FYP Final Report

CReKG: A Knowledge Graph Based Contract Review System

by

CHEN Zixin, SHUM Kashun and Tu Bingying

SYQ2

Advised by

Prof. Song Yangqiu

Submitted in partial fulfillment of the requirements for COMP 4981

in the

Department of Computer Science

The Hong Kong University of Science and Technology

2021-2022

Date of submission: April 20, 2022

Acknowledgments

We would like to express our sincere gratitude to our project supervisor Professor SONG Yangqiu for giving us invaluable suggestions and guidance throughout the project. He has met with us several times and deeply inspired us with his vision and sincerity. We are extremely grateful for his patience and willingness to help.

We are very much thankful to Professor Sung Hun KIM and Professor Dimitris for being the second reader of our project and providing insightful and constructive comments on our reports.

We would like to express our thanks to Mr.CHAN Tsz Ho, Gio for his valuable suggestions on the construction of the knowledge graph and the overall pipeline. Also we are grateful to the volunteers participated in the user study. Their time and feedback have been very much appreciated.

Abstract

Contract review has always been a complex and significant task. Even for professional lawyers, reviewing the specifications of a contract requires extensive data research and retrieval. For the majority of the parties in the contract, reviewing the contract is a very difficult task. To address this pain point, help attorneys and general users review their contracts, we build our powerful CReKG system.

The core technology behind CReKG is knowledge graph, a knowledge base that use a graph-structured data model or topology to integrate valuable information. We are the first to build a legal contract knowledge graph, and implement specific visualization modes and interaction functions for different needs. Meanwhile, we design and implement an automated contract review system. Users are guided through six key terms to review their contracts, search for legal information, and end up with a review report.

This report details the research and development process, technical points, testing and evaluation of the CReKG system.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

With the gradual increase in the legal awareness of the public, the workload of law firms and lawyers has reached an unprecedented level. According to the 2018 Legal Trends Report[5], lawyers in the United States work an average of 49.6 hours per week. Furthermore, most lawyers (75%) report frequently working outside of regular business hours, which negatively affects their personal lives. However, many lawyers spend roughly 50% of their time reviewing contracts, which is considered especially tedious than other legal tasks[7]. Therefore, the research and development of automated contract review have substantial application value to society.

In recent years, law firms embraced Artificial Intelligence (AI) based technology to help automate the contract review process. Nevertheless, current automated contract review systems mainly adopt simple machine learning models, ignoring the legal profession knowledge[9, 4, 8],. The introduction of Legal Knowledge Graph can enhance legal professionalism in contract review because Knowledge Graph is a connected graph composed of nodes and edges which consist of a large amount of legal knowledge. knowledge graphs, knowledge bases that use a graph-structured data model or topology to integrate data, can benefit people from all walks of life to improve searching accuracy and working efficiency. Figure 1 is an example of a knowledge graph with entities presented as nodes and relations as edges.

Therefore, this project researches automated contract review based on the legal Knowledge Graph and aims to propose a legal Knowledge Graph-based automated contract review system that focuses on solving the lack of legal knowledge problem. In conclude, our group decided to:

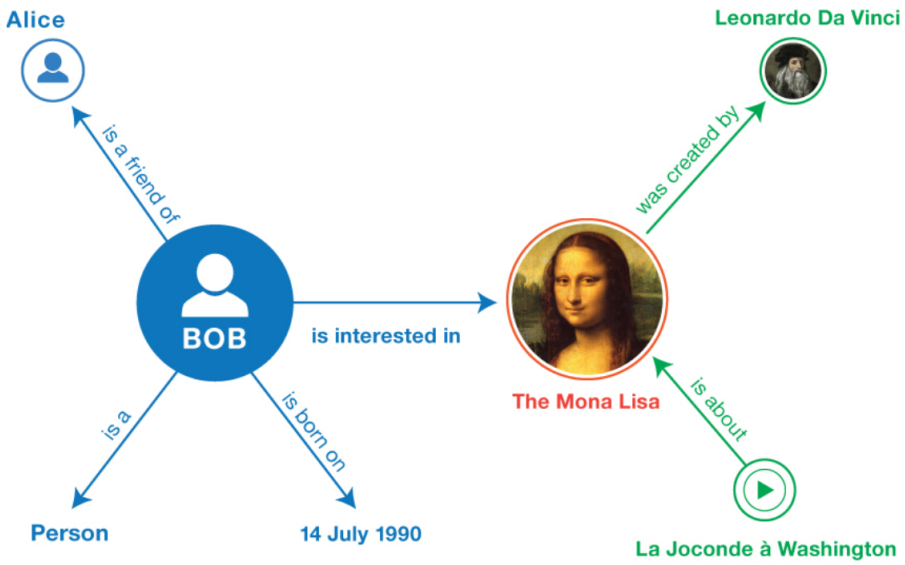

Figure 1: Example of a knowledge graph.

1. Build a knowledge graph from high-quality contract data to acquire important knowledge in contract review.

2. Establish an automated contract review system to help lawyers and contract parties review contracts.

In this project, we used the Contract Understanding Atticus Dataset (CUAD) [7], an expert-annotated dataset for legal contract review. The dataset identifies important clauses that legal professionals look for when reviewing contracts.

