic Figure . . . . .	53
1	Oops! Ontology Report . . . . .	67

# Chapter 1

## Introduction

This chapter will provide a general overview of the research. Background Section(1.1) corroborates that our motivation raised in Motivation Section(1.2) are realistic and reasonable, and thus further illustrate the value of our research objectives in Research Objectives Section(1.3). The methodology's outline structure and description are illustrated in Research Methodology Section(1.4). Expected Contributions are listed in Section 1.5. The last section(1.6) is a synopsis of the whole dissertation.

### 1.1 Background

#### 1.1.1 Law in Ireland

Common Law, also known as judge-made law, is the body of law created by judges and similar quasi-judicial tribunals as stated in written opinions[9]. It is characterized by Case Law, i.e., repeated references to decisional precedents (precedent) that eventually produce universal, agreed-upon laws (customary rules) similar to moral concepts in general. One-third of the world's population lives in common law jurisdictions or systems mixed with civil law, including Ireland[18].

The Constitution of Ireland, which was enacted in 1937, is a written constitution of Ireland. The Constitution is the fundamental Law of the State. Stature Law comprises bills passed by the Oireachtas and signed into law by the president. Common Law is originally developed from courts' decisions based on traditional custom and precedent. It establishes the principles other judges follow in other cases. The courts' decisions have given substance to many constitutional rights. For example, the courts have held that people have to protect others who are likely to be affected by their actions, allowing people who have been injured to sue whoever is responsible for the injury caused. European Law is made of treaties, decisions of the European courts, and a series of other provisions, including

regulations and directives. It has priority over Irish national law. International Law is the treaties and agreements between different countries, which an act of the Oireachtas should implement[30].

Common-law legal systems attach importance to deciding cases according to consistent principles to ensure similar facts will lead to predictable outcomes. The principle by which judges are bound to precedents is stare decisis. The Irish courts follow the doctrine of stare decisis[9] to apply clear precedents set by higher courts and courts of co-ordinate jurisdiction.

### 1.1.2 Mainstream search platform

The common law system relies heavily on precedents, which can be concluded from Section 1.1.1. Therefore, how to appropriately quote citations will be the daily routine of legal staff. Legal practitioners use specialized case search platforms to find the relevant legal basis. The authors surveyed the current public mainstream Irish case law database as follows:

**Courts Service:** The Courts Service is the body responsible for administering the courts and supporting the judiciary in Ireland. It is an independent State Agency established under the Courts Service Act 1998[29]. Their official website has a case search function, and the interface is shown as Figure 1.1.

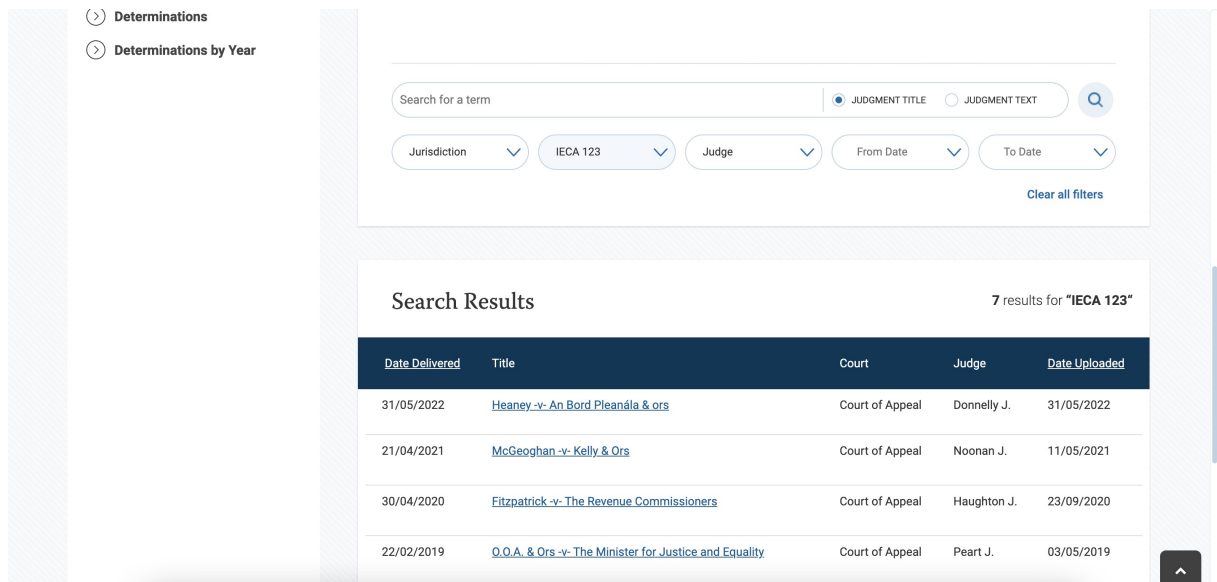


Figure 1.1: Court Service

When using the filter option "Neural Citation", the results presented cases with the number "IECA 123" over the years. The platform is unique in that it offers term matching for text content.

**IRLII (Irish Legal Information Initiative):** IRLII, hosted by UCC School of Law and sponsored by the Arthur Cox Foundation, is the Irish hub of the global Free Access to Law Movement (FALM). FALM is committed to providing legal decisions to all people on a non-profit basis[16]. The interface of IRLII is shown as Figure 1.2.

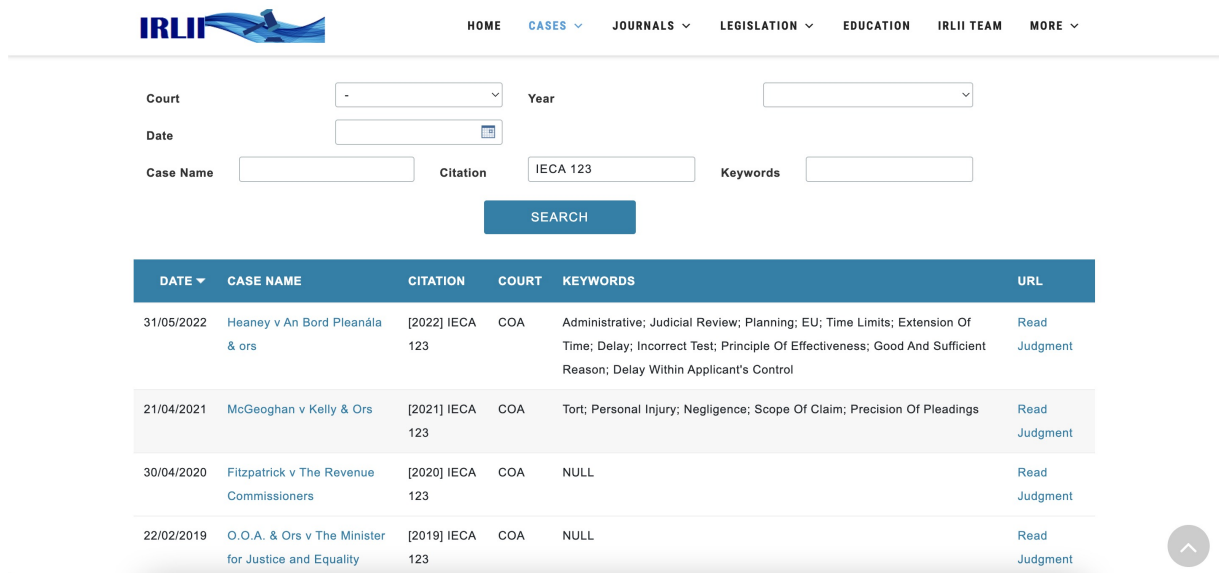


Figure 1.2: IRLII

When using the filter option "Neural Citation", the results presented cases with the number "IECA 123" over the years. The platform is unique in that it summarizes the keywords for each case and provides keyword matching.

**BAILII (British and Irish Legal Information Institute):** BAILII, hosted by the Institute of Advanced Legal Studies, London and the Law Faculty, University College Cork, provides free access to the most comprehensive set of British and Irish primary legal materials. The interface of BAILII is shown as Figure 1.3.

It can be said that the mainstream free platform for legal text retrieval is based on keyword matching and tag filtering.

## 1.2 Motivation

The law permeates all aspects of people's lives, even if people are often unaware of it. However, the law continues to be a highly specialized discipline. Although countries provide legal aid services to their citizens to help them defend their rights, these legal aid services usually have to meet specific requirements. For example, the Legal Aid Board

Figure 1.3: BAILII

The platform offers a rich set of search terms and rules. The platform has the richest material but does not include cases from the "Court of Appeal" established in Ireland on October 28, 2014.

only provides legal aid and advice to Irish people whose financial situation must fall below certain limits and whose case must be on the merits.[15].

Ireland is a common-law country and thus relies heavily on legal precedents to make a judgment. Legal precedents are essential for all judicial workers since they have to cite them as a basis for a judgment. Also, for citizens who do not know the law, it is necessary but challenging to have a brief understanding of their lawsuits. In most cases, blind as a bat, they must seek the help of an attorney who takes full charge of their cases.

Existing solutions are based on information retrieval techniques, which means people have to use specialized terminology as the term and plenty of manual screening to find the proper relevant cases. This process requires the expertise of lawyers to understand the lawsuits, mine for the most appropriate search keywords, and compare the lawsuit and the candidate sets. It is a very rigid search that usually takes considerable time and effort from legal practitioners and is particularly unfriendly to non-legal professionals.

Although some fee-based platforms have compelling features, such as JustisOne[34] and Lexis[20], they target law firms and university law departments as their customers and profit from their services. The application of the law, an essential means of self-protection for people, remains in the hands of lawyers and does not reach the masses. People have difficulty retrieving information to get easy and timely advice for their situations.

This research tries to simplify the search process for relevant cases and provide a lower threshold for ordinary people to be informed of their cases. Therefore, the motivation of

the research is to apply the knowledge graph to the legal field to optimize the current case search experience of public platforms.

### 1.3 Research Objectives

Regarding the word 'optimize' in the motivation mentioned above, a system will be created to recommend highly relevant precedents based on neural citations, which is our research objective. The target system can be divided into three modules to achieve the final recommendations, knowledge extraction to collect the data, knowledge integration to build the graph and recommendation algorithm to provide case feedback. Therefore, the research has the following three sub-objectives under this system:

- Relation and attribute extraction from semi-structured and unstructured data.
- Establish and evaluate a knowledge graph based on ontology design.
- Leverage the graph embedding model to recommend precedents.

### 1.4 Research Methodology

The entire three modules will form a pipeline as a whole. Each module applies different techniques, languages, models, tools, data formats, and protocol standards.

**Knowledge extraction:** Web crawler techniques and Natural Languages Processing(NLP) techniques are used in the knowledge extraction module to extract relations and attributes from semi-structured and unstructured data. Data is saved in tabular form in this phase.

**Knowledge integration:** Protégé tool in Web Ontology Language(OWL) and R2RML tool in Resource Description Framework(RDF) with Turtle language are used in ontology construction. Data is saved in GraphDB in this phase. SPARQL, an RDF query language, supports basic data manipulation.

**Recommendation algorithm:** The latent Dirichlet allocation model is used for dataset labelling, so the MetaGraph2Vec model can be trained on the dataset to generate graph embedding with cosine similarity as a recommendation algorithm.

## 1.5 Expected Contribution

According to Section 1.3, our three sub-objectives show a cascading relationship with each module corresponding to a phased output to achieve our final recommendation goal. Therefore this research will have three contributions as follows:

1. A small Irish Court Case Dataset about 'Justice'.
2. An ontology design and a knowledge graph compatible with the dataset.
3. A system that recommends relevant precedents/statutes.

Moreover, the final evaluation shows that the system can generate appropriate recommendations in the legal field with promising prospects.

## 1.6 Outline of Dissertation

The structure of the dissertation is as follows: chapter 2 will review the technical background and closely related projects, listing the techniques covered in this paper and the literature referenced. Chapter 3 discusses the design of the system and ontology, including the further elaboration of the research methodology and design choices employed in creating the ontology. Chapter 4 discusses the implementation of the whole system, detailing the steps and codes taken to build up the automation pipeline. Chapter 5 evaluates the completed system at three levels: ontology, model, and system. Finally, chapter 6 concludes the research and discusses the project's shortcomings and future work.



# Chapter 2

## State of the Art

This chapter presents the situation in which the research is located from a technical and project perspective. Section 2.1 provides information on the technologies and tools used. Section 2.2 lists the projects that are closely related to this research topic and provides a short description of each project. Section 2.3 summarizes the background and project part and compares those projects to this research.

### 2.1 Background

#### 2.1.1 Web crawler

Crawling is the process of automatically exploring web applications. Web crawlers, also called spiders, are the software that resides on the host and programs that get web pages from the Web by following hyperlinks. The web crawler aims to discover the web pages of a web application by navigating through the application and achieving the Web according to some strategies. By simulating user interactions[23] like sending GET, POST and HEAD instructions that follow the HTTP protocol, web crawl receives the document, including HTML and XML documents and other multimedia information from web servers. Technically, Web crawlers are the tools for data acquisition.

**Google Chrome Driver** Google Chrome is a web browser, and Chrome Driver is a tool for automation testing of web applications. It provides functions such as web browsing, user input, and JavaScript execution. The primary purpose of the Chrome Driver is to launch Google Chrome.

**Selenium** The Selenium Project[28], responsible for Selenium, is a range of tools and libraries that enable and support automation for web browsers. The Selenium suite

of components works across different browsers, platforms, and programming languages to emulate user interaction with browsers without the need to learn a test scripting language.

**Beautiful Soup** Beautiful Soup is a Python library for web scraping, extracting data from HTML and XML files. It works with parsers to provide idiomatic methods for navigating, searching, and modifying parse trees from page source code and can be used to extract data in a hierarchical, more readable manner.

## 2.1.2 Natural Language Processing

Natural Language Processing (NLP) includes a range of computational techniques for representing and automatically analyzing human language. The importance of NLP is that over 20 billion web pages in the WWW can be used as a vast data resource from which meaningful information can be found simply through NLP. The NLP field focuses on machine translation, information retrieval, information extraction, question answering, text summarization, topic modelling, opinion mining, and other areas. The current approaches to NLP are based on the syntactic representation of text, i.e., relying on the word occurrence frequencies, because semantics is considered a more challenging problem[6].

### Information Extraction

Information extraction systems input natural language text and produce structured information. There are various sub-tasks of IE, such as Named Entity Recognition, Coreference Resolution, Named Entity Linking, Relation Extraction, and Knowledge Base reasoning.[31].

**Named Entity Recognition** Named Entity Recognition (NER) is the problem of locating and classifying important nouns and proper names in a text[24]. For example, in the following sentence, bolded named entities hold critical information and are helpful for language processing applications: "JUDGMENT of **Ms. Justice Donnelly** delivered on the **28th day of October 2021**".

**Blackstone** [13] Blackstone is a spaCy model and library for long-form, unstructured legal text from ICLR/D, the Institute of Law Reporting Research Laboratory for England and Wales. The Named-Entity Recogniser(NER) component of the Blackstone model has been trained to detect the following entity types:

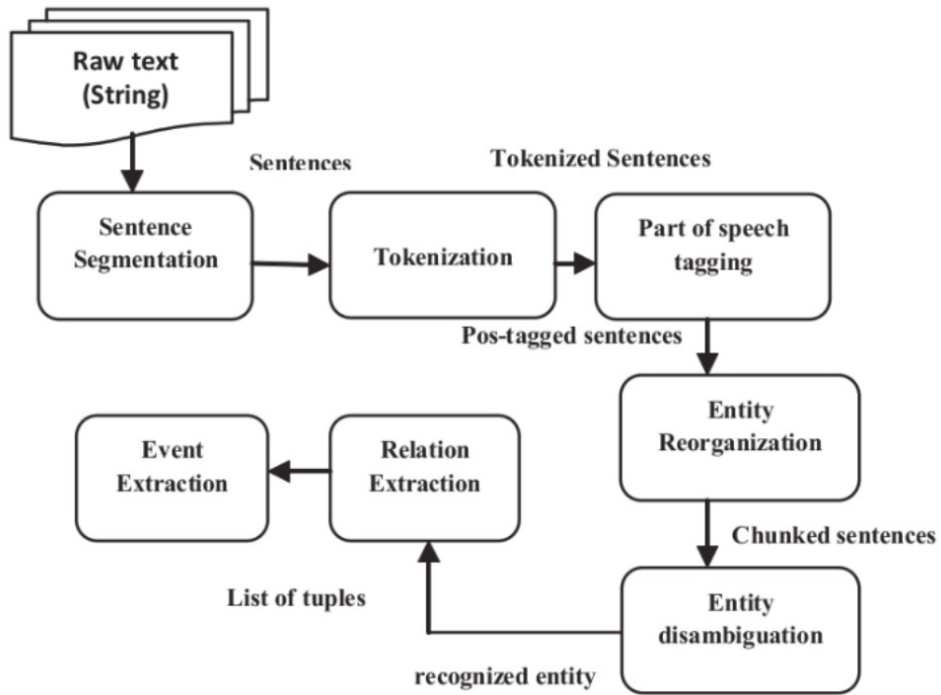


Figure 2.1: General Information Extraction Architecture[7]

Ent	Name	Examples
CASENAME	Case names	e.g. Smith v Jones, In re Jones, In Jones' case
CITATION	Citations (unique identifiers for reported and unreported cases)	e.g. (2002) 2 Cr App R 123
INSTRUMENT	Written legal instruments	e.g. Theft Act 1968, European Convention on Human Rights, CPR
PROVISION	Unit within a written legal instrument	e.g. section 1, art 2(3)
COURT	Court or tribunal	e.g. Court of Appeal, Upper Tribunal
JUDGE	References to judges	e.g. Eady J, Lord Bingham of Cornhill

Table 2.1: Blackstone NER: Entity Types and Examples[13]

## Topic Modeling

Topic models work on connecting documents with similar patterns through the patterns of word usage. Documents can be considered as mixtures of topics, where a topic is a probability distribution over words. Topic models rely on the bag-of-words assumption and ignore information from the ordering of words[2].