Our CReKG project includes a user interface for users to upload contracts, search keywords in the knowledge graph, visualize the results and download an analysis report. Meanwhile, with the interactive knowledge graph, users can also explore the data and other information. With our CReKG system, attorneys and other users can efficiently retrieve accurate legal information, and expedite the review of their contracts. Our CReKG system contributes to this subject's literature by applying Knowledge Graph to contract review for the first time, building a full-stack system for users to review contract and visualize Knowledge Graph.

1.2 Objectives

The goals of our project was to construct a knowledge graph consists of valuable entities, events, and relations from contracts. And then establish a system to help lawyers and contract parties review contracts. To achieve the goals, we worked on the following objectives:

1. To construct a knowledge graph from high quality legal contract datset.
2. To refine the knowledge graph with extra professional legal documents.
3. To do solid validation on the established knowledge graph.
4. To realize beautiful and useful visualizations for the knowledge graph.
5. To build up a database for storing the knowledge graph and supporting information query.
6. To establish a contract review pipeline with the help of knowledge graph.
7. To develop a full stack system to integrate all functionalities. The UI should allow users to upload their contract and review it with the help of our interactive knowledge graph.
8. To design an analysis report to aid contract review.

1.3 Literature Survey

Recent advances in knowledge-graph-based research have focused on leveraging domain-specific knowledge graphs. In this section, we discuss two previous works on domain-specific knowledge graphs as our reference.

1.3.1 Medical Knowledge Graphs

Previous work has focused on medical knowledge graphs. Some researchers built a knowledge graph from a medical text corpus of 29,500 papers and provided an interactive web application for users to access it [3]. By using this large-scale medical knowledge graph, medical profes-

sionals are able to understand COVID-19 better. The setting is quite similar to that of our project so we are using their work as our backbone for our legal knowledge graph.

1.3.2 Similar Cases Recommendation

One previous work focused on a similar cases recommendation function by using a large-scale legal knowledge graph [6]. By utilizing a knowledge graph, they show improvement upon similar cases recommendation systems. Different from their task, our goal was to first build a large-scale knowledge graph from legal documents and then utilize the information to provide some related laws when given input keywords.

At the same time, contract review task is also essential to our project. To date, several studies have investigated the application of Artificial Intelligence on Automated Contract Review. Within this broad area of inquiry, there have been several streams of research. One stream focuses on extracting information in contracts by searching manual rule-based keywords[2, 1]. Other streams focus on utilizing machine learning models and deep learning models to retrieve information[9, 4, 8]. Given the expensive manual-labelling effort and unknown accuracy on unfamiliar agreements problems, more research attention has been paid to machine learning based methods nowadays. This review of related work below focuses on machine learning approaches as our project is an extension of this stream.

Most previous machine learning based approaches used supervised learning techniques which means a large number of contract examples are fed into the training system. The system takes this data and learns languages that are relevant to a given provision concept. Compared to rule-based keywords search models, it brings more accurate results, even on unfamiliar agreements.

However, we argue that current machine learning models neglect legal profession knowledge, resulting in a long period of Question and Answering (Q&A) to obtain such knowledge from the users. For example, “does the contract mention confidentiality obligations” is initially asked, if that answer were yes, the engine then follows up with “does the contract confer confidentiality obligations upon us?” etc. [4]. This Q&A framework seeks to provide specific legal knowledge in a contract to the model. However, it is still considered time-consuming and not fully automated. Therefore, this project aims to introduce legal profession knowledge into the model by constructing a large-scale legal Knowledge Graph. By building the Legal Knowledge Graph, our project can leverage the legal knowledge to provide a fully automated contract review for users.

None of the related work has constructed a fully automated and high-accuracy contract review system. Current studies either suffer from unstable accuracy problems or lack legal profession knowledge, resulting in a Q&A section. As such, our project extends the previous research by incorporating a legal Knowledge Graph to fill this research gap.

2 Methodology

In accordance with our objectives, we built up a Knowledge Graph for legal contracts, designed a contract review process, and developed a web application to integrate knowledge graph visualization and contract review together. The details about our overall design, knowledge graph construction, contract review process, project testing and evaluation are shown in this Methodology session.