**Latent Dirichlet allocation (LDA)** Latent Dirichlet Allocation (LDA) is based on statistics (Bayesian) and is a widely used text mining algorithm that belongs to generative and unsupervised topic models in which words are collected into documents. The occurrence of each word is attributed to a topic in the document[2].

### 2.1.3 Knowledge Graph

#### Ontology

Ontologies are knowledge bases of concepts and their relations, illustrated in a formal logic-based knowledge representation language. Ontologies are the tools for data integration, sharing and discovery.

**Web Ontology Language(OWL)** [37] OWL became a W3C standard in 2004, and a revised version was created in 2012. OWL is based on description logic, a sublanguage based on first-order predicate logic that only uses singular and dual predicates and restricts the use of quantifiers, and is designed in such a way that the logical deductive reasoning of the language is decidable[12].

**Protégé** [27] Protégé is an ontology editor and a knowledge management system. It is widely popular and used worldwide as an ontology editor.

#### Semantic Web

The Semantic Web is considered an enhancement of the World Wide Web with machine-understandable information and services that use that information. Machine-understandable information is achieved by giving the data expressive metadata. In the Semantic Web, metadata is usually in the form of ontologies that allow reasoning over the meaning of the data[12]. In 2012 Google introduced the concept of the term Knowledge Graph, which is primarily a new framing of ideas coming directly out of the Semantic Web field[12]. The W3C standards formed around RDF, OWL and SPARQL establish technical interchange formats that are unified at the syntactic and semantic levels (to some extent). The application of knowledge graphs aims to establish effective methods for data sharing, discovery,

integration and reuse.

**Resource Description Framework(RDF)** [38] RDF became a W3C standard in 2004 and was revised in 2014. RDF is a syntax for expressing directed graphs, labelled type graphs by using OWL to specify ontologies of types and their relations, which are then used as types in RDF graphs and relations as edges, more or less compatible with OWL. From this point of view, OWL ontologies can be used as schemas for RDF graphs[12].

**R2RML** [35] R2RML became a W3C standard in 2010 and was revised in 2012. R2RML is a language expressing custom mappings from relational databases to RDF datasets. This mapping provides the ability to view existing relational data in the RDF data model to map structural and target vocabulary expressions chosen by the author. R2RML mappings themselves are RDF graphs, written down in Turtle syntax.

**SPARQL** [36] SPARQL is the W3C's RDF query language standard for adding, deleting and retrieving data from RDF triplet stores/graph databases. SPARQL queries can not only match the subject-predicate-object triples, but can also use mathematical operations and functions to create filter conditions and new variable bindings. Results can be sorted, grouped, and these groups can be aggregated.

**GraphDB** [25] GraphDB is a semantic graph database following W3C Standards. Semantic graph databases, also called RDF triplestores, use graph structures for semantic queries to represent and store data with nodes, edges, and attributes. It is easy to visualize relationships in GraphDB, which is very useful for large amounts of interconnected data.

## 2.1.4 Graph Embedding

Most graph methods suffer the high computation and space cost issues, while graph embedding can transform the graph data (nodes, edges and features) into a low dimensional space in which graph structure and information are maximally preserved. The input of graph embedding varies in different scenarios according to the types of graphs (homogeneous graph or heterogeneous graph). The output of graph embedding is a low-dimensional vector representing a part of the graph(or a whole graph). By representing a graph as a low dimensional vector, other algorithms can then be computed efficiently on the graph. [5].

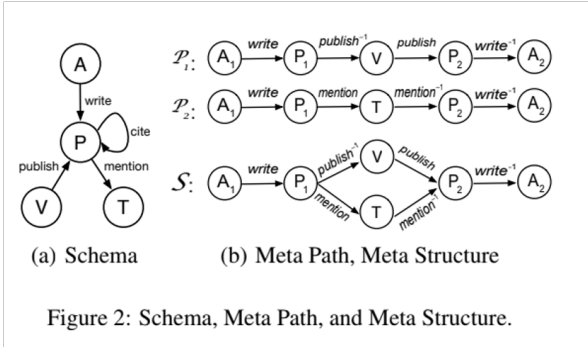


Figure 2: Schema, Meta Path, and Meta Structure.

**Algorithm 1** The MetaGraph2Vec Algorithm

**Input:**

- (1) A heterogeneous information network (HIN):  $G = (V, E)$ ;
- (2) A metagraph:  $\mathcal{G} = (N, M, n_s, n_t)$  with  $n_s = n_t$ ;
- (3) Maximum number of iterations:  $MaxIterations$ ;

**Output:**

- Node embedding  $\Phi(\cdot)$  for each  $v \in V$ ;
- 1:  $S \leftarrow$  generate a set of random walks according to  $\mathcal{G}$ ;
  - 2:  $\mathbb{F}(v_i, v_j) \leftarrow$  count frequency of node context pairs  $(v_i, v_j)$  in  $S$ ;
  - 3:  $Iterations \leftarrow 0$ ;
  - 4: **repeat**
  - 5:    $(v_i, v_j) \leftarrow$  sample a node context pair according to the distribution of  $\mathbb{F}(v_i, v_j)$ ;
  - 6:    $(\Phi, \Psi) \leftarrow$  update parameters using  $(v_i, v_j)$  and Eq. (10);
  - 7:    $Iterations \leftarrow Iterations + 1$ ;
  - 8: **until** convergence or  $Iterations \geq MaxIterations$
  - 9: **return**  $\Phi$ ;

(a) Meta Structure

(b) MetaGraph2Vec Algorithm

Figure 2.2: MetaGraph2Vec Model

**Meta Structure** [14] Meta Structure is a directed acyclic graph of object types with edge types connecting in between to measure the proximity between objects. The advantage of the meta structure is that it can describe the complex relationship between two heterogeneous information networks (HINs) like two papers share the same topics and authors. An example of Meta Structure is shown in Figure 2.2a, where it integrates two paths as a whole to represent the schema more comprehensively.

**MetaGraph2Vec Algorithm** [41] In the MetaGraph2Vec algorithm, the deep walk algorithm randomly generates paths from one node to another at a certain length. In this way, the model transforms the graph into the combination of the selected meta paths, each of which has a similar structure where nodes are treated as words in a sentence. Therefore, we can apply the Skip-gram algorithm, a text embedding model originating from Word2Vec. In Figure 2.2b, line 1 generates a set of meta graph-guided random walks. Line 2 does the embedding work. The code block from 4 to 8 is a stochastic gradient descent to learn the hyperparameters.

In brief, **MetaGraph2Vec** = **Meta Structure** (data structure) + **Random Walk** (create random paths which is similar to sentences) + **Skip-gram** (embedding model originated from Word2Vec).

## 2.2 Closely-Related Projects

Each of the projects surveyed in this section are given a number, so that these projects can be referenced more easily in subsequent chapters of the dissertation.

## 2.2.1 Ontology Design

- Using Mapping Languages for Building Legal Knowledge Graphs from XML files[17]:

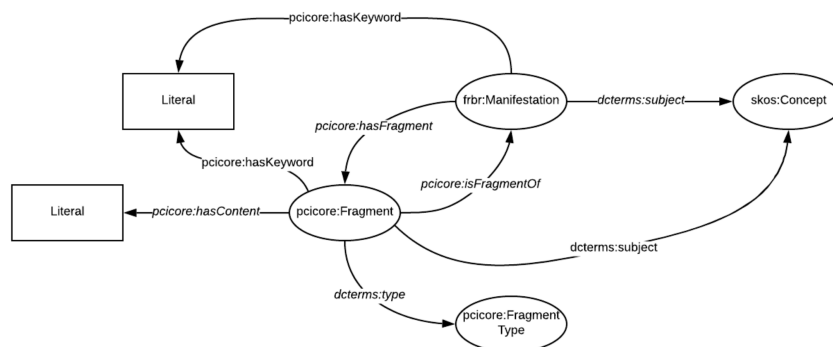


Figure 2.3: Related paper 1: Semantic model for legal documents

This paper presents the experience of building RDF knowledge graphs for an industrial use case in the legal domain. The semantic model is shown as Figure 2.3 with fragments like manifestation, content, keyword, type and concept. The authors reuse existing ontologies, including Functional Requirements for Bibliographic Records (FRBR), the Simple Knowledge Organization System (SKOS) and the Dublin Core (DC) terms ontologies and compare various mapping engines (XSPARQL, SPARQL-Generate, RML-Mapper, CARML).

- A legal case OWL ontology with an instantiation of Popov v. Hayashi[40]:

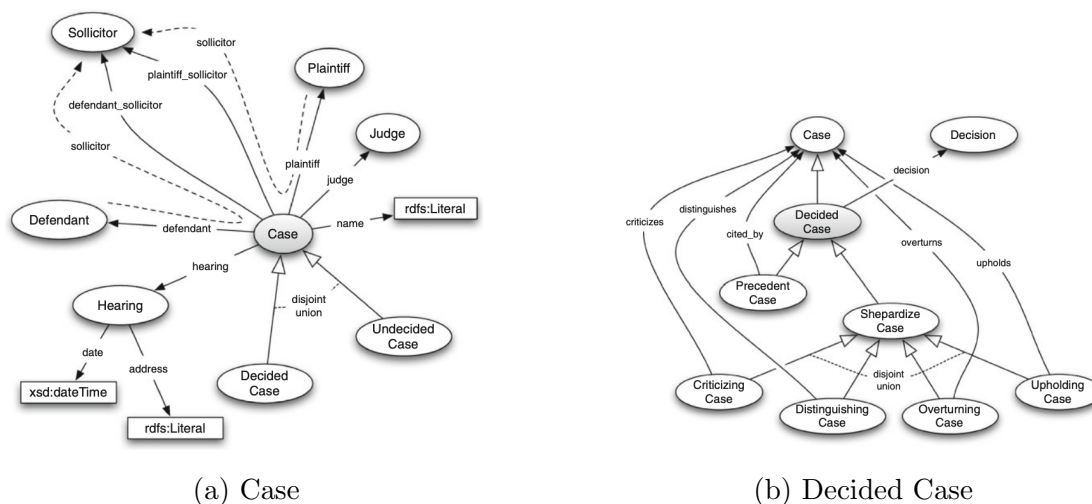
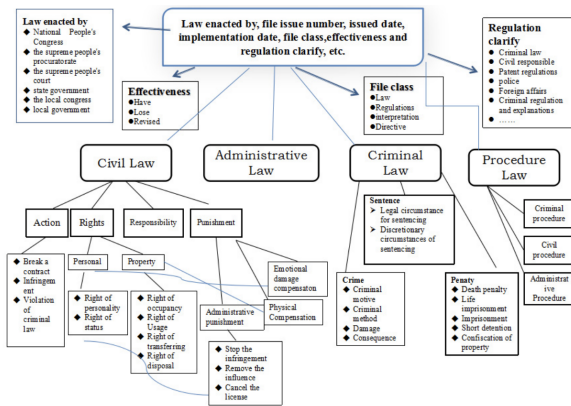


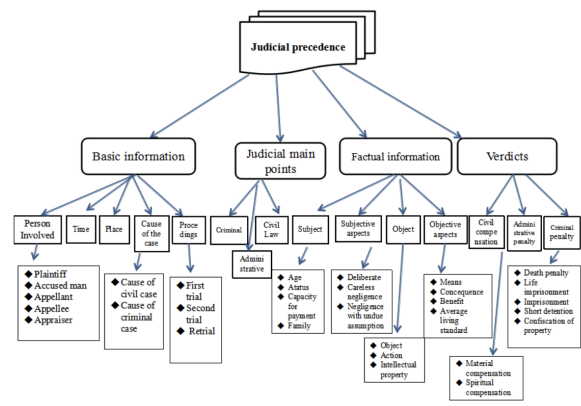
Figure 2.4: Related paper 2: Ontological representation of legal cases

The paper provides an OWL ontology for the legal case Popov v. Hayashi. The ontological design of this paper is comprehensive and professional. The authors provide a detailed reading of the judgement, with different types of cases possessing different ontological designs and complete classification of testimony, evidence, and legal texts. The ontology of this paper is excellently designed, but its excessive granularity makes it challenging to form automatic pipelines for general use in multiple cases.

### 3. An Ontological Chinese Legal Consultation System[42]:



(a) Chinese statutes ontology



(b) Judicial cases ontology

This paper constructed a detailed statutes ontology and a case ontology, respectively and then performed experiments using emotional damage compensation as an example by combining genetic algorithm and k nearest neighbor approaches to give similarity of the cases and statutes, respectively. Since China is a civil law system, the user input will first enter into a word segmentation system, which means that the system's core is based on the similarity of words.

## 2.2.2 Recommendation

### 4. Extracted Summary Based Recommendation System for Indian Legal Documents[33]:

This paper wants to prove that a concise, summarized version of the case is a non-trivial task, so the authors use only the extracted summary of legal documents to retrieve similar documents. Doc2Vec embeddings are used to transform articles into word vectors, and Support Vector Machines are used for classification tasks. The results show a better F1-Score, and the time and space complexity is reduced compared to the entire documents.



5. Text-guided Legal Knowledge Graph Reasoning[21]:  
This paper aims to predict the related legal provisions of affairs through a knowledge graph completion task and a text-guided graph reasoning approach. The authors use an instance encoder with BERT and an MLP layer to reduce the dimension of features; employ GNN to learn explicit relational knowledge; utilize TransE, DistMult and SimpleE as triple scores functions.
6. Quick Check: A Legal Research Recommendation System[32]:  
This paper aims to design a system that extracts the legal arguments from a user’s brief and recommends highly relevant case law opinions. The system leverages a multitude of case discovery pathways and ranking models trained over a considerable annotated training set to extract the most relevant cases to a given legal issue. The technical approach is search-engine-based candidate discovery + citation-based candidate discovery with two ranking SVM models.
7. Citation Recommendation with a Content-Sensitive DeepWalk-based Approach[10]:  
This paper calculates similarities between the target and candidate papers at the node content level, selecting an initial seed set of papers. At the citation network structure level, this paper exploits the citation relationships between papers to study the latent representation of the scientific papers based on a deep natural language method, Deep Walk. These two levels are executed in order.
8. Scientific Article Recommendation by using Distributed Representations of Text and Graph[11]:  
This paper proposes a model which combines network embeddings with the semantic embeddings of the text using CCA. CCA is a popular method to combine two different modalities by projecting both in a common representation space to maximize the correlation between both modalities in the common space. Text similarity methods are TF- IDF and BM25 ranking-based methods on paper’s content. Representation learning methods(network embeddings) are DeepWalk(graph), LSI, LDA, word2vec, doc2vec (text). The authors try to find a common space for the two areas.
9. IntelliLegalRec: An RDF Based Metadata Driven Semantically Compliant Recommendation System for Socio-legal Judicial Documents[19]:  
The paper model is based on topics, which rely on the Latent Dirichlet Allocation(LDA) model and treat topics as metadata. Also, the paper uses an existing dataset to train/evaluate the recommendation model, but maps the recommendation results to the local dataset in some unknown way. The author calculates the

similarity between METADATA from users and CLASS LABEL from datasets to recommend documents.

## 2.3 Summary

Crawler technology allows this research to obtain data from official government sources as a free agent. Natural language technology provides a solution to extract information from judgment texts and model textual topics. Knowledge graph technology transforms tabular data into graph data, accelerating information retrieval and enhancing information utilization. Graph embedding technology extracts features from complex graphs and makes it possible to calculate node similarity effectively.