2.1 Design

2.1.1 Overall System Design

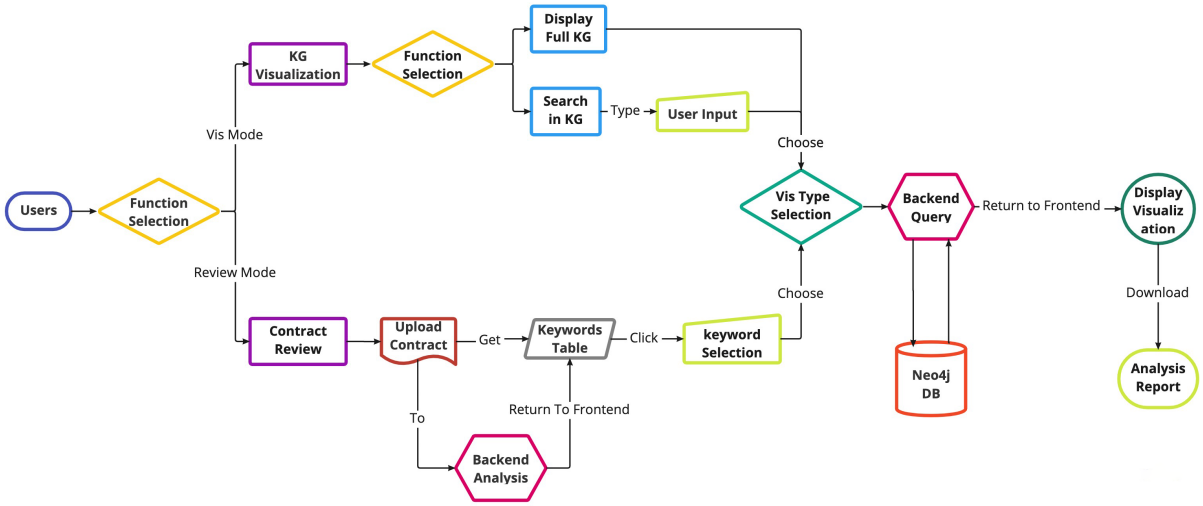

Figure 2: Overall Design Flow for the Users.

Figure 2 shows the overall design flow of our CReKG system from users' perspective. As shown in the figure, our system mainly supports two core functions: **the knowledge graph visualization and contract review.**

To use our CReKG system, users first decide the main functionality. The knowledge graph visualization mode supports node and relationships search functionality as well as visualization. Users can choose to display the whole knowledge graph, or input the keywords they are interested in and choose one of our four visualization modes(Circular,Force-directed, Hierarchical and Dynamic Physical) to display the sub-graphs. Fuzzy matching will query the database and return all sub-graphs that contain this keyword. All our graph visualizations support a variety of interactions. Users can further explore the graphs and gain high-quality information. The details about knowledge graph visualizations and interactions are shown in **2.3.8 Knowledge Graph Visualization** section.

The contract review mode allow users to upload their contract. Our system will do contract analysis from 6 key clauses. The analysis result will be returned as a keyword table. Keywords will be highlighted in user's contract and each keyword will trigger a query in our knowledge graph. The query results provide extra information to aid the users' review. Finally, a comprehensive analysis report will be generated and users can download the PDF report directly from our system. The details about this contract review process and the screenshot of PDF report are shown in **2.3.9 Pipeline for Contract Review** section.

2.2 User Interface Design

A website is designed as the user interface for our CReKG system and two separate website tabs are used for visualization and contract review. User can click the *Visualization* and *Contract Review* button on the navigation bar on the top to switch between the two tabs.

2.2.1 UI Design - Knowledge Graph Visualization

To achieve the Knowledge Graph Visualization functionality illustrated in Section 2.1.1, We designed the user interface in figure 3 below.

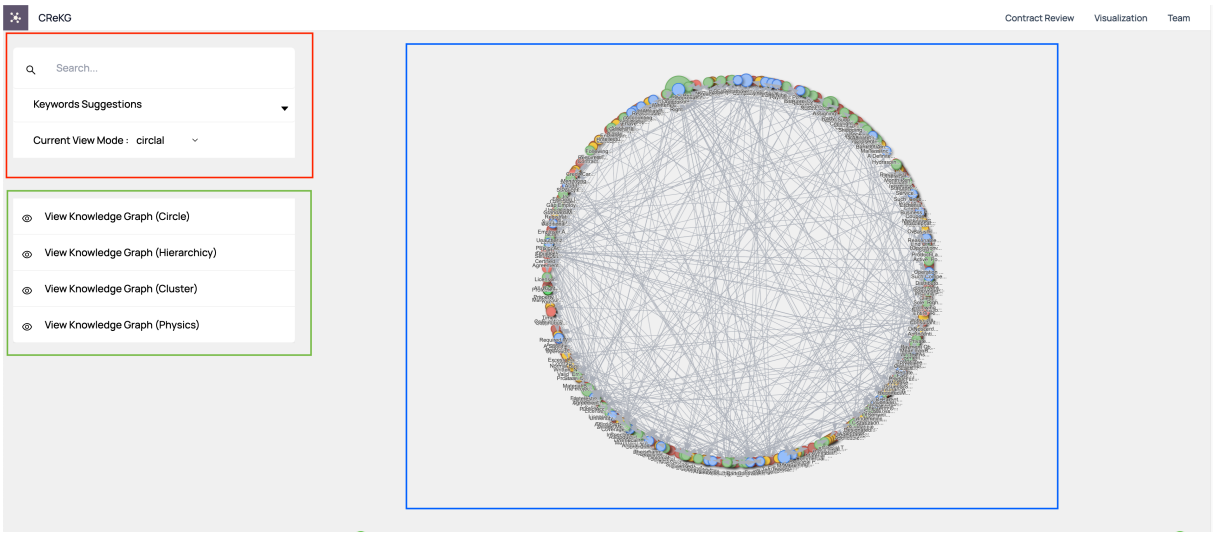

Figure 3: Knowledge Graph Visualization Main User Interface

The whole page can be divided into three blocks. The blocks bounded by the red line is the search block. User can input their interested keywords in the search bar (i.e. entities, relations), then select one of the four visualization mode by the drop-down list to the right of *Current View Mode*, and finally click the search button. All sub-graphs that contain this keywords will display in the block bounded by the blue line. To make it easy for user to search keywords, we also provide a drop-down keywords suggestion table which includes 9 common Legal-related keywords in our Knowledge Graph. A demo of search UI is shown in figure 4 below.

On top of the search function, user can explore the full knowledge in different mode as well. As shown in the block bounded by the green line, there are four buttons indicating four different modes to view the full Knowledge Graph. By clicking the button, the Knowledge Graph with the corresponding mode will display in the block bounded by the blue line.

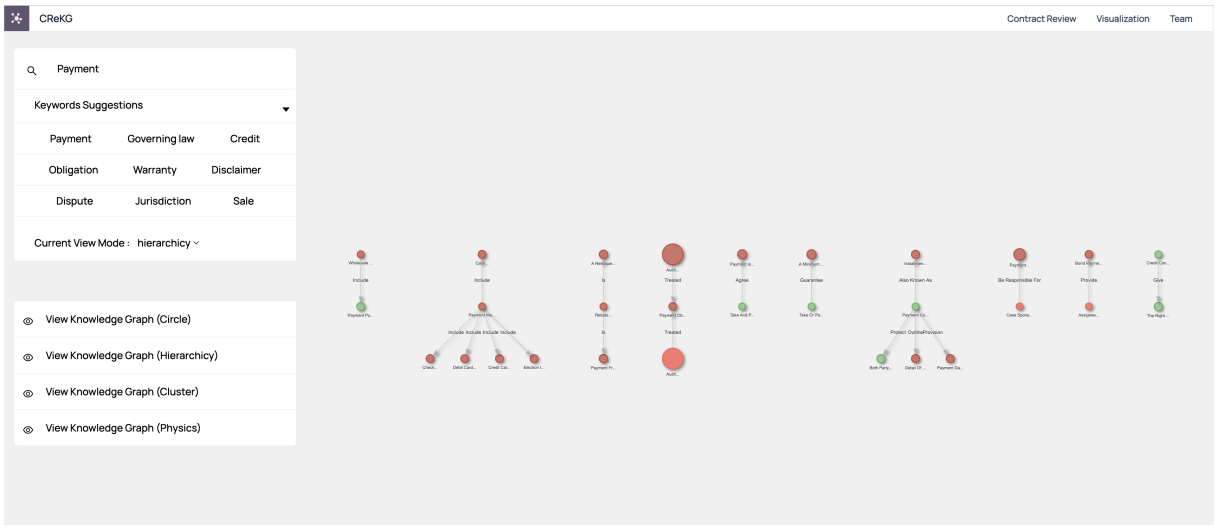

Figure 4: Knowledge Graph Visualization Search User Interface

2.2.2 UI Design - Contract Review

To achieve the Contract Review functionality illustrated in Section 2.1.1, we designed the user interface in figure 5 below.

In the Contract Review page, user can take several steps to let the system automatically review