Project 1 provides a comparison of mapping tools and a reference to available libraries. Project 2 gives a delicate reference to ontology design, leading to automation difficulties. Project 3 references ontology design and system construction, but the problem are that its object is a civil law case. Project 4 is not to make recommendations but to demonstrate the improvement in results from using a summary. However, this paper’s approach (word embedding + SVM classification) inspires the research. Project 5’s approach is graph reasoning(BERT + GNN + TranE/DistMult/SimplE). The ontological design is not explicitly solidified but based on the features extracted and learned by BERT and GNN. The idea of the system is very similar but different from the research in terms of information extraction and feature processing. Project 6 utilizes a two-layer SVM model for recommendations, incorporating information retrieval techniques. The second layer of Project 7 has a homogeneous rather than heterogeneous graph of citation relations, and the first layer directly calculates the similarity of contents rather than extracting information as in this research. Project 8 combines two modalities(graph and text) by projecting both in a common representation space to calculate similarity. The techniques used in Project 9 are very similar to this research but are used differently, and the system architecture is different.

The research differs from closely related projects in the following areas:

- Different ontology objects and design: Irish cases focused on citation
  - Facts of the case (Project 2)
  - Legal provisions (Project 3)
- Different way to recommend: Graph embedding on the heterogeneous graph
  - Word Embedding + SVM on text information (Project 4, 6)

- DeepWalk on homogeneity graph (Project 7)
- GA-KNN algorithm on text information (Project 3, 5)

The granularity of the research's ontology design is not as refined but more about application-level feasibility. Highly granular ontologies are usually modelled from one or two individual cases, while the research's model needs to be formed from a broader range of cases. As for the recommendation, most of them use word embedding rather than graph embedding. Those using graph structure tend to be homogeneous graphs; there is only one node type and one relationship type in the graph. The research takes case, judge, jurisdiction, and citation into account.

# Chapter 3

## Design

This chapter includes system design and ontology design. System Design Section(3.1) is a further elaboration of the research methodology(Section 1.4), spelling out in detail the purpose, functions, steps and data input of each module. Ontology Design Section(3.2) emphasizes the principles that the ontology needs to focus on(3.2.1), presents 10 Competency Questions(3.2.2), gives the process of ontology design and final draft details description(3.2.3).

### 3.1 System Design

#### 3.1.1 Overview of the Pipeline

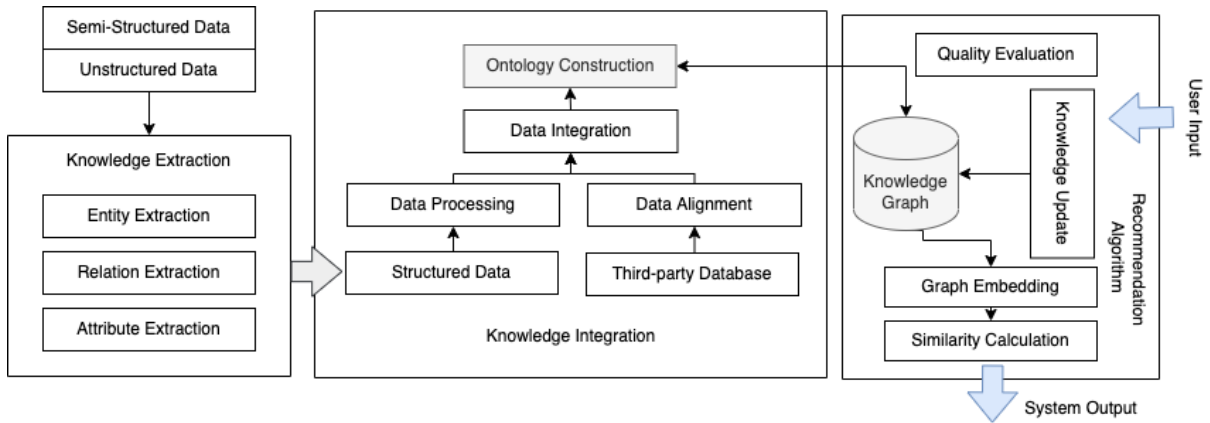


Figure 3.1: System Automation Pipeline Overview

The whole system can be described as an automation pipeline as Figure 3.1. In brief, three sub-modules (knowledge extraction, knowledge integration, and recommendation algorithm) undertake the tasks of collecting and processing the data, establishing the

dataset with labels and links, constructing the ontology by mapping files, embedding the knowledge graph and giving recommendations based on similarity scores. The user input is the neutral citation (official unique id of each case). The algorithm gives the recommended cases and statutes list, and related information is retrieved from the database as system output.

### 3.1.2 Knowledge Extraction

The knowledge extraction module is responsible for extracting entity, relation, and attribute from semi-structured and unstructured data to generate structured data for further processing. This module is mainly used to create repositories/datasets for the whole system.

The data source is courts.ie[29], the official website of the Irish law service as stated in Section 1.1.2. Since the official data interface is not provided and no existing data sets are available, the author has to gather the data through websites or platforms.

Most of the semi-structured data are embedded in the web page. Thus, crawler technology will extract web content, including case lists, case links, case files and especially case meta information of each case. One of the examples of semi-structured data can be found in Figure 3.2 where one can obtain neutral citation, judge, court, date, status, result, composition, title, which can be split into plaintiff and defendant, and case link information.

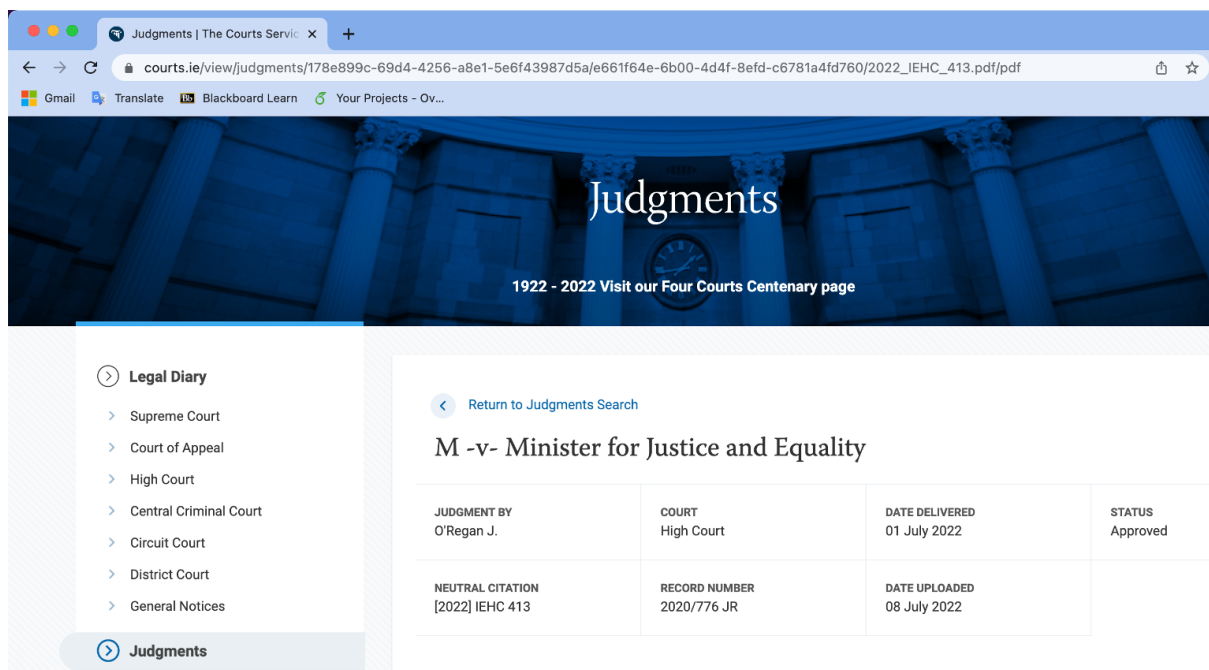


Figure 3.2: Semi-structured Date Example

Some cases are available as downloadable Docx files. The case content of these documents is presented in an unstructured manner, as Figure 3.3 showed. For the processing of legal texts, natural language techniques need to be approached to extract the entities and values, particularly reference relationships that are not available in semi-structured data, including precedent citation and legal provisions.

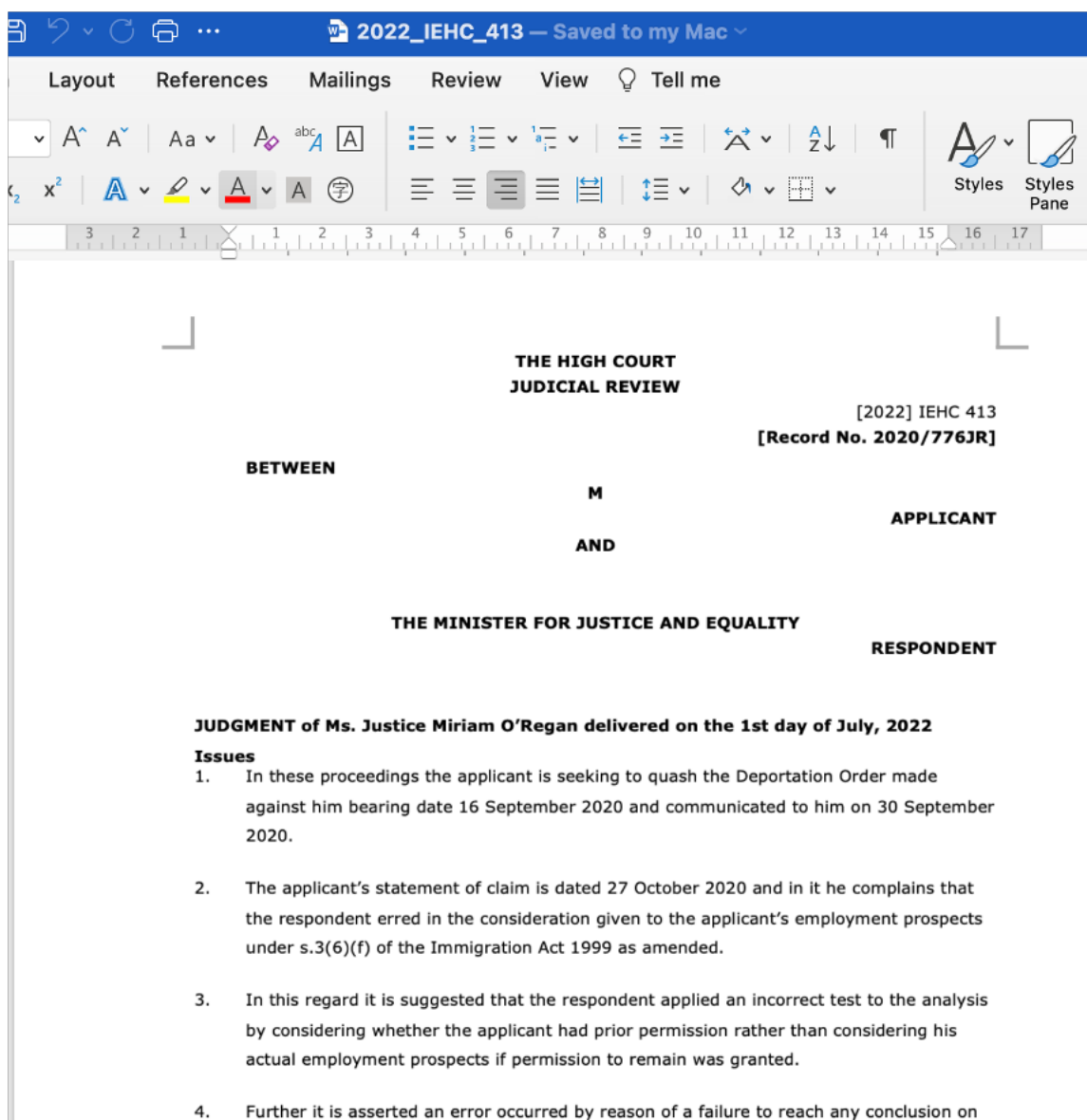


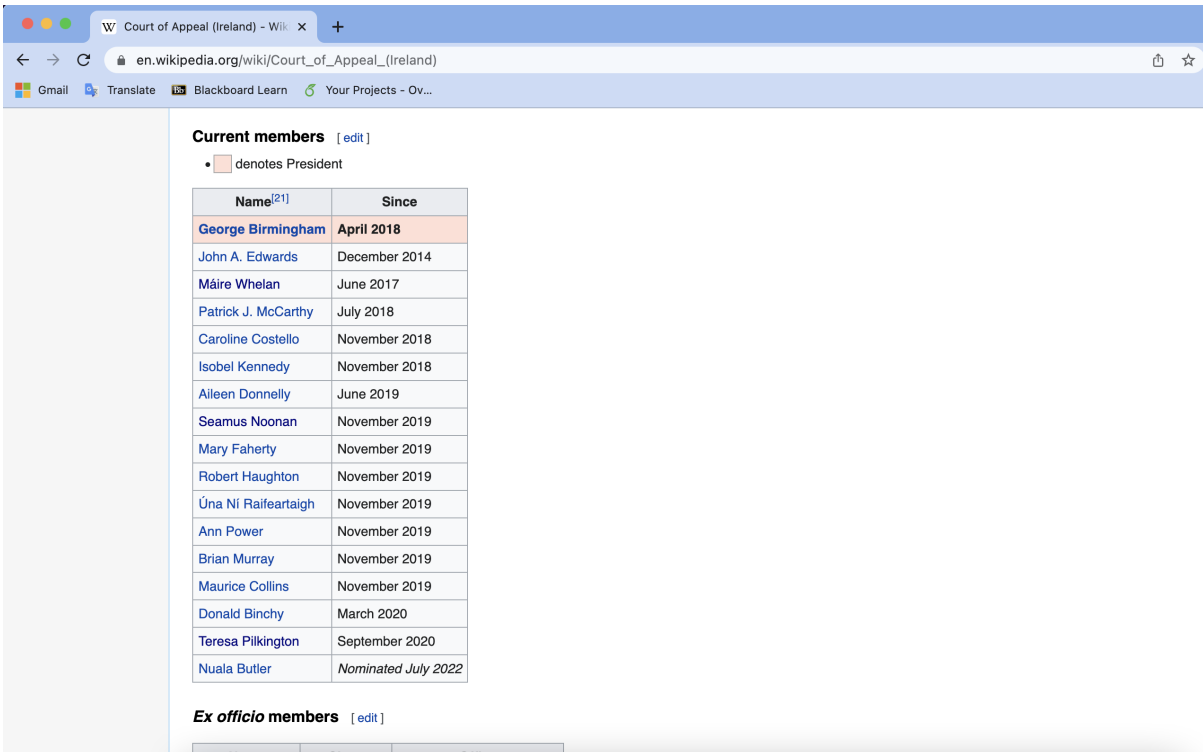
Figure 3.3: Unstructured Date Example

The graph embedding model is a semi-supervised model, so labels are needed for training to improve the model performance. This module also includes the preparation of data labels. An unsupervised topic model will be applied to all legal texts to determine the topic to which each case belongs.

### 3.1.3 Knowledge Integration

The knowledge integration module needs to split structured data(data processing), align our dataset with third-party data(data alignment), and then merge third-party information into our dataset(data integration). The final output of the module is fed into ontology mapping files. This module lays the foundation for establishing a knowledge graph based on ontology design.

The third-party database used by the authors is Wiki data, with personal profiles of former and current Irish judges categories by jurisdiction as shown in Figure 3.4. Furthermore, Electronic Irish Statute provides Book additional content on legal provisions. All third-party information is given in the form of links.



The screenshot shows a web browser window with the URL `en.wikipedia.org/wiki/Court_of_Appeal_(Ireland)`. The page content includes a section titled "Current members" with a legend indicating that a square icon denotes the President. Below this is a table with two columns: "Name<sup>[21]</sup>" and "Since". The table lists 16 members with their names and the date they started their term. The first row, "George Birmingham", is highlighted in orange and includes the text "April 2018".

Name <sup>[21]</sup>	Since
<b>George Birmingham</b>	<b>April 2018</b>
<a href="#">John A. Edwards</a>	December 2014
<a href="#">Máire Whelan</a>	June 2017
<a href="#">Patrick J. McCarthy</a>	July 2018
<a href="#">Caroline Costello</a>	November 2018
<a href="#">Isobel Kennedy</a>	November 2018
<a href="#">Aileen Donnelly</a>	June 2019
<a href="#">Seamus Noonan</a>	November 2019
<a href="#">Mary Faherty</a>	November 2019
<a href="#">Robert Haughton</a>	November 2019
<a href="#">Úna Ní Raifeartaigh</a>	November 2019
<a href="#">Ann Power</a>	November 2019
<a href="#">Brian Murray</a>	November 2019
<a href="#">Maurice Collins</a>	November 2019
<a href="#">Donald Binchy</a>	March 2020
<a href="#">Teresa Pilkington</a>	September 2020
<a href="#">Nuala Butler</a>	<i>Nominated July 2022</i>

Figure 3.4: Third Party Database Example

The mapping file will convert the data from table format to Resource Description Framework (RDF) format, which can be imported into the graph database with the ontology design to build a complete knowledge graph.

### 3.1.4 Recommendation algorithm

This module references the structural layers of Section 2.2 Project 4&6 but changes the model used for each layer; in particular, the word embedding is changed to a graph em-

bedding. Due to the structure of heterogeneous graphs, this module does not use the DeepWalk model as in Project 8 but a model for heterogeneous graphs.

The core idea of the recommendation algorithm module is to transform nodes, edges and features into vector space (a lower dimension) while maximally preserving graph structure and information. This module leverages the output of a graph embedding model to calculate the similarity scores of nodes to give final recommended cases and statutes.

In terms of user input, the author restricts it to neutral citations, i.e., the official unique ID of the case. User input under this restriction leads to two scenarios, one in which the neutral citations entered by the user exist in the database and the other in which they do not. The former will use a pre-trained model with faster feedback to give recommendations, while the latter needs to reacquire case-related information to generate nodes to join the existing knowledge graph and train the model with slow feedback. At the same time, the case information will be updated to the database and model (knowledge update). In the ideal situation, when we build up a large enough dataset, the latter schema will be used for daily back-end batch updates, so the time spent will not be reflected in the user's real-time retrieval process.

The recommendation algorithm module will give a list of recommendations, combined with other information fetched from the graph database and returned to the user as system output.

## 3.2 Ontology Design

### 3.2.1 Design Scope and Principles

**Target group:** The objective is to facilitate the user search experience on existing public platforms so that the system will target non-legal professionals more than law firms using paid platforms.

**Scope:** It should also be clear that the system is only aimed at cases in Ireland. Thus, the ontology should be designed to conform to the Irish judicial system, trial composition, case format, and other locally appropriate elements. For example, the design will focus more on customary law referencing relationships than on the statutory framework because of features of the Irish Common Law System.

**Granularity:** Ontology should be designed for non-legal professionals to understand, but at the same time, should also take specialization and expansion into account, which means the granularity of the ontology would be quite refined but still keep the basic crucial information.



**Compatibility:** Ontology should be reconciled with data and data type.

### 3.2.2 Competency Questions

Competency questions define a set of conditions that can be answerable once the ontology has been completed and characterise the problems that the ontology can solve. Ideally, capturing domain tasks as competency questions involves producing a set of questions that people might wish to learn from a knowledge base.

Competency questions should be able to cover the search items provided by the existing platform like date range, jurisdiction, and judge. However, they should also embody features specific to knowledge graphs, such as the connectivity relationships between entities. Since SPARQL language has mathematical operations and a wide range of utility functions to create filters and new variable bindings, competency questions should also include some calculation and filtering to demonstrate enhanced search capability. In addition, the relationship between Classes and the relationship between Properties in the ontology design should enrich the search function.

Competency questions will be used as part of the feedback survey evaluation criteria for legal and non-legal professionals as Section 5.4.2 planned. Competency questions will be reviewed and added to over time as feedback is gathered whilst demonstrating the initial system to legal experts in the future.

**(...) for entity, /.../ for variable placeholder, (...-...) for subclass relationship, # for annotation**

1. (Case) after the /certain day/. # Basic search item: Date
2. What is (Case) from (Jurisdiction)? # Basic search item: Jurisdiction
3. What is (Case) judged by (Judge)? # Basic search item: Judge
4. What are /approved/ (Case) on /Topic/?  
# Intermediate search function: Entity relationship
5. Which (Judge) deals with the most cases in /Year/?  
# Intermediate search function: Entity relationship
6. How many (Case) have contested? # Advanced search function: Calculation
7. What are failure rates on /Topic/? # Advanced search function: Calculation
8. The ratio of (Precedent) : (Statue) in (Case).  
# Advanced search function: Calculation

9. What is the most frequently used (Statue-Act)?  
# Ontology search function: Class and sub-class
10. What (Legal Document) can support my case on /Topic/?  
# Ontology search function: Class and sub-class

### 3.2.3 Identification of Ontology

Referring to Section 2.2.1 Project 2&3 , the author proposes the following ontology design as shown in Figure 3.5, which combines all the valuable features mentioned in the literature.

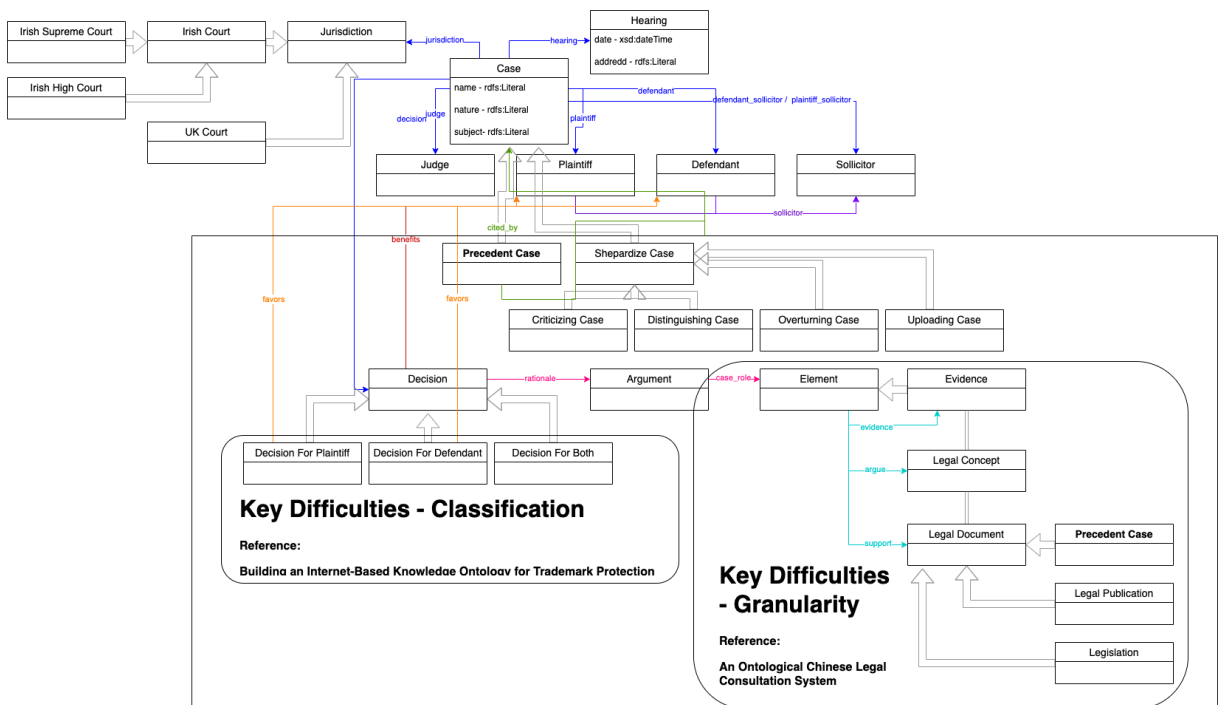


Figure 3.5: Ontology Design Edition 1

However, due to the premise of the common law system, where the weight of legal provisions is not as heavy as in civil law systems and the need for ubiquity and universality of ontology for large-scale data, this version of ontology needs to be pruned and modified even though it is a comprehensive one.

Then, the author proposes another ontology design, as shown in Figure 3.6. The ontology shifts attention from the hierarchical construction of legal texts to the construction of categories of cited objects.

After branching the ontology, the authors made minor modifications to the properties of entities and the relationships between entities based on the collected database, which

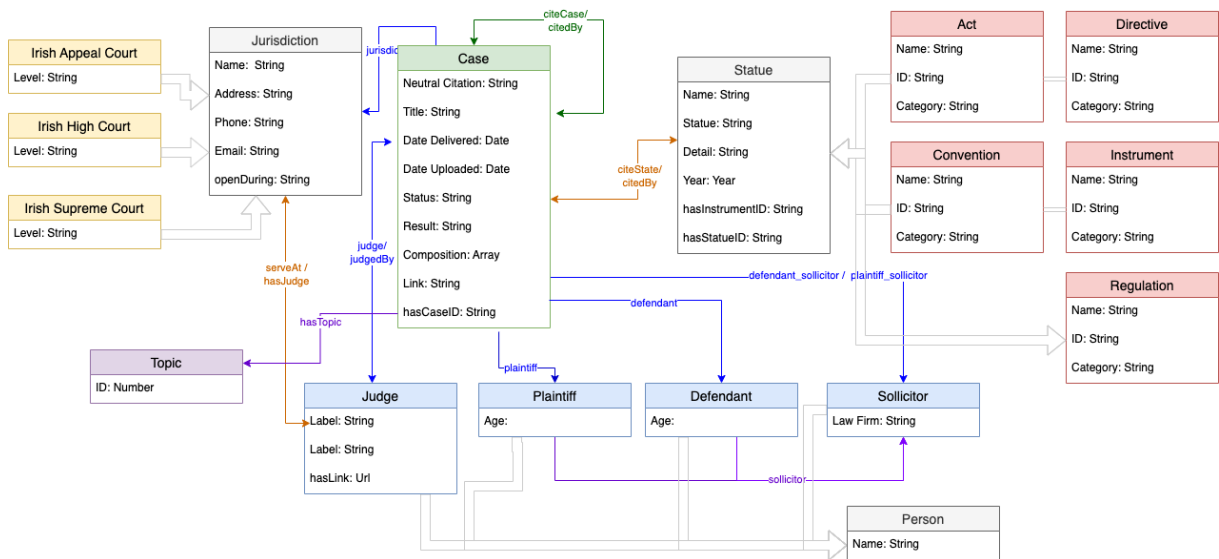


Figure 3.6: Ontology Design Edition 2

makes it more relevant to the application scenario and application value of the system. The data source lacks solicitor information, but the solicitor entity is retained here for the integrity of ontology for additional information in the future.

It has 16 Classes with 33 object properties. A detailed ontology document generated by WIDOCO[8] can be found in the project GitHub repository(<https://github.com/kongkongYuki/YuXin/blob/main/Ontology/myDocumentation/doc/index-en.html>).

### Classes

Among all classes, Case is the core of ontology. Three other classes(Jurisdiction, Person, Statute) are connected to Case. The class under Jurisdiction are disjoint with each other.

$$\{IrishAppealCourt, IrishHighCourt, IrishSupremeCourt\} \subset Jurisdiction$$

$$\{Defendant, Judge, Plaintiff, Sollicitor\} \subset Person$$

$$\{Act, Convention, Directive, Instrument, Regulation\} \subset Statue$$

### Properties

Among all properties, there are Subordinate relationships.

$$\{(Case)citeCase(Case), (Case)citeStatue(Statue)\} \subset cite$$

$$\{(Case)hasCaseID, (Statue)hasInstrumentID,$$

$$(Judge)hasJudgeID, (Statue)hasStatueID\} \subset hasGraphID$$

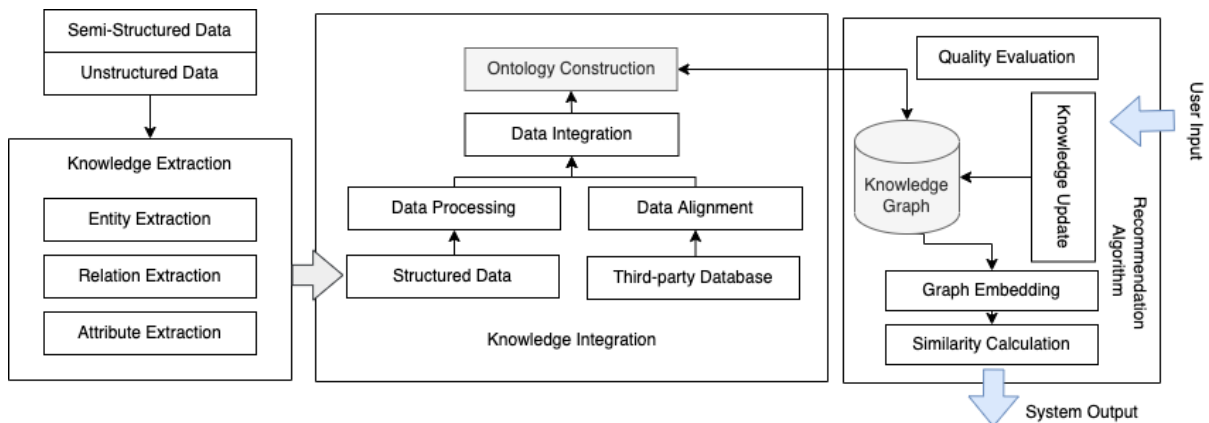


# Chapter 4

## Implementation

This chapter starts from the project structure (Section 4.1) and goes into the specific execution code in each file within each module (Section 4.4, Section 4.5, Section 4.6) according to the pipeline flowchart. Dataset Section (4.2) gives the scope and number of datasets, a description of the format of the dataset, the respective roles of the datasets, and the merging and transformation between datasets. Ontology Section (4.3) presents the key code in the mapping file and answers the Competency Questions raised in the last Chapter with SPARQL queries, using the data combined with the ontology in GraphDB.

### 4.1 Project Structure



The codes of the project are integrated into a python project, where `create_repository.py` is the entrance file to create our repository and `main.py` is the entrance file to run the system. The entire project structure corresponds to the pipeline chart as follows:

- Knowledge Extraction

- create\_repository\_crawl.py: Extract information from semi-structured data.
  - create\_repository\_nlp.py: Extract information from unstructured data.
  - create\_repository\_topic.py: Train the LDA model from unstructured data and classify the topic of each case.
- Knowledge Integration
    - create\_repository\_link.py: Split the nested arrays in the "Judge" and "Instrument" columns into other tabular tables. Then align the data from the third-party database and link the URLs (Output file: Instrument\_link.csv & Judge\_link.csv).
    - create\_repository\_dic.py: Split the nested arrays in the "Citation" column into another tabular table (Output file: Citation.csv).
- Ontology Construction
    - Ontology/ono.owl: Ontology design written with the help of Protégé.
    - Ontology/mapping.ttl: Mapping file to transform tabular data into RDF data.
- Recommendation Algorithm
    - OpenHINE Folder: Metagraoh2Vec model. What is needed is the final embedding matrix stored in the TXT file (Output file: test\_node.txt and graph\_rw.txt).
    - create\_dataset\_label.py: Convert the tabular data to TXT files according to the format of model input.
    - recommend.py: Calculate and rank the similarity scores.
    - process\_input\_caseid.py: Processes neutral citations/cases entered by the user that are not recorded in the database.
    - execute\_graph\_query.py: Script to interact with GraphDB.

## 4.2 Dataset

### 4.2.1 Dataset for Mapping

#### Case - case\_all\_with\_topic\_id.csv (primary dataset)

There are 1109 cases after dropping duplicates. More than 200 have 'Justice' in title from Aug 11th, 2021, to July 8th 2022. Furthermore, the remains have 'Justice' in text from

May 10th, 2022, to July 8th 2022. Upon observation, their neutral citations are nearly consecutive, meaning that this condition, ‘Justice in the text’, nearly encompasses all cases during that period. This situation proves that ‘Justice’ is not a good keyword for information retrieval, and it also means that our dataset will have a variety of topics. The format and sample of the dataset are in Table 4.1. Composition, Instrument, and Citation columns must be further processed and extracted into separate CSV files.

Column	Technique	Usage	Example
Judge	Crawler	NA	Whelan J.
Composition	Crawler	Ontology & Graph	Whelan J.; Noonan J.; Pilkington J.
Court	Crawler	Ontology	Court of Appeal
Date Delivered	Crawler	Ontology	2021-10-27
Date Uploaded	Crawler	Ontology	2022-02-23
Status	Crawler	Ontology	Approved
Result	Crawler	Ontology	Appeal Dismissed.
Neutral Citation	Crawler	Ontology & ID	2021_IECA_283
Title	Crawler	Ontology	C.M. -v- Health Service Executive
Plaintiff	Split Title	Ontology	C.M.
Defendant	Split Title	Ontology	Health Service Executive
Case Link	Crawler	Ontology	<a href="https://www.courts.ie/view/judgments/7772da48-9c1c-4765-bbb5-b9e07751161383d4c72-363b-4421-acfe-c911e3822eb2021_IECA_283.pdf/pdf">https://www.courts.ie/view/judgments/7772da48-9c1c-4765-bbb5-b9e07751161383d4c72-363b-4421-acfe-c911e3822eb2021_IECA_283.pdf/pdf</a>
CITATION	NLP	Ontology & Graph	[‘[1980] I.R. 251’, ‘[2005] Act’, ‘[1994] 1 IR 101’, ‘[2006] 1 I.R. 421’, ‘[2020] IESC 10’, ‘[2001] 4 I.R. 113’, ‘[1979] I.R. 326’, ‘[2021] IECA 101’]
INSTRUMENT	NLP	Ontology & Graph	[‘EPSEN Act’, ‘Disability Act’, ‘National Council for Special Education (’, ‘EPSEN Act 2004’, ‘Education Act’, ‘Disability Act 2005’, ‘Act of 2004’, ‘Health Acts 1947’, ‘2007 Regulations’, ‘2005 Act’, ‘2004 Act’, ‘Interpretation Act’, ‘Disability Act 57’, ‘Health Act’, ‘s. 8 Disability Act’]
Topic	LDA	Ontology & Label	3
Neutral Citation ID	UUID	Ontology & Graph	12062

Table 4.1: case\_all\_with\_topic\_id.csv Format and Sample

### Citation - citation.csv (secondary dataset)

In the main file, the column exists as an array. After expanding those arrays as rows and filtering out the invalid citations, there are 4821 rows in the Citation file. The format and sample of the dataset are in Table 4.2.