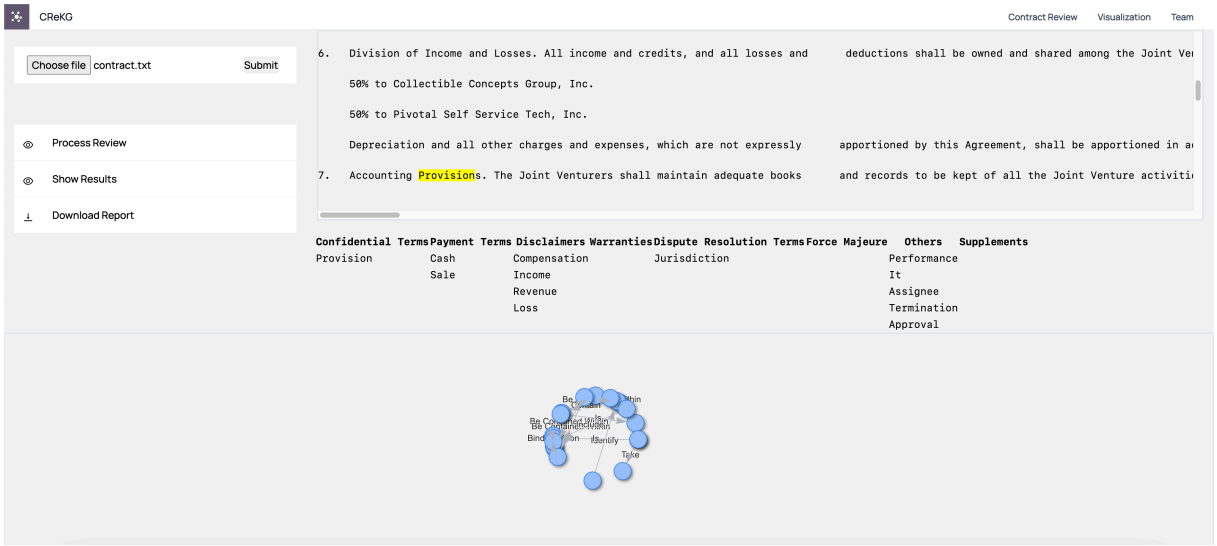

Figure 5: Knowledge Graph Visualization Contract Review User Interface

the contracts from input the contract to download a report. It can be generally divided into four blocks. The left side of the user interface contains the sequential buttons to complete a review process. Firstly, user can click the *choose file* button to select the contract he would like review and then click the *submit* button to display the contract at the top right corner. Then by

clicking the *Process Review* button, the contract will be sent to the back end for processing which includes keywords extraction, keywords classification etc. After that, user can click the *Show Results* button to get the processed result right below the contract display area. Each keyword is clickable which provide original text highlighting in the contract display area and Knowledge Graph contain that keywords at the bottom. In addition, these keywords are classified as 7 categories which belongs to different part of the contract in order to provide user a more clear structure of the contract. Finally, user can click the *Download Report* button to download a pdf version of report which contains the review result and suggestions from our system and it completes the review process.

2.3 Implementation

In this section, we introduce the detailed process of the constructing the knowledge graph. Figure 6 demonstrates the pipeline of KG construction. The following subsections introduce the details in this pipeline.

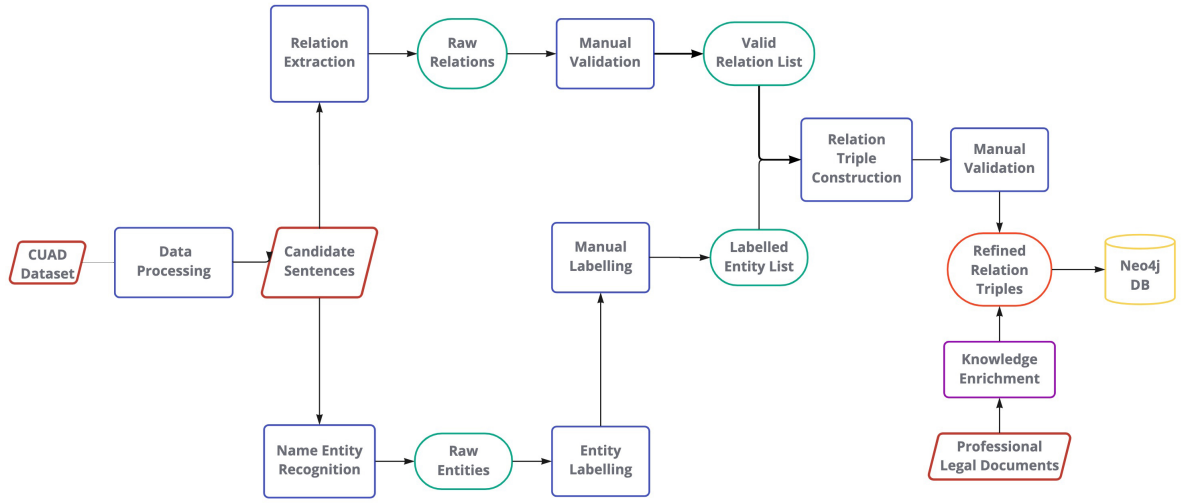

Figure 6: The pipeline of KG construction

2.3.1 Data Set Selection

The data set we used for our Knowledge Graph Construction is the CUAD(Contract Understanding Atticus Dataset) data set. CUAD is a high-quality data set created by legal experts. It extracted important parts in contracts for human review. The data set consists more than 600 contracts in different domains and was revised in Nov. 2021. Currently, it is the only available data set with high quality contract corpus

2.3.2 Data Processing

The data processing includes data cleaning, sentence segmentation and candidate sentences selection.

Although the CUAD dataset is a high quality data source, some garbled files, garbled sentences and duplicated documents still exists. We wrote a Python script to filtered out the garbled files and the duplicated documents by obtaining each contract's title. Each unique and meaningful title corresponds to one valid contract and we added it into our corpus. After that, we used Python again to further cleaning our corpus by deleting the page notes, handling the blanks and removing the garbled sentences.

```
pattern_1 = re.compile("\(Page [0-9]*\)") # regular expression to handle the page note in the data
pattern_2 = re.compile("\. ") # handle the blank behind .
pattern_3 = re.compile("\s\s\s*") # handle the multiple blanks
```

Figure 7: Screenshot of part of the data cleaning process.

The first step to handle our corpus is sentence segmentation. Sentence segmentation split the text document or article into sentences. In our project, we utilized **spaCy**, an open-source software library for advanced natural language processing, to do this task. Compared to other famous NLP package like NLTK, spaCy focuses on providing effective tool for production usage, rather than teaching. We processed our data and compared the processing results with different packages, and spaCy provided the best performance.