Column	Sample
Neutral Citation	2021_IECA_282
Citation	2013_IESC_6
Citation ID	12671

Table 4.2: citation.csv Format and Sample

### Instrument - instrument\_link.csv (secondary dataset)

Similarly, after expanding the Instrument column of the main file to rows of another file, an Instrument file with more than 6000 rows was generated. The automatic extraction of legal documents is affected by invalid data, such as the 1956 Act, so manual filtering is required to clean up the data.

There are 3111 rows in the Instrument file at last. If the Statue exists on the Electronic Irish Statute Book website, then add a new column called Instrument Link.

The format and sample of the dataset are in Table 4.2. The statue is the main body of the Instrument. Instrument ID is the UUID generated by Instrument Column, while Statue ID is by Statue.

Column	Sample
Neutral Citation	2020_IEHC_344
Instrument	Criminal Justice (Terrorist Offences) Act 2005
Statue	Criminal Justice Act
Year	2005
Detail	Terrorist Offences
Instrument Link	<a href="https://www.irishstatutebook.ie/2005/en/si/0416.html?q=Criminal+Justice+Terrorist+Offences+Act&amp;years=2005">https://www.irishstatutebook.ie/2005/en/si/0416.html?q=Criminal+Justice+Terrorist+Offences+Act&amp;years=2005</a>
Instrument ID	48537
Statue ID	64073

Table 4.3: instrument\_link.csv Format and Sample

### Judge - judge\_link.csv (secondary dataset)

The extra information for Judge is from Wiki Data, where people can find the profiles of judges. Judges' names in Wikipedia must be formatted to correspond to the dataset for



Column	Sample
Neutral Citation	2021_IECA_282
Judge	Whelan J.
Court	Court of Appeal
Judge Link	<a href="https://en.wikipedia.org/wiki/M%C3%A1ire_Whelan">https://en.wikipedia.org/wiki/M%C3%A1ire_Whelan</a>
Judge ID	49920

Table 4.4: judge\_link.csv Format and Sample

linking. There are 1801 rows in the Judge file.

## 4.2.2 Dataset for Training

### Model Input - edge.txt

There are 19460 lines with three pairs of relationships in the dataset, as listed in table 4.5, where P represents the Case, A represents the Judge, T represents the Instrument, and C represents the Citation. The edges of a node will have different weights depending on the relationship of the edges. Correspondingly,

- p-a: Case judgedBy Judge; a-p: Judge judge Case (1 weight).
- p-t: Case citeStatue Statue; t-p: Statue citedBy Case (3 weights).
- p-c: Case citeCase Citation; c-p: Citation citedBy Case (2 weights).

Node ID	Node ID	Relationship	Weight
35081	49920	p-a	1
49920	35081	a-p	1
26957	29638	p-t	3
29638	26957	t-p	3
60268	7740	p-c	2
7740	60268	c-p	2

Table 4.5: Model Input Format and Sample

### Label - label.txt

The file label.txt contains only two columns of data "Case Node ID" (e.g. p35081) and "Topic" (e.g. 9). There are 1009 lines with ten topics as training labels. The LDA model classifies topics after training it with corpus. The amount of data under each category is not as much but differs significantly, i.e., unbalanced data, as Figure 4.1 showed.

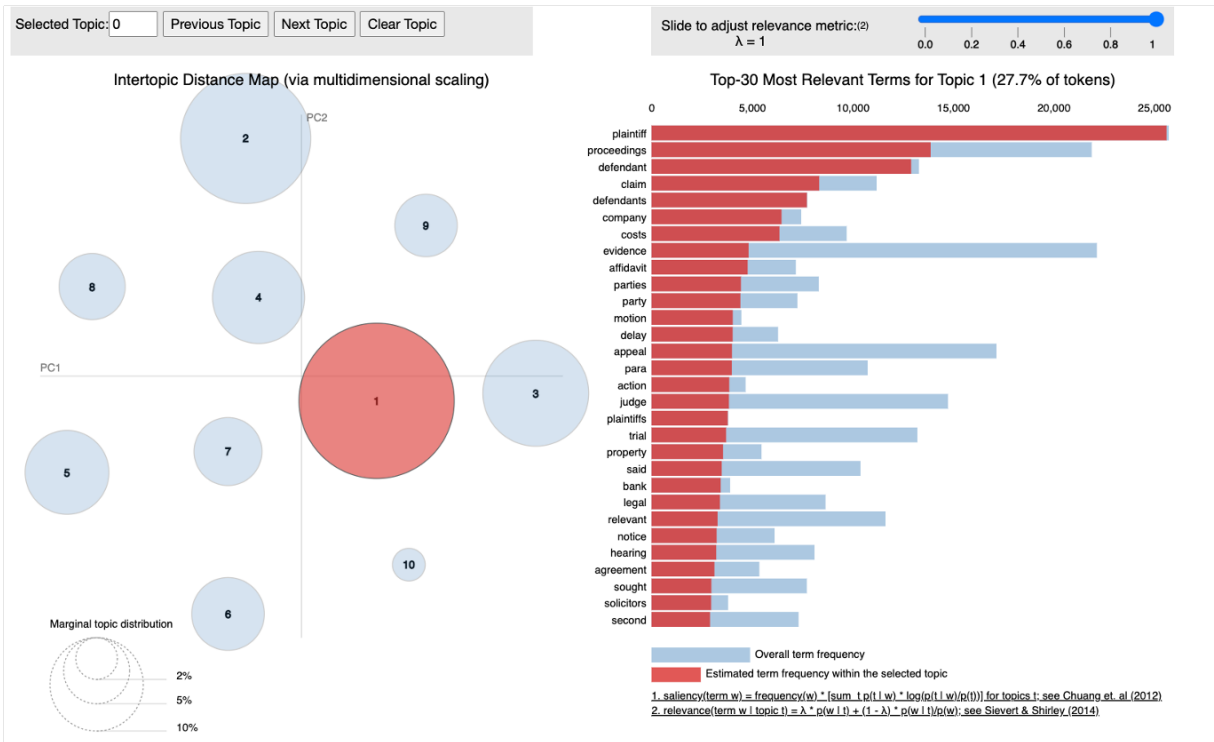


Figure 4.1: Topic Distribution

## 4.3 Ontology

### 4.3.1 Mapping Tools and Files

#### Difficulties Statement

- How to map the nested arrays in the dataset needs to be considered.

Three mapping tools are compared in six latitudes according to Section 2.2 Project 1, where ✓ means full support, while a (✓) means partial support.

	R1	R2	R3	R4	R5	R6
XSPARQL	✓	✓	✓	(✓)	(✓)	✓
SPARQL-Generate	✓	(✓)	✓	(✓)	✓	✓
RML-Mapper	✓	(✓)	✓	✓	✓	✓

Table 4.6: Mapping engines comparison

However, neither can solve the problem of nested data in columns, which means it is impossible to rely on the syntax of the mapping tool itself to implement the mapping. For this problem, the author finally adopted a roundabout solution by splitting and downgrading the nested arrays with python (Section 4.5.2) and mapping the different single-dimensional tables separately. Considering that R2RML-F

can be used for lightweight data processing, RML (R2RML) was finally chosen as the mapping tool.

- How to deal with Case entity citing itself.

Both ‘Case’ and ‘Citation’ belong to the Case entity. That is, there are a large number of Case entities that refer to their entity types, which is a recursive structure. For this problem, the author completes the entity mapping of the two separately and then links the two together via rr:IRI. Identical IRIs are considered to be the same node.

## Language and Tool

- Web Ontology Language (language) + Protégé (tool) for Ontology Construction
- R2RML(language) + GraphDB(database) for Knowledge Graph

## Mapping File - mapping.ttl

The snippet below shows how to map the citation relationship between two entities. For example, both citation entities and case entities are case classes. When mapping a citation entity, use rr:IRI to associate it with an already defined case.

```
<#CitationMapping> a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "citation" ] ;

  rr:subjectMap [
    rr:template "http://www.yuxin.com/law/Case/{CITATION}";
    rr:termType rr:IRI;
    rr:class ono:Case
  ];

  rr:predicateObjectMap [
    rr:predicate ono:citedBy;
    rr:objectMap [
      rr:template "http://www.yuxin.com/law/Case/{Neutral Citation}";
      rr:termType rr:IRI;
      rr:class ono:Case
    ]
  ]
.
```

The model will combine the node type and number to identify a node specifically. Moreover, the node information stored in our database must be consistent with the node information of the model. Therefore, the original data must be processed (add prefix) while mapping. The snippet below shows how to handle this problem with the help of R2RML-F.

```
@prefix rrf: <http://kdeg.scss.tcd.ie/ns/rrf#> .

<#addPrefixT>
rrf:functionName "addprefixT" ;
  rrf:functionBody """
    function addprefixT(citation_id) {
      return "t" + citation_id
    }
  """ ;
.
...
rr:predicateObjectMap [
  rr:predicate ono:hasCitationID;
  rr:objectMap [
    rrf:functionCall [
      rrf:function <#addPrefixT> ;
      rrf:parameterBindings (
        [ rr:column "Citation ID" ]
      ) ;
    ] ;
  ] ;
];
...
```

After running "java -jar Tools/r2rml.jar Ontology/config.properties", we will get mapping.output.ttl, which will be fed into GraphDB to generate a graph. The details of mapping.output.ttl can be found in the Github repository (<https://github.com/kongkongYuki/YuXin/blob/main/Output/.mapping.output.ttl>).

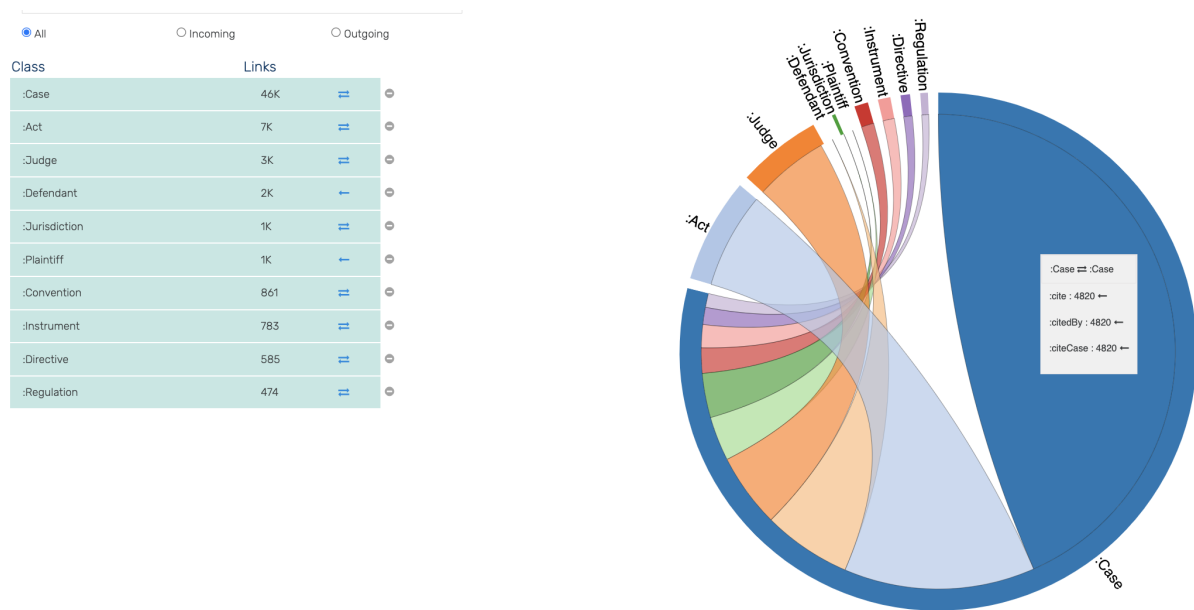


Figure 4.2: Class relationships in GraphDB

### 4.3.2 Database Visualization

### 4.3.3 Competency Questions Query

The SPARQL language can perform basic information retrieval and answer specific competency questions. The ontology can answer ten previously designed questions, reflecting that the knowledge graph effectively enriches the retrieval experience. Specific query statements will be shown in this section, while all queries need to reuse these URLs:

```
PREFIX ono: <http://www.yuxin.com/law/ontologies/2022/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

In the competency questions, (...) stands for entity, /.../ stands for variable placeholder, (...-...) stands for subclass relationship and "...” means the content of the variable.

1. (Case) after the /certain day/ "2022-01-01".

```
select ?case ?date where {
  ?case a ono:Case .
  ?case ono:deliverDate ?date .
  ?case ono:hasLink ?link.
  FILTER (?date > "2022-01-01"^^xsd:date)
}
```

2. What is (Case) from (Jurisdiction)"High Court"?

```
select ?case where {
  ?case a ono:Case .
  ?case ono:jurisdiction ?court .
  ?court rdfs:label ?court_name .
  FILTER (?court_name = "High Court")
}
```

3. What is (Case) judged by (Judge)"Hyland J."?

```
select ?case where {
  ?case a ono:Case .
  ?judge ono:judge ?case .
  ?judge rdfs:label ?judge_name .
  FILTER (?judge_name = "Hyland J.")
}
```

4. What are /approved/" Approved" (Case) on /Topic/"1"?

```
select ?case where {
  ?case a ono:Case .
  ?case ono:topic ?topic .
  ?case ono:status ?status.
  FILTER (?topic = "1" && ?status = "Approved")
}
```

5. Which (Judge) deals with the most cases in /Year/"2022"?

```
select ?name (COUNT(?case)AS ?count) where {
  ?case ono:judgedBy ?judge.
  ?judge rdfs:label ?name.
  ?case a ono:Case.
  ?case ono:year ?year.
  FILTER (?year = "2022")
}group by(?name)
order by desc (?count)
```

6. How many (Case) have contested?

```
select (COUNT(?case)AS ?contest_count) where {
```

```

?case a ono:Case .
?case ono:result ?result .
?case ono:defendant ?defendant.
FILTER (regex(?result, "The respondent contested" ))
}

```

7. What are failure rates on /Topic/"4"?

```

select ?rate (?y/?x AS ?rate) where {
  {
    select (COUNT(?case_a)AS ?x) where{
      ?case_a a ono:Case.
      ?case_a ono:topic ?topic_a.
      FILTER (?topic_a = "4")
    }
  }
  {
    select (COUNT(?case)AS ?y) where{
      ?case a ono:Case.
      ?case ono:topic ?topic.
      ?case ono:result ?result.
      FILTER (?topic = "4")
      FILTER (REGEX(?result, "dismissed") ||
              REGEX(?result, "refused"))
    }
  }
}

```

8. The ratio of (Precedent) : (Statue) in (Case).

```

select ?case (COUNT(?citation)AS ?citation_count)
           (COUNT(?statue)AS ?statue_count) where {
  ?citation ono:citedBy ?case.
  ?citation a ono:Case.
  ?statue ono:citedBy ?case.
  ?statue a ono:Case.
  {
    select ?case where{
      ?case a ono:Case.
      ?case rdfs:label ?case_name.
    }
  }
}

```

```
    }  
  }  
}group by(?case)
```

9. What is the most frequently used (Statue-Act)?

```
select ?act_name (COUNT(?case)AS ?c) where {  
  ?case ono:citeStatue ?act.  
  ?act a ono:Act.  
  ?act rdfs:label ?act_name.  
}  
group by (?act_name)  
order by desc (?c)
```

10. What (Legal Document) can support my case on /Topic/"5"?