After sentence segmentation, we gained 36,534 valid sentences from contracts and legal documents. Since the original CUAD data set already filtered the sections unrelated to human review, the sentences we got were highly related to our requirements. Thus, we named them as **candidate sentences** and used them for our knowledge graph construction.

```
def get_candidate_sentences(Series):
    candidate_sentences =[]
    for i in range(len(Series)):
        temp = str(Series[i])
        temp = temp.replace('\n', '')
        temp = temp.replace('\n\n', '')# handle the '\n' in sentences
        temp = pattern_1.sub("",temp)
        temp = pattern_2.sub(".",temp)
        temp = pattern_3.sub("",temp)
        sentences = [j for j in nlp_spacy(temp).sents]
        for k in range(len(sentences)):
            if str(sentences[k]) != ' ':
                if str(sentences[k]) != 'nan':
                    candidate_sentences.append(str(sentences[k]))
    return candidate_sentences
```

Figure 8: Screenshot of using spaCy and regular expression to get clean candidate sentences.

Relation triple is the basic element in a knowledge graph. A typical relation triple consists of two entities and one relationship. In the following sections, we introduce the processes of getting our entities and relations from candidate sentences.

2.3.3 NER: Named Entity Recognition

To gain the entities, Named Entity Recognition is required. Named Entity Recognition(NER) is the significant subtask of information extraction. It seeks to locate and classify named entities mentioned in unstructured text. The identified entities are the source of nodes in knowledge graph.

The state-of-art methods for NER is to utilize some pre-trained models like BERT and do fine-tune with extra corpus. The idea behind BERT is called transfer learning. It is a technique to train a high quality model to perform similar tasks on other datasets. Thus, the performance of BERT highly depends on the similarity of tasks. However, most of existing pre-trained NER models are designed to recognize entities in general text, not domain-specific. And there are no

existing pre-trained English NER models for the legal domain, only one in German.

We first tried these general NER models, together with some other notable NER platforms like OpenNLP and spaCy. Whereas, the results were not satisfactory. All models failed to recognise enough desired entities from our candidate sentences. The main reason was most of valuable terms in legal domain were not included in the general models. We retained the recognized entities and tried to explore a new method, rather than depending on existing pre-trained models. Training our own NER model was also impossible. It requires a dozen of legal experts to label the corpus, a extremely huge training dataset and the most advanced hardware. After some exploration, we decided to use **rule-based methods** to solve this problem.

Rule-based NER is to create several rules based on the existing knowledge-base and apply the rules to recognize entities in corpus. To create the suitable rules, we further explored our candidate sentences. We found that the most prominent properties of contract sentences were:

1. Strictly grammatical sentences, which means a sentence has only one root verb.
2. The valuable terms, like the parties, agreements, laws, confidential terms always appeared as subjects and objects in the candidate sentences.
3. Subjects and objects often have modifiers and determiners.
4. The average length of sentences is long and the structure is complex.

Based on these properties, we decided to use **dependency parsing**, a process of analyzing the grammatical structure of a sentence based on the dependencies between the tokens, to analysis the grammatical structures and create our rules. **The main idea was to select the subjects and objects from the sentences. Then, choosing the core subject and direct object base on the**

position of the verb. Finally, extracting the subject and object with their modifiers and determiners together as our entities. According to the above analysis, we implemented the following NER rule-based pipeline with the help of spaCy:

Step 01: First do general NER with the pre-trained models in spaCy.

Step 02: Break each candidate sentence into tokens and get its dependency tag with spaCy.

Step 03: For each token and its dependency parsing result, first check if the current token is a punctuation mark. If token is a punctuation mark then move on to the next token.

Step 04: If the current token is the root verb, record its position in the sentence.

Step 05: If the current token is a compound word, and check if its previous word was also a "compound" then add them together and add them to the "prefix".

Step 06: If the current token is a modifier, check if its previous word was a "compound" then add them together and add them to the "modifier".

Step 07: If the current token is a nominal subject, then prefix and modifier will be added and it will be captured as a candidate of the first entity. Variables such as prefix, modifier will be reset.

Step 08: If the current token is a direct object, then prefix and modifier will be added and it will be captured as a candidate of the second entity. Variables such as prefix, modifier will again be reset.

Step 09: Return the subject and object that are nearest to the root as entity 01 (source node) and entity 02 (target node). If two subjects or objects have the same distance to the root, keep the former one.

Step 10: Check if the gained entity has already been found and labelled in Step 01. If true, add the spaCy's label to this entity.

Liability
Contract
Termination
Repudiation Of Contract
Procedure And Remedy
Limit Of Liability
Legal Liability
Ownership Of Original Content
Use Of Copyrighted Materials
Copyright Infringement Claim
Party For Harm
Indemnity
Direct Claim

Figure 9: Example for entities we got from above NER pipeline.

The screenshot of the code for this pipeline is in **Appendix-D**. After applying the above pipeline, we were happy to found that the resulting entities consist of many valuable elements for contract review, like Termination Clause, Copyright Disclaimer and Indemnification Clause. In total, we got 1162 entities. Compared to previous NER results, our rule-based method successfully extracted enough valuable legal entities from candidate sentences. The comparison results of our methods and previous methods are shown in the evaluation section.

2.3.4 Manual Labelling the Entities

Since most of the entities gained from our rule-based method were unlabelled, we further discussed and defined the categories for our entities, conducted manual entity annotation, calculated and evaluated the Agreement Metric between different labelers to ensure the validity of our labelling.

We went through our entity list, defined 10 categories according to the reference artile *Keywords Standard Terms for Contract* and the general NER labels :

1. Legal Related Term; 2. Organization(ORG); 3. Business Related Term; 4. Person;

5. Legal Related Practitioner; 6. Contract Category; 7. Product; 8. State Or Country;

9. Geopolitical Entity(GPE); 10. General

Among these categories, 2, 8 and 9 were defined by the general NER model in spaCy. The others were based on the properties of legal contracts. We invited two senior law students from a famous University in mainland China to help us, in order to guarantee the correctness of each category's selection and definition, as well as their scope of coverage. After identifying these categories, we labelled all the entities and summed up our results. The statistic of each category is shown in the below table. Meanwhile, we calculated the Cohen's kappa coefficient to ensure consistency across different labelers. The calculation result is shown in the testing and evaluation section.

Label Type	Count	Percentage
Legal Related Term	206	0.177280551
Organization(ORG)	259	0.222891566
Business Related Term	278	0.239242685
Person	17	0.014629948
Legal Related Practitioner	60	0.051635112
Contract Category	5	0.004302926
Product	8	0.006884682
State Or Country	77	0.06626506
Geopolitical Entity(GPE)	3	0.002581756
General	249	0.214285714
Sum	1162	

Figure 10: Statistic about labelling result.

2.3.5 Relation Extraction

To gain the relations between each entities pairs, relation extraction is required. Relationship extraction is the task of extracting semantic relationships from a text. These relationships work as "bridges", linking entity pairs together. From previous analysis, we knew that the valuable entities in a contract clause are the main subject and object. Therefore, the verbs and verb phrases in each contract clause are the relationship between its subject and object. Furthermore, since the contract clauses are strictly grammatical sentences with only one root verb, we could easily identify the relations. The only challenge here was to capture the whole verb phrase.

Since the relationships were based on previous extracted entities, here we continued to use self-defined rule based pattern matching to get them. **The main idea is to first find the root verb by dependency parsing. Once the ROOT is identified, we further check if it is followed by a preposition or an agent word. Finally we add the preposition and the agent word to the root verb and form a complete relation.** In total, we gained 2094 valid relations from RE.

```
def get_relation(sent):

  doc = nlp_spacy(sent)

  # Matcher class object
  matcher = Matcher(nlp_spacy.vocab)

  #define the pattern
  pattern = [{'DEP':'ROOT'},
             {'DEP':'prep','OP':"?"},
             {'DEP':'agent','OP':"?"},
             {'POS':'ADJ','OP':"?"}]

  matcher.add("matching_1", [pattern])

  matches = matcher(doc)
  k = len(matches) - 1

  span = doc[matches[k][1]:matches[k][2]]

  return(span.text)
```

Figure 11: Relation Extraction with rule-based pattern matching.

After getting the labelled entities pairs and their relations, 2094 relation triples were constructed. To guarantee the quality of relation triples, we manually filtered each of them and cleaned the inaccurate triples. As a result, 1709 accurate relation triples were gained and served as the primary content for knowledge graph construction.

2.3.6 Knowledge Enrichment

The relation triples we gained from CUAD dataset were valuable information in contract clauses. However, understanding some terms in a contract may require extra legal knowledge. For example in the relation triple *Signed employee - agrees - the Termination Clause*, people without any legal background may wonder what is a "Termination Clause". As our project objective was to build up a knowledge graph for legal contract and aid the users' contract review. Some external legal knowledge should be enriched to our knowledge base.

To implement knowledge enrichment, we utilized several professional legal documents as external sources to gain more valuable legal knowledge for contract review. These professional legal documents includes **Legal Keywords And Standard Terms** from Microsoft Keywords Studio, materials from **Law Insider** and resources from **ContractWorks**. More details of these professional legal documents are discussed in the Discussions Section. We adopted the same pipeline to handle the data and gain the extra relation triples. The results were quite satisfactory. In total, 418 relation triples were gained. And most of them help a lot in providing extra legal knowledge. For example, we gained the relation triples like *"Termination Clause - Describe - Termination Procedure And Remedy", "Termination Clause - Also Known As - Severance Clause", "Termination Clause - Preset - Termination Payment"*. These extra relation triples provided very useful information for general users to understand their contracts and review the

clauses. The statistic of labelled entities after knowledge enrichment is as follows.