```
select DISTINCT ?document_name where {  
  ?case a ono:Case.  
  ?case ono:topic ?topic.  
  ?case ono:citeStatue ?document.  
  ?document rdfs:label ?document_name.  
  FILTER (?topic = "5")  
}
```

The returned results show that the knowledge graph can answer the above questions correctly and accurately.

## 4.4 Knowledge Extraction

### Difficulties Statement

- How to gather required data.

There are no existing data sets, only publicly available official government websites. The primary question is how to get the required information from the available resources. For this problem, the author selects a series of crawler tools. The question then shifted to how to get the required data on a web page, i.e., positioning conditions. In Chrome's developer mode, developers can browse the web page source code and write corresponding positioning conditions based on HTML and CSS syntax.



- How to handle exceptional cases.

Individual cases may have exceptional circumstances, such as complex case titles, lack of Docx files, unusual metadata, etc. The code should consider not only the general case but also the particular case to enhance the code's robustness. It also includes data cleaning.

- How to find the best parameters for LDA model.

The LDA model has a crucial parameter that is `n_components` (number of topics). In order to find the best values for the parameters, a function in Scikit-learn (`LdaModel.log_perplexity`) can be applied to compare the results. Perplexity is a statistical measure of how well a probabilistic model can predict a sample. The lower the perplexity score, the better the generalization performance is indicated.

## Language and Tool

- Semi-Structured Data: Python (language) + BeautifulSoup(library) + Selenium(library)
- Unstructured Data: Python (language) + Blackstone(library)
- Label: Latent Dirichlet allocation(model) + Scikit-learn[3](library)

### 4.4.1 create repository crawl

The key is the combination of library features and positioning conditions. Below is the code to gather case links from every page of the websites.

```
table = driver.find_elements_by_xpath(
    "//table[@class='table alfresco-table']/tbody//td[2]")
for td in table:
    a = td.find_elements_by_tag_name('a')
    for item in a:
        text = item.get_attribute('href')
        if text.endswith('.pdf'):
            urls.append(text)
```

Below are the conditions to find the needed metadata from the HTML page.

```
downloaddoc = driver.find_element_by_partial_link_text('.docx')
title = soup.find("div", {"class": "alfresco-title"})
    .find("div", recursive=False).text.strip()
details = soup.find("div", {"class": "alfresco-properties"})
    .findAll("div", recursive=False)
```

```
composition = soup.find("div", {"class": "alfresco-comp"})
    .find("div", recursive=False)
```

## 4.4.2 create repository nlp

Below is the code to apply Blackstone to complete the Named Entity Recognition task and establish the structured data.

```
def get_results(text, citation):
    nlp = spacy.load("en_blackstone_proto")
    doc = nlp(text)
    sentence_segmenter = SentenceSegmenter(nlp.vocab, CITATION_PATTERNS)
    nlp.add_pipe(sentence_segmenter, before="parser")
    # Narrow the extraction to reduce the number of special cases that need
    # to be processed.
    results = {
        'CITATION': [],
        'INSTRUMENT': []
    }
    for ent in doc.ents:
        if ent.label_ in results.keys():
            if ent.text not in results[ent.label_]:
                tmp = str(ent.text).replace("[", "").replace("]",
                    "").replace(" ", "_")
                # Deal with the situation that the model will also extract
                # its own case number in the citation.
                if ent.label_ == 'CITATION' and tmp == citation:
                    continue
                results[ent.label_].append(ent.text.strip().replace("\n", "
                    "))
    return results
```

## 4.4.3 create repository topic

Below is the code to build an LDA model and use it to determine cases' topics.

```
# Build model and train.
vectorizer = CountVectorizer(min_df=5, max_df=0.9,
    stop_words='english', lowercase=True,
```

```

        token_pattern='[a-zA-Z\-\_][a-zA-Z\-\_]{2,}')
lda_model = LatentDirichletAllocation(n_components=NUM_TOPICS,
                                     max_iter=10, learning_method='online')
topic_scores = lda_model.transform(vectorizer.transform(document_list))
# Decide each document's topic.
topic_list = []
for doc in topic_scores:
    temp = doc.tolist()
    topic = temp.index(max(temp)) + 1
    topic_list.append(topic)

```

Below is the code to train the LDA model with the number of topics from 5 to 15 and observe the perplexity scores, i.e., model performance, for the different topics.

```

precom = []
for i in range(5,16):
    lda_model = models.LdaModel(corpus=corpus, num_topics=i,
                                id2word=dictionary)
    precom.append(lda_model.log_perplexity(corpus))
>> [-8.039661683398652, -8.05545537835374, -8.064944671305442,
    -8.076615081337685, -8.088781621325323, -8.099291910926855,
    -8.116034851310102, -8.13002214534761, -8.139280313174476,
    -8.161920375172803, -8.172038281568707]

```

Even though the scores continued to go lower, the author prefers 10 over 15 to avoid the risk of overfitting due to the small data set (the scores dropped significantly between 9 and 10).

## 4.5 Knowledge Integration

### Difficulties Statement

- How to create a mapping to a third-party database.  
By observation, most conversions of judges fit the "Name J." pattern. For cases that do not fit the pattern, handcrafted dictionaries are created for mapping.
- How to generate a globally unique code.  
There are many ways to generate a unique identifier, for example, generating random integers, using UUIDs, and creating a mapping based on actual values using CRC32 or MD5Sum. Considering data size, collision rate and character length, the author

truncates the characters generated by UUIDs with some minor risk by using the first 4 ( $10^4 = 10000$ ) hex characters, that is, DIGITS=4, converting those to an int and then putting each ID in the uniqueness loop check.

- How to extract and process strings.  
Regular expressions are the basic skills to deal with this problem and are suitable for format verification, information extraction, illegal name filtering and other scenarios.

## Language and Tool

- Third-party Database: Wiki Data + Electronic Irish Statute Book
- Data Alignment: Python (language) + Wikipedia(library) + BeautifulSoup(library)

### 4.5.1 create repository link

Below is the code to fetch the judges' links from Wikipedia.

```
def generate_df(page, level):
    judges = []
    judges_links = []
    soup = BeautifulSoup(page.html())
    # Positioning conditions
    table = soup.find("table", {"class": "wikitable"})
    judge_links = table.find_all('a', attrs={'title': True}, href=True)

    for judge in judge_links:
        judge_ = judge.text
        # Deal with general and special cases.
        if judge_ in list(mapdic.keys()):
            judges.append(mapdic[judge_])
        else:
            tmp = judge_.split(" ")
            judges.append(tmp[1] + " J.")
            judges_links.append('https://en.wikipedia.org/' + judge['href'])
    d = {'Court': [level] * len(judges), 'Judge': judges, 'Link':
        judges_links}
    return pd.DataFrame(d)

def wiki_judge_link():
    wikipedia.set_lang("en")
```

```

# Fetch Wiki page
high = wikipedia.page("List_of_judges_of_the_High_Court_(Ireland)")
supreme =
    wikipedia.page("List_of_judges_of_the_Supreme_Court_of_Ireland")
appeal =
    wikipedia.page("List_of_judges_of_the_Court_of_Appeal_(Ireland)")

d_high = generate_df(high, 'High Court')
d_supreme = generate_df(supreme, 'Supreme Court')
d_appeal = generate_df(appeal, 'Court of Appeal')

df = pd.concat([d_high, d_supreme, d_appeal])
return df

```

Below is the code to generate UUIDs.

```

original_ids = instrument_link['Instrument'].unique()
new_ids = {cid: int(uuid.uuid4()).hex[:DIGITS], base=16) for cid in
    original_ids}
instrument_link['Instrument ID'] = instrument_link['Instrument'].map(new_ids)

```

Below is the condition to extract detail and body information from Instrument.

```

year = re.match(r'.*([1-3][0-9]{3})', row['Instrument'])
detail = re.findall(r'\(.*?\)|\d*/.*', row['Instrument'])

```

## 4.5.2 create repository dic

Below is the code to expand nested arrays in the column into rows in another file.

```

def generate_citation(case_all):
    dic = {'Neutral Citation': [], 'Citation': []}
    for index, row in case_all.iterrows():
        tmp = row['CITATION'].strip('[]').split(',')
        for item in tmp:
            # Format conversion
            cit = item.strip()[1:-1].replace('[', '').replace(']',
                '').replace(' ', '_')
            # Extract only valid references that match the format
            matchObj = re.match(r'\d{4}_[a-zA-z]{4}_\d+', cit)
            if matchObj:

```

```

        dic['Neutral Citation'].append(row['Neutral Citation'])
        dic['Citation'].append(matchObj.group())
citation = pd.DataFrame(dic)
citation.drop_duplicates(inplace=True)
return citation

```

## 4.6 Recommendation algorithm

### Difficulties Statement

- How to implement graph embedding.

At first, the author referred to Section 2.2 Project 7 and adopted the DeepWalk model, but DeepWalk can only apply to homogeneous graphs, not to heterogeneous ones. Therefore, the author chooses the MetaGraph2Vec model, which is based on the DeepWalk but improved for heterogeneous graphs. The structure of the dataset is similar to that of the ACM dataset. Therefore, the authors also refer to the performance of each model on the ACM dataset.

ACM dataset	Micro-F1	Macro-F1	NMI
DHNE	0.7201	0.7007	0.3280
HAN	0.8401	0.8362	0.4241
HeGAN	0.8308	0.8276	0.4335
Metapath2vec(PAP)	0.7823	0.7725	0.2828
MetaGraph2vec	0.8085	0.8019	0.5095
PTE	0.7624	0.7543	0.3781

Table 4.7: HetG embedding model comparison on ACM dataset[4]

- How to use the features of the graph after dimension reduction.

The system's goal is to recommend, while the model's output is a feature matrix. A proper bridge must be found between the two, where similarity comes in. There are various similarity measures such as Persson correlation, Cosine similarity, Jaccard similarity, Spearman Rank Correlation, etc. Among them, Persson correlation and Cosine similarity are the most commonly used method, while the former is to find a linear correlation between two vectors, and the latter is to calculate the cosine angle between these vectors[1]. The author prefers to consider the relationship of all nodes at once, so cosine similarity is chosen as the metric.

- How to get labels for model training.

MetaGraph2Vec is a semi-supervised model, so the system needs labels to improve

the model performance and thus produce good recommendation results. Due to the lack of corresponding information in the official data source, the author can only use a clustering model (unsupervised model) to label the categories to which each node belongs as labels. According to Section 2.2 Project 4& 6& 9, the author argues that the LDA model is more relevant to the problem scenario than the SVM model and that the topic to which the case belongs can be considered the category to which the case belongs. It is worth noting that the annotation generated by clustering is feasible but cannot achieve the accuracy of manual labelling.

## Language and Tool

- Graph Embedding: MetaGraph2Vec(model)
- Knowledge Update: SPARQL(language) + SPARQL Wrapper[39](library)

### 4.6.1 OpenHINE

OpenHINE[4] is an open-source toolkit for Heterogeneous Information Network Embedding. It implements many popular models, including DHNE, HAN, HeGAN, HERec, HIN2vec, Metapath2vec, MetaGraph2vec, and RHINE. Here we choose MetaGraph2vec.

output/embedding/MetaGraoh2Vec/test\_node.txt: This is the ultimate file we need for similarity calculation.

```
p19656 0.076290965 -0.06172059 -0.1518152 0.07601022 0.13944595 -0.09014524
-0.11957887 0.14623547 -0.12914641 -0.07518722 -0.15120642 -0.14260426
0.14509615 0.05379134 -0.15115607 0.14265539 0.063791186 -0.08120041
0.061679743 -0.14567123 0.14420219 0.114089146 0.037378863 0.09723315
0.057445735 -0.14002267 0.1517213 -0.08478357 -0.09782345 -0.14996529
0.15285838 -0.1372418 -0.14112075 0.13776636 0.112719044 -0.13258146
0.05495679 0.15133242 0.055032577 -0.0703112 -0.07609697 0.096910365
-0.062505595 0.053169772 -0.12044188 0.024641298 -0.13994785 0.14944553
0.1415161 -0.14698099 0.084385395 0.111259066 0.09012766 0.13024577
0.15127277 0.13676456 0.14197007 -0.068906695 0.0694399 0.11988271
-0.0591615 0.1413959 -0.0754143 0.15167172
```

output/tmp/MetaGraoh2Vec/graph\_rw.txt: This is a temporary file containing the metapath generated by the random walk.

```
a1303 p14783 c43255 p56370 t35952 p56370 c15384 p56370 c15384 p37693 t63669
p37693 t36623 p19101 c47478 p37693 c13049 p19101 t8502 p19101 c13049
```

```
p37693 t63669 p37693 c47478 p37693 c15384 p37693 c47478 p19101 t8502
p37693 a59347 p13973 a59347 p61531 c31500 p36015 a25571 p10866 c63146
p64436 a46967 p7381 t64975 p7381 t11172 p7381 t13878 p7381 c51520 p7381
t11172 p7381 t11172 p7381 t9287 p7381 a46967 p57644 c31429 p52818 c12840
p52818 c12200 p16779 c12200 p44561 t31254 p2981 a37106
```

## 4.6.2 recommend

Below is the code to calculate the cosine similarity of the embedding matrix in order to rank the top 8 as recommendations.

```
def sortSparseMatrix(m, rev=True, only_indices=True):
    col_dict = dict()
    for i in range(m.shape[0]):
        d = m.getrow(i)
        s = zip(d.indices, d.data)
        sorted_s = sorted(s, key=lambda v: v[1], reverse=True)
        if only_indices:
            col_dict[i] = [element[0] for element in sorted_s]
        else:
            col_dict[i] = sorted_s
    return col_dict

def recommend(embedding, query_case):
    embedding.set_index(0, inplace=True)
    index_list = list(embedding.index)
    # Find the index of the input case
    index = index_list.index(query_case)
    A_sparse = sparse.csr_matrix(embedding)
    # Calculate similarity
    similarities_sparse = cosine_similarity(A_sparse, dense_output=False)
    # Rank similarity
    rec = sortSparseMatrix(similarities_sparse)[index]
    recommendation = []
    # Only return Case or Statue
    for i in rec:
        if 'p' in index_list[i] or 'c' in index_list[i] and i != index:
            recommendation.append(index_list[i])
    return recommendation[0:8]
```



### 4.6.3 process input case id

As mentioned, there are two scenarios, one in which the neutral citations entered by the user exist in the database and the other in which they do not. This file is written for the latter scenario, which requires obtaining information related to the case to generate the nodes, adding them to the existing knowledge graph and then re-training the model to get the new embedding matrix output. The process of the user input is similar to creating a repository, except that the order of magnitude is different.

### 4.6.4 execute graph query

In this file, various interfaces are written using SPARQL Wrapper to execute SPARQL queries in order to support the recommendation and update the database.

- `query_case_exist(self, neutral_citation):`
- `query_recommendation(self, recommend_list):`
- `query_case_info(self, neutral_citation):`
- `query_judge_info(self, neutral_citation):`
- `query_statue_info(self, neutral_citation):`
- `query_statue_info_by_id(self, statue_id):`
- `query_citation_info(self, neutral_citation):`
- `query_statue_id_by_name(self, name):`
- `query_citation_id_by_name(self, name):`
- `insert_new_case_info(self, dic):`
- `delete_new_case_info(self, dic):`

Below is the code to check if a case is in the database.

```
def query_case_exist(self, neutral_citation):
    query_str = """
        PREFIX ono: <http://www.yuxin.com/law/ontologies/2022/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

        SELECT ?id WHERE {
```

```

        ?case a ono:Case.
        ?case rdfs:label ?case_name.
        FILTER (?case_name = '"" + neutral_citation + ""')
        FILTER EXISTS { ?case ono:hasLink ?link . }
        ?case ono:hasCitationID ?id.
    }"""
self.sparql.setQuery(query_str)
try:
    ret = self.sparql.queryAndConvert()
    result = ret["results"]["bindings"]
    if len(result) == 0:
        return False
    else:
        for r in result:
            r_ = r['id']['value']
            if str(r_).startswith('p'):
                return r_
except Exception as e:
    print(e)

```

It is worth noting that additional authentication is required when performing insert/delete queries, and the POST method should be used instead of GET.

```

self.sparql.setHTTPAuth(DIGEST)
self.sparql.setCredentials("Name", "Password")
self.sparql.setMethod(POST)
self.sparql.setQuery(query_str)

```

## 4.7 Summary

This chapter lists the code files of the three modules according to the pipeline diagram and isolates the ontology and dataset for further explanation.

The Dataset section is divided into data for the ontology and data for the model, while the former is the basis for the latter and the latter is a re-transformation of the former.

The description of this section focuses on the structure and content of each data file.

The subsequent sections are presented in three perspectives: Difficulties Statement, Language and Tool, and Implementation Snippets. Difficulties Statement states the problems and solutions encountered in the implementation and reflection processes. Language and

Tool gives the conclusion of the choice. Implementation Snippets are the critical code and comments corresponding to the problem encountered.

The Ontology section mainly states the solution for nested arrays and recursive entities. An intuitive presentation of the database can be found in Section 4.3.2. Feasible queries for the Competency Questions raised in the previous chapter can be found in Section 4.3.3. The Knowledge Extraction section mainly discusses the solution to gathering data and label problems. The Knowledge Integration section mainly states the solution to processing data. The Recommendation algorithm section mainly states the selection of models and methods. See the code snippets in each section for details of the solutions.

# Chapter 5

## Evaluation

This Chapter consists of four parts: ontology, model, system, and overall evaluation. Among them, the Overall Evaluation was only designed but not executed to collect feedback due to time and personnel factors. Good results were achieved in the rest of the evaluation, with minor ontological defects in the Ontology Evaluation Section(5.1), 72% macro-f1 scores in the Model Evaluation Section(5.2)and 48.15% system validity ( $\gg$  probability of correct random recommendation: 5.29%) in the System Evaluation(5.3).

### 5.1 Ontology Evaluation

For the ontology evaluation, we use the tool OOPS(Ontology Pitfall Scanner)[26], an online ontology evaluation to assess the reliability and integrity of the ontology.

The result shows that ontology has four classes of minor and two classes of important problems. Since many entities have "Range" in the default string format, it would be easier to retrieve them by leaving them blank. Since the research does not have the plan to publish the ontology, the license problem can also be ignored. The report's details can be found in Appendix .1.

SPARQL queries for each Competency Question (Section 4.3.3) are also a form of ontology evaluation. Some of the results of the queries can be found in Appendix .2. However, readers can also download the relevant files from the Github repository (Ontology/ono.owl and Output/mapping.output.ttl), import them into GraphDB, and execute the corresponding queries to see the results.

Overall, according to its evaluation report and query results, the ontology is relatively accurate and reliable compared to its initial claims (Competency Questions).

## 5.2 Model Evaluation

### Model parameter vs. Model Hyperparameter

The model parameters are configuration variables within the model whose values can be estimated from the data. They are usually summarized from past training data.

The model hyperparameters are configurations external to the model, and their values cannot be estimated from the data. The model hyperparameters are set manually and are used in the process to help estimate the model parameters.

### Experiment purpose

The purpose of the experiment is to determine better hyperparameters. Based on this, the better parameters are trained by the model itself. Appropriate model hyperparameters and model parameters are required to produce good model performance. The model's performance is equivalent to the accuracy of the embedding matrix it gives. Thus, evaluating the model's performance equals evaluating the credibility of the model's embedding matrix (graph eigenvalues). When the model performs poorly, the embedding matrix-based recommendations are bound to be inaccurate. An essential step to getting valid recommendations is to improve the model's performance.

### Indicator explanation

There are differences in the calculation methods of micro-F1 and macro-F1. In the case of highly unbalanced data, micro f1 can only reflect a large class sample size. Since the amount of data under each category is not as much but differs significantly, that is unbalanced data, so macro-f1 is better than micro-f1 in our situation. F1-score = 1 is the best value (perfect precision and recall); the higher the NMI, the more information is shared between nodes; the lower the loss is, the better our model works.

### Cross-validation schema

Cross-validation schema is 80% training set, 20% testing set. The model will use the label as a correct criterion to gradually adjust the understanding of the graph. The model learns on 80% of the data, then uses the unseen 20% of the data to evaluate the learning effect, i.e., model performance.

## Optimization algorithm

The sparse Adam algorithm is used. The parameters of the model determine the way the model is represented. An optimization algorithm is an effective tool for estimating model parameters to achieve better metric values.

### 5.2.1 Hyperparameter Experiment Results

In the following five experiments, two control groups (Experiment 1 & 2 and Experiment 3 & 4 & 5) are set up to explore the choice of the three hyperparameters (num\_walks, walk\_length, epoch). Also, observe how much of the model’s optimal performance can be achieved under each experiment.

When num\_walks=50, it takes 25 epochs to achieve relatively acceptable performance with Macro F1 = 70%.

	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>
MacroF1	0.69	0.695	0.7	0.705	0.7	0.69	0.695	0.695	0.7
NMI	0.4982	0.4981	0.404	0.5003	0.5075	0.5008	0.5038	0.5059	0.5064
Loss	2.4703	2.4046	2.3432	2.2876	2.2334	2.1855	2.1384	2.097	2.0618

Table 5.1: Experiment 1: num\_walks=50, walk\_length=100, epoch =25

When num\_walks=70, it takes 15 epochs to achieve relatively acceptable performance with Macro F1 = 72%.

	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
Macro f1	0.61	0.655	0.675	0.685	0.7	0.705	0.715	0.725	0.72
NMI	0.4717	0.4851	0.4939	0.4981	0.5071	0.5106	0.5019	0.5045	0.5107
Loss	2.7341	2.7041	2.6594	2.5912	2.5032	2.4066	2.3134	2.2311	2.1584

Table 5.2: Experiment 2: num\_walks=70, walk\_length=100, epoch=15

When num\_walks=100, it takes 10 epochs to achieve relatively acceptable performance with Macro F1 = 72.5%, while walk\_length=70.

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>
Macro f1	0.26	0.365	0.43	0.565	0.645	0.675	0.685	0.705	0.725
NMI	0.0725	0.1092	0.3761	0.4397	0.5039	0.4903	0.4958	0.5073	0.5141
Loss	3.2092	2.8974	2.8	2.7523	2.7194	2.67896	2.6107	2.5082	2.399

Table 5.3: Experiment 3: num\_walks=100, walk\_length=70, epoch=10

When walk\_length=50, Macro F1 = 68%.

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	10
Macro f1	0.29	0.305	0.435	0.46	0.58	0.65	0.655	0.665	0.68
NMI	0.068	0.0696	0.2501	0.3962	0.4551	0.4547	0.4698	0.4711	0.4834
Loss	3.3871	3.0083	2.8687	2.8045	2.7658	2.7347	2.7009	2.649	2.5744

Table 5.4: Experiment 4: num\_walks=100, walk\_length=50, epoch=10

When walk\_length=100, Macro F1 = 69.5%.

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	10
Macro f1	0.295	0.415	0.645	0.645	0.665	0.675	0.695	0.705	0.695
NMI	0.0714	0.2839	0.48	0.48	0.4962	0.5131	0.5128	0.5182	0.5059
Loss	3.0624	2.8185	2.7488	2.7113	2.6614	2.5655	2.4306	2.2933	2.1718

Table 5.5: Experiment 5: num\_walks=100, walk\_length=100, epoch=10

## 5.2.2 Experiment Conclusion

It is concluded that a more significant number of walks leads to convergence with fewer epochs (Experiment 1 vs Experiment 2). The length of walks affects the final performance of the model, where 100 is too large, and 50 is too small, with the final score less than 70% (Experiment 3 vs Experiment 4 vs Experiment 5). The best macro f1 score of the model is 72%, which is acceptable performance.

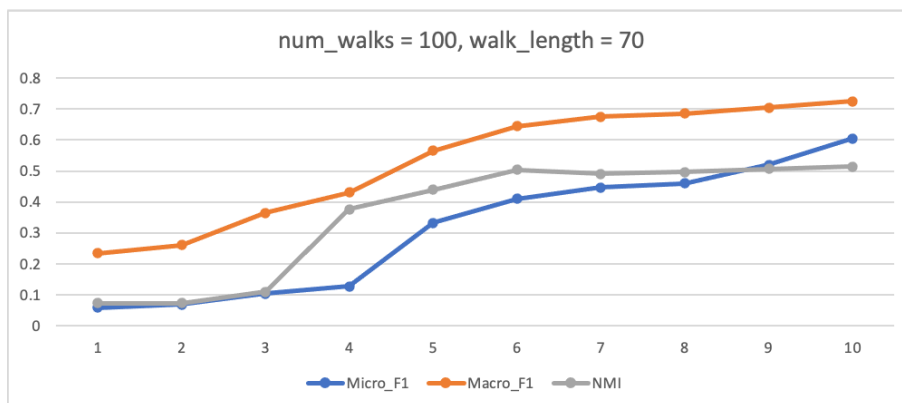


Figure 5.1: Experiment 3 Statistic Figure

The best performance works with the hyperparameter as follows:

num_walks	walk_length	neg_num	window_size	batch_size	alpha	epochs	lr_decay
100	70	5	5	64	0.001	10	0.001

## 5.3 System Evaluation

To evaluate the system's recommendation results more objectively, the author customises a metric using the specificity of certain cases and calculates the probabilistic results to compare with the probability of the recommendation given randomly by the system. This metric is only a rough prediction of the system's recommendation performance when there is a lack of manual evaluation, and the final evaluation will be based on the manual evaluation.

### Dual attributes case definition

Unlike in actual use, Statue is not included in the recommendation list when performing the system evaluation. A dual-attribute case is a case that is itself a case and is also cited by other cases, which can be defined as the equation below:

$$z \in Case, z = \begin{cases} 1, & z \in CitationList(any), \\ 0, & z \notin CitationList(any). \end{cases}$$

The following code filters all cases of dual attributes, getting Num.(Z) = 189.

```
case = pd.read_csv("case_all_with_topic_id.csv")
citation = pd.read_csv("citation.csv")
common = [x for x in list(citation['Neutral Citation']) if x in
          list(citation['Citation'])] # Common Elements
len(set(common)) # 189
```

### Recommendation valid criteria

The author assumes a strong connection between the case and its cited cases. Therefore, it is reasonable for the cited cases to appear in the recommendation list. For cases with dual attributes, the author believes the precedent it is cited should appear in its recommendation list. The recommendation is valid if the precedent citing the case is in the recommendation list and vice versa. For example, 2021\_IECA.322 cited by 2021\_IECA.344, therefore 2021\_IECA.344 should be in the recommendation list of 2021\_IECA.322. The valid recommendation criteria can be defined as the equation below:

$$x \in Z, x \in CitationList(y), x = \begin{cases} 1, & y \in RecommendList(x), \\ 0, & y \notin RecommendList(x). \end{cases}$$



## System Validity

$$Validity = \frac{\sum_{n=1}^{N-1} x}{\sum_{n=1}^N z}, N = 189 \quad (5.1)$$

Validity = Number of cases that meet recommendation valid criteria / Number of cases of dual attributes.

### 5.3.1 Topic speculation

Under each topic, the author selects the most frequently occurring cases to observe their recommendation results while inferring the topic's meaning based on the recommendation results. The following code counts the frequency of occurrence of each dual property case and sorts the dictionary. The most frequently occurring dual property cases under each topic are used as samples.

```
count = {}
for i in common:
    if i not in count.keys():
        count[i] = 1
    else:
        count[i] = count[i] + 1
dict(sorted(count.items(), key=lambda item: item[1], reverse=True))
>> {'2022_IEHC_393': 42,
     '2022_IEHC_167': 35,
     '2021_IECA_337': 34,
     .....}
```

Topic Number	Sample	Occurrence frequency	Topic speculation
1	2021_IECA_322	20	Criminal Trafficking
2	2022_IEHC_83	12	Planning and Development
3	2022_IEHC_393	42	Natural Planning
4	2022_IEHC_141	10	Illegal Immigration
5	2022_IEHC_167	35	Court Service
6	2020_IECA_159	7	Criminal Arrest
7	2021_IEHC_773	11	Landlord and Tenant
8	2020_IECA_184	25	Illegal Immigration
9	2021_IECA_248	20	Land and Conveyancing
10	2021_IECA_1	16	Citizens and Communities

Table 5.6: Topic speculation

After browsing through all the recommendations, topic speculations are made in Ta-

ble 5.6. Among them, Topic 2 and Topic 3, Topic 4 and Topic 8 are very similar. The recommendation results of the above samples are available in the GitHub repository (<https://github.com/kongkongYuki/YuXin/tree/main/Output/Recommendation>).

### 5.3.2 Analysis of the reasons for recommendation

The observation of samples' recommendation results leads to the following discoveries:

- All the recommendations are on the same topic.
- Recommended Statues frequently appear in the statute list cited in the recommended cases.
- The model does not use the citation directly.

Discovery 1 is intuitive because the system learns the information from the graph using the topic label as the correct answer. A qualified recommendation algorithm should be able to classify cases accurately. Discovery 1 verifies that the primary factor influencing the recommendation is the topic.

Discovery 2 is also intuitive because the recommendation algorithm gives Statue the heaviest weight(3). Discovery 2 verifies that statute is also an essential factor to consider when making recommendations.

There are situations where the citation is not included in the precedent recommendation list for Topic 2 and Topic 3 because the target case (2022.IEHC\_327) of Topic 2 (2022.IEHC\_83) is classified as Topic 3 while the target case (2022.IEHC\_395) of Topic 3 (2022.IEHC\_393) is classified as Topic 2. There are three target cases for Topic 8 (2020.IECA\_246 & 2022.IECA\_89 & 2021.IECA\_106). Only 2021.IECA\_106, accurately classified as Topic 8, is in the recommendation list. The recommendation of the rest of the topics proves to be valid. This phenomenon is the negative outcome of Discovery 1. It also shows that the model does not give recommendations directly based on citations, which is Discovery 3.

### 5.3.3 Recommendation efficiency

#### Hypergeometric Distribution

Define the situation in which the system's random recommendation is valid: Randomly select  $n$  cases from  $N$  different cases, where there are  $M$  target cases among  $N$  cases. In the absence of put-back sampling, the probability of the existence of  $k$  target cases among

the selected  $n$  cases is:

$$P_k = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n}, 0 \leq k \leq \min(n, M) \quad (5.2)$$

which is the hypergeometric distribution formula in probability theory.

Assume that there is one and only one target case in the whole group, thus,  $C_M^k = C_1^1$ .  $N = 189, n = 10, M = 1, k = 1$ , the probability of a correct random recommendation is

$$P_1 = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n} = \frac{C_1^1 C_{189-1}^{10-1}}{C_{189}^{10}} = 0.0529$$

according to the Formula 5.2.

### System Validity Calculation

The following code calculates the value of Equation 5.1. When the system returns 10 recommended cases,

$$Validity = \frac{91}{189} = 0.4815 \gg P_1 = 0.0529$$

which proves the system effectiveness.

```

for com_case in list(set(common)):
    com_id = case_id.loc[com_case]['Graph_ID']
    embedding = pd.read_table(EMBEDDING_PATH, names=list(range(0, 65, 1)),
        sep=' ')
    recommendation, evaluation = recommend(embedding, "p" + str(com_id))
    case = [int(item.replace('p', '')) for item in recommendation if
        item.startswith('p')]
    cit = list(tmp.loc[tmp['Citation ID'] == com_id]['Graph_ID'])
    common = set(case) & set(cit)
    if len(list(common)) > 0:
        count = count + 1
    else:
        invalid.append(com_case)
print(count)
print(invalid)

```

## 5.4 Overall Evaluation

### 5.4.1 Questionnaire

After discussing with a lawyer, the author designed a questionnaire to determine who is more critical in the Common Law system, keywords or citations (Question 2). And if citations are more important, how important are they ultimately (Question 3 & 5)? The lawyer mentioned those different regions might have different outcomes on the same case (Question 4). Question 5 is a ranking of all the indicators involved in the system.

- Do you think the system has practical value?
- For case search, which do you think is more important: keywords or citations?
- At least, to what level of granularity should citations be refined?
- Do you think the system needs to be filtered for regions?
- Please rank the following items in order of importance in your mind (Court, Status, Date, Result, Judge, Plaintiff/Defendant, Precedent, Statute, Keyword, Category).

The author interviewed three students pursuing the LLM degree at Trinity College Dublin. The interview transcripts are in the Appendix .3. The following is the author's summary based on the interviews:

1. The system has value, but mainly as a secondary support function. Keywords and citations are two stages in the retrieval process. After the relevant precedents are retrieved through keywords, the citation of precedents becomes the focus. But a complete system always starts with keywords and then citations.
2. It is necessary but not urgent to distinguish regions. The scope of use of a precedent is limited for case-law countries, especially regarding specific cases such as regional jurisdiction. But it will not be of much use for legal research.
3. The granularity of citation should effectively avoid the secondary search. It should summarize the core points of relevant precedents - "denial of this precedent", "continuation of this precedent", "similar to this precedent", and "previous relevant precedent of this precedent" or at least be precise to the article.

The next step is refining questions into multiple choice questions, distributing the online questionnaire to 30 Irish law students, and analysing the data results.

### 5.4.2 Feedback

The author designed a feedback test as a convincing, unbiased and comprehensive system evaluation. The author plans to invite ten non-legal professionals and ten legal professionals to rate their satisfaction with Competency Questions' query results, and the system's recommendation results meet their requirements from 0 to 10. Also, the authors will collect additions from respondents on the Competency Questions.

The next step will be to deploy a simple web portal, combined with GraphDB APIs, the system's APIs and a new rating API, on a public web server for respondents to access online. See Appendix .4 where the system interface has been implemented. The number of recommendations will be reduced to 5 to reduce the time required for respondents to evaluate the recommendations.

# Chapter 6

## Conclusions

### 6.1 Primary Contributions

This research set out to optimize the current case search experience of public platforms by applying the knowledge graph to the legal field, under the background that the current public platform is dominated by keyword searching and tag filtering. The overall system consisted of three parts: knowledge extraction, knowledge integration and recommendation algorithm, involving web crawlers, natural language processing, ontology construction and graph embedding. This research confirmed the value and feasibility of combining knowledge graphs and graph embedding for recommendations in the legal field through three kinds of evaluation at different levels. The achievements obtained in this research are promising if considered as a complementary feature to the existing platforms. If this research is considered a reform of the existing platforms, the current system presents far less functionality. Further studies need to be conducted to improve the current system.

### 6.2 Future Work

In order to change the orientation of this research from a complementary feature to a well-established system, the following additional aspects need to be improved:

1. The involvement of legal professionals  
Execute the plans mentioned in Section 5.4 but not completed to collect respondents' feedback on the current system and incorporate the comments into the scope of the subsequent system improvements.
2. Explainable reasons for recommendations  
Currently, the system can only give the recommendations results, and the author can

only speculate and analyze the basis of the recommendations based on the results. The system cannot provide a basis for recommendations other than similarity, which has no reference value to the user. The solution to this problem must be advanced in several ways, including expanding the dataset, refining citation granularity, and incorporating keywords.

### 3. Explicit label classification

According to Section 5.3.2, the label is critical to the recommendation result, and the current topic classification has errors. Explicit label classification is essential to improve model recommendations. While the clustering method often does not achieve the desired accuracy. Label relies on reliable data sources or extensive manual labelling; the former solution is more feasible than the latter.

### 4. Refinement of citation granularity

According to the answers to the questionnaire in Section 5.4.1, the granularity of the citation should be refined to avoid secondary search. This problem requires a classification of the different precedents cited in the same case, and the citation of the law needs to be precise to the article.

### 5. Combination with keyword

According to the answers to the questionnaire in Section 5.4.1, for a complete system, keywords and applications should be used in combination, first with keywords and then with citations. The recommendation algorithm will include two layers, one for information retrieval and filtering and one for graph embedding recommendation.

### 6. Dataset

- (a) Gather more data, not restricted to the word ‘Justice’. Instead, use all cases rather than a period with little pre-set.
- (b) Dig for information in the judgment and other metadata columns like the result column.

### 7. Model

Introduce more relationship pairs into the graph (now three pairs) and try other Heterogeneous Graph Embedding models.

## 6.3 Final Remarks

This research acted according to local conditions, presenting the idea of building knowledge graphs and using graph embeddings to recommend relevant cases to optimize the

local platforms' case search experience based on the characteristics of precedents and statutes cited in Irish legal judgement. Using natural language and crawlers makes it possible to build databases from original web pages and judgments, enabling the knowledge graph construction. This research also designed a relatively complete and feasible ontology and system architecture to achieve the research objectives and implemented a specific project using various tools. After that, this research evaluated the feasibility and effectiveness of the system step by step at three levels (ontology, model, system) and achieved promising results. This dissertation conducted a comparatively comprehensive study and achieved results/contributions within expectations.

The Irish judgement dataset from the research has the potential to be published on Ireland's Open Data Portal. There is an opportunity to publish the research's Knowledge Graph as Linked Open Data and link some of the Knowledge Graph entities into non-legal data sources to enrich the context of judgements. The potential possibilities/benefits of applying a knowledge graph approach to the legal system are to enhance and enrich the existing tag filtering search methods. Also, the recommendation system based on citation relationships can work as a feature to enhance the search experience after locating cases with keywords.



# Bibliography

- [1] Agarwal, A., Chauhan, M., et al. (2017). Similarity measures used in recommender systems: a study. *International Journal of Engineering Technology Science and Research*, 4(6):619–626.
- [2] Alghamdi, R. and Alfalqi, K. (2015). A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1).
- [3] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.
- [4] BUPTDMGroup (2022). Openhine. Website. last checked: 17.08.2022.
- [5] Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.
- [6] Chowdhary, K. (2020). Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649.
- [7] Costantino, M., Morgan, R. G., Collingham, R. J., and Carigliano, R. (1997). Natural language processing and information extraction: Qualitative analysis of financial news articles. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pages 116–122. IEEE.
- [8] Garijo, D. (2017). Widoco: a wizard for documenting ontologies. In *International Semantic Web Conference*, pages 94–102. Springer, Cham.
- [9] Garner, B. A. (2004). *Black’s Law Dictionary, (Black’s Law Dictionary (Standard Edition))*, volume 1805. Thomson West: Toronto, ON, Canada.

- [10] Guo, L., Cai, X., Qin, H., Guo, Y., Li, F., and Tian, G. (2019). Citation recommendation with a content-sensitive deepwalk based approach. In *2019 international conference on data mining workshops (ICDMW)*, pages 538–543. IEEE.
- [11] Gupta, S. and Varma, V. (2017). Scientific article recommendation by using distributed representations of text and graph. In *Proceedings of the 26th international conference on world wide web companion*, pages 1267–1268.
- [12] Hitzler, P. (2021). A review of the semantic web field. *Communications of the ACM*, 64(2):76–83.
- [13] Hoadley, D. (2020). Blackstone. Website. last checked: 17.08.2022.
- [14] Huang, Z., Zheng, Y., Cheng, R., Sun, Y., Mamouli, N., and Li, X. (2016). Meta structure: Computing relevance in large heterogeneous information networks. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining*, pages 1595–1604.
- [15] Information, C. (2022). Legal aid board. Website. last checked: 17.08.2022.
- [16] Initiative, I. L. I. (2007). About us. Website. last checked: 17.08.2022.
- [17] Junior, A. C., Orlandi, F., O’Sullivan, D., Dirschl, C., and Reul, Q. (2019). Using mapping languages for building legal knowledge graphs from xml files. *arXiv preprint arXiv:1911.07673*.
- [18] JuriGlobe (2016). Alphabetical index of the 192 united nations member states and corresponding legal systems. Website. last checked: 17.08.2022.
- [19] Krishnan, A. S. and Deepak, G. (2022). Intellilegalrec: An rdf based metadata driven semantically compliant recommendation system for socio-legal judicial documents. In *International Conference on Digital Technologies and Applications*, pages 407–416. Springer.
- [20] LexisNexis (2022). Lexisnexis: Solutions for professionals who shape the world. Website. last checked: 17.08.2022.
- [21] Li, L., Bi, Z., Ye, H., Deng, S., Chen, H., and Tou, H. (2021). Text-guided legal knowledge graph reasoning. In *China Conference on Knowledge Graph and Semantic Computing*, pages 27–39. Springer.
- [22] Lohmann, S., Negru, S., Haag, F., and Ertl, T. (2016). Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419.

- [23] Mirtaheri, S. M., Dinçtürk, M. E., Hooshmand, S., Bochmann, G. V., Jourdan, G.-V., and Onut, I. V. (2014). A brief history of web crawlers. *arXiv preprint arXiv:1405.0749*.
- [24] Mohit, B. (2014). Named entity recognition. In *Natural language processing of semitic languages*, pages 221–245. Springer.
- [25] ontotext (2022). Graphdb. Website. last checked: 17.08.2022.
- [26] Poveda-Villalón, M., Gómez-Pérez, A., and Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34.
- [27] Protege (2022). A free, open-source ontology editor and framework for building intelligent systems. Website. last checked: 17.08.2022.
- [28] Selenium (2022). The selenium browser automation project. Website. last checked: 17.08.2022.
- [29] Service, C. (1998a). About us. Website. last checked: 17.08.2022.
- [30] Service, C. (1998b). Let’s look at the law. Website. last checked: 17.08.2022.
- [31] Singh, S. (2018). Natural language processing for information extraction. *arXiv preprint arXiv:1807.02383*.
- [32] Thomas, M., Vacek, T., Shuai, X., Liao, W., Sanchez, G., Sethia, P., Teo, D., Madan, K., and Custis, T. (2020). Quick check: a legal research recommendation system. In *NLLP@ KDD*.
- [33] Trivedi, A., Trivedi, A., Varshney, S., Joshipura, V., Mehta, R., and Dhanani, J. (2020). Extracted summary based recommendation system for indian legal documents. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE.
- [34] vLex (2022). vlex: Your world of legal intelligenc. Website. last checked: 17.08.2022.
- [35] W3C (2012). R2rml: Rdb to rdf mapping language. Website. last checked: 17.08.2022.
- [36] W3C (2013a). Sparql. Website. last checked: 17.08.2022.
- [37] W3C (2013b). Web ontology language (owl). Website. last checked: 17.08.2022.

- [38] W3C (2014). Resource description framework (rdf). Website. last checked: 17.08.2022.
- [39] W3C (2022). Sparql endpoint interface to python. Website. last checked: 17.08.2022.
- [40] Wyner, A. and Hoekstra, R. (2012). A legal case owl ontology with an instantiation of popov v. hayashi. *Artificial Intelligence and Law*, 20(1):83–107.
- [41] Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018). Metagraph2vec: Complex semantic path augmented heterogeneous network embedding. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 196–208. Springer.
- [42] Zhang, N., Pu, Y.-F., Yang, S.-Q., Zhou, J.-L., and Gao, J.-K. (2017). An ontological chinese legal consultation system. *IEEE Access*, 5:18250–18261.

# Appendix

## .1 OOPS report

### Evaluation results

It is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, each pitfall has an importance level attached indicating how important it is. We have identified three levels:

- **Critical** 🚫 : It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** ⚠️ : Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** 🟡 : It is not really a problem, but by correcting it we will make the ontology nicer.

[Expand All] | [Collapse All]

<b>Results for P04: Creating unconnected ontology elements.</b>	<b>1 case   Minor</b> 🟡
Ontology elements (classes, object properties and datatype properties) are created isolated, with no relation to the rest of the ontology.	
• This pitfall appears in the following elements: > <a href="http://www.yuxin.com/law/ontologies/2022/Person">http://www.yuxin.com/law/ontologies/2022/Person</a>	
<b>Results for P08: Missing annotations.</b>	<b>49 cases   Minor</b> 🟡
<b>Results for P11: Missing domain or range in properties.</b>	<b>21 cases   Important</b> ⚠️
<b>Results for P13: Inverse relationships not explicitly declared.</b>	<b>27 cases   Minor</b> 🟡
<b>Results for P22: Using different naming conventions in the ontology.</b>	<b>ontology*   Minor</b> 🟡
The ontology elements are not named following the same convention (for example CamelCase or use of delimiters as "-" or "_") . Some notions about naming conventions are provided in [2].	
*This pitfall applies to the ontology in general instead of specific elements.	
<b>Results for P41: No license declared.</b>	<b>ontology*   Important</b> ⚠️
The ontology metadata omits information about the license that applies to the ontology.	
*This pitfall applies to the ontology in general instead of specific elements.	

According to the highest importance level of pitfall found in your ontology the conformace badge suggested is "Important pitfalls" (see below). You can use the following HTML code to insert the badge within your ontology documentation:

Figure 1: Oops! Ontology Report

## .2 Query Result

- Question 4: How many [Case] have contested?

```
"results" : {
  "bindings" : [
    {
      "contest_count" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#integer",
        "type" : "literal",
        "value" : "38"
      }
    }
  ]
}
```

- Question 6: What are failure rate on /Topic/(4)?

```
"results" : {
  "bindings" : [
    {
      "rate" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
        "type" : "literal",
        "value" : "0.156862745098039215686275"
      }
    }
  ]
}
```

- Question 8: What is the most frequently used [Statue-Act]?

```
"results" : {
  "bindings" : [
    {
      "act_name" : {
        "type" : "literal",
        "value" : "European Arrest Warrant Act"
      },
      "c" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#integer",
```

```

        "type" : "literal",
        "value" : "105"
    }
},
{
    "act_name" : {
        "type" : "literal",
        "value" : "Legal Services Regulation Act 2015"
    },
    "c" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#integer",
        "type" : "literal",
        "value" : "85"
    }
},
.....

```

## .3 Questionnaire

### .3.1 Respondent 1:

- Do you think the system has practical value?  
Have practical value. Because whether we do legal research or practical research, we have to do case studies. A case study would require a case search based on keywords, cause of action, or different laws. The legal case search tools that exist in the market do not have much functionality to recommend legal-related cases.
- For case search, which do you think is more important: keywords or citations?  
Keywords. Because in all cases there are corresponding keywords, for example, civil, criminal, and administrative are the keywords on legal categories; borrowing, lending, and contract assignment are the keywords on the parties' behaviour. The citation function is also important, but I think it should be used as a secondary tool.
- At least, to what level of granularity should citations be refined?  
The respondent misunderstood the question. The Respondent discussed the granularity of the keywords.
- Do you think the system needs to be filtered for regions?

This feature should be available but not necessary. Because it will not be of much use to the region for legal research. But this is not absolute; if it is a matter of regional jurisdiction or a dissenting appeal, the region will be a more important factor.

- Please rank the following items in order of importance in your mind (Court, Status, Date, Result, Judge, Plaintiff/Defendant, Precedent, Statue, Keyword, Category).  
Statue >Precedent >Status >Keyword >Category >Court >Plaintiff/Defendant >Date >Judge >Results

### **.3.2 Respondent 2:**

- Do you think the system has practical value?

I'm not very clear on how this system works, so can only answer based on my general understanding. The retrieval of precedents is meaningful for the common law system, and the key to retrieval is how to find the precedents that are most relevant to the facts to be retrieved. If the system can satisfy the input of relevant keywords and the final output retrieval results are within a reasonable range (the more accurate the range of similar cases, the more time can be saved for the user), then the system is meaningful.

- For case search, which do you think is more important: keywords or citations?

In my opinion, keywords and citations are two stages in the retrieval process. When you understand the relevant legal facts and need to find relevant precedents, the initial search can only be done through keywords. Even if it can be designed to search through the description of the facts of the case, unless the system is intelligent enough, the search results will still be presented through the search keywords in the end. After the relevant precedents are retrieved through keywords, the citation of precedents becomes the focus. From a vertical perspective, a precedent is often based on overturning or acknowledging the previous precedent. It would be very meaningful if the precedent could be cited in the precedent. From a horizontal perspective, it would also be helpful for users if the differences between similar but different precedents could be presented. These contents are mainly realized through citation in the main case.

- At least, to what level of granularity should citations be refined?

In addition to showing the basic case number, you can consider using several keywords to summarize the core points of relevant precedents - "denial of this prece-



dent”, ”continuation of this precedent”, ”similar to this precedent”, and ”previous relevant precedent of this precedent”

- Do you think the system needs to be filtered for regions?

For case-law countries, the scope of use of a precedent is limited. Taking the United States as an example, it is divided into state and federal courts. The state courts have the precedents and legislation of the state courts, and the federal courts have the precedents and legislation of the federal courts. Therefore, it is necessary to distinguish regions.

- Please rank the following items in order of importance in your mind (Court, Status, Date, Result, Judge, Plaintiff/Defendant, Precedent, Statue, Keyword, Category). As mentioned earlier, keywords are the core of searching a case, and regional classification by courts is also a way. As for other items, they can be juxtaposed as secondary search classification options.

### **.3.3 Respondent 3:**

- Do you think the system has practical value?

This question is too broad to answer.

- For case search, which do you think is more important: keywords or citations?

Keywords. Citations are only valuable once you have found the case and can find what you want faster than keywords.

- At least, to what level of granularity should citations be refined?

Generally, when we apply precedent, we need to see how precedent is judged; the ideal system is to reduce this secondary search. At least, it should be specific to the article.

- Do you think the system needs to be filtered for regions?

The region may involve the discretion of the judge. Guilty or not guilty will not make much difference; sentencing may.

- Please rank the following items in order of importance in your mind (Court, Status, Date, Result, Judge, Plaintiff/Defendant, Precedent, Statue, Keyword, Category). Not during the search, Status >Results >the remains. During the search, Keywords >Status >Results >Date >the remains.

The Statute becomes essential when searching for a particular case based on the Statute, so it is still important to sort it according to the purpose when searching.

Regarding keywords and citations, it must be keywords first and then citations. It is not that citations are unimportant, but it is the point at which the question proceeds that starts to become important.

## .4 System Interface

The screenshot displays a REST client interface with the following components:

- URL Bar:** `http://127.0.0.1:5000/api/2022_IEHC_141`
- Method:** `GET`
- Send Button:** A blue button labeled "Send".
- Params Tab:** A table titled "Query Params" with the following structure:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		
- Response Tab:** Shows a `200 OK` response with a time of `23.72 s` and a size of `6.4 KB`. The response is displayed in "Pretty" JSON format:

```
1  {
2    "p15426": {
3      "Citation": [
4        {
5          "link": null,
6          "name": "2010_IEESC_3"
7        },
8        {
9          "link": null,
10         "name": "2019_IEHC_273"
11       },
12       {
13         "link": null,
14         "name": "2012_IEESC_26"
15       },
16       {
17         "link": null,
```