Label Type	Count	Percentage
Legal Related Term	292	0.226356589
Organization(ORG)	259	0.200775194
Business Related Term	310	0.240310078
Person	17	0.013178295
Legal Related Practitioner	60	0.046511628
Contract Category	15	0.011627907
Product	8	0.00620155
State Or Country	77	0.059689922
Geopolitical Entity(GPE)	3	0.002325581
General	249	0.193023256
Sum	1290	

Figure 12: Statistic about labelling result after knowledge enrichment.

The manually filtered relation triples we gained from CUAD, as well as the enriched triples from professional legal documents together formed the main component of our legal contract knowledge graph.

2.3.7 Neo4j database for storing, querying and applying graph algorithms

To store the relation triples and and apply graph algorithms for further exploration, we chose Neo4j, a graph database management system with native graph storage and processing, as our database. Neo4j is the most famous graph database, not only because it delivers the lightning-fast read and write performance, but also because it supports various Plugin Libraries such as GDSL(Graph Data Science Library).

In Neo4j, we stored our entities as nodes according to their labelled categories, and built relationships. Afterwards, query statements were also implemented.

We used Cypher query language, Neo4j's default graph query language, to retrieve data from

"graphName"	"database"	"memoryUsage"	"sizeInBytes"	"nodeProjection"	"relationshipProjection"
"LKG"	"neo4j"	"581 KiB"	595644	{"State_or_Counrty":{"label":"State_or_Counrty","properties":{}},"PERSON":{"label":"PERSON","properties":{}},"ORG":{"label":"ORG","properties":{}},"Business_Related_Terms":{"label":"Business_Related_Terms","properties":{}},"Contract_Categories":{"label":"Contract_Categories","properties":{}},"Product":{"label":"Product","properties":{}},"General":{"label":"General","properties":{}},"GPE":{"label":"GPE","properties":{}},"Legal_Related_Terms":{"label":"Legal_Related_Terms","properties":{}},"Legal_Related_Practitioner":{"label":"Legal_Related_Practitioner","properties":{}}}	{"Relation":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"Relation","properties":{}}}

Figure 13: The deployed Neo4j database

the graph. To realize fuzzy search, we added Regular expression in our Cypher queries. For example, the following Cypher query retrieve all nodes and edges has "elementToSearch" included in their names:

```
MATCH     p = (n)-[r]-(m)
WHERE     n.name = ~.*elementToSearch.*
   OR     n.relation = ~ .*elementToSearch.*
   OR     r.relation = ~.*elementToSearch.*
   OR     m.name = ~.*elementToSearch.*
   OR     m.relation = ~'.*elementToSearch.*
RETURN    *
```

We utilized GDSL(Graph Data Science Library) to ran several graph algorithms to get valuable node properties and insights from our knowledge graph. The algorithms we chose includes **Page Rank, Label Propagation, Weakly Connected Components and Kamada Kawai algorithm**.

The Page Rank algorithm measures the importance of each node within the graph based on the number of incoming relationships and the importance of the corresponding source nodes. In order to visualize the centrality and the significance of each node in the graph, we scaled the

nodes' size based on their page rank scores. Meanwhile, the page rank scores were stored as node property, and users can easily check it by hovering the mouse on the nodes.

```
CALL gds.pageRank.write('LKG', {
  maxIterations: 20,
  dampingFactor: 0.85,
  writeProperty: 'pagerank'
})
YIELD nodePropertiesWritten, ranIterations
```

Figure 14: Ran PageRank Algorithm

Both the Label Propagation and the Weakly Connected Components are algorithms that aim to detect communities in the graph. Kamada Kawai algorithm is an algorithm to generate Force-directed graph drawing. The main purposes of these algorithms we used here were to generate visualizations to help the users to evaluate how groups of nodes are clustered or partitioned, as well as their tendency to strengthen or break apart.

The results showed that the Label Propagation did a more localized clustering than the Weakly Connected Components, and the Force-directed graph generated by Kamada Kawai algorithm provided the best cluster visualization. As a result, we kept the Kamada Kawai algorithm results in our system and made it as one of our four types KG visualizations. The screenshot of these visualizations, including the Kamada Kawai layout are shown in the following **Knowledge Graph visualization** section.

2.3.8 Knowledge Graph Visualization

To visualize our knowledge graph in our front end web application, we used **Neovis.js (2.0.0-alpha.9)** package. Neovis.js is a library that can combine JavaScript visualization and Neo4j database in seamless integration. The outstanding advantage of Neovis.js is that the data format it expects aligns with the database, as it is built with Neo4j's property graph model in mind. Compared to other knowledge graph visualization packages, our advanced alpha-version can handle large amounts of dynamic data and enables manipulation and interaction with the data. Most important, it allowed customized visual styles (e.g., color combinations, font styles and node styles) and multiple the interaction functionalities (e.g., mouse hovering, select and delete nodes, highlighting and dragging).

To provide the best visualizations and meet the users' needs, we designed and implemented four types of different visualizations, which are: **Circular Type(default), Force-directed(Cluster) Type, Hierarchical Type and Dynamic Physical Type**.The hierarchical type is designed for the query result, since we believe a hierarchical structure can show the inner relations among a graph clearly and help users to gain their desired information. We show screenshots of each type of visualization below.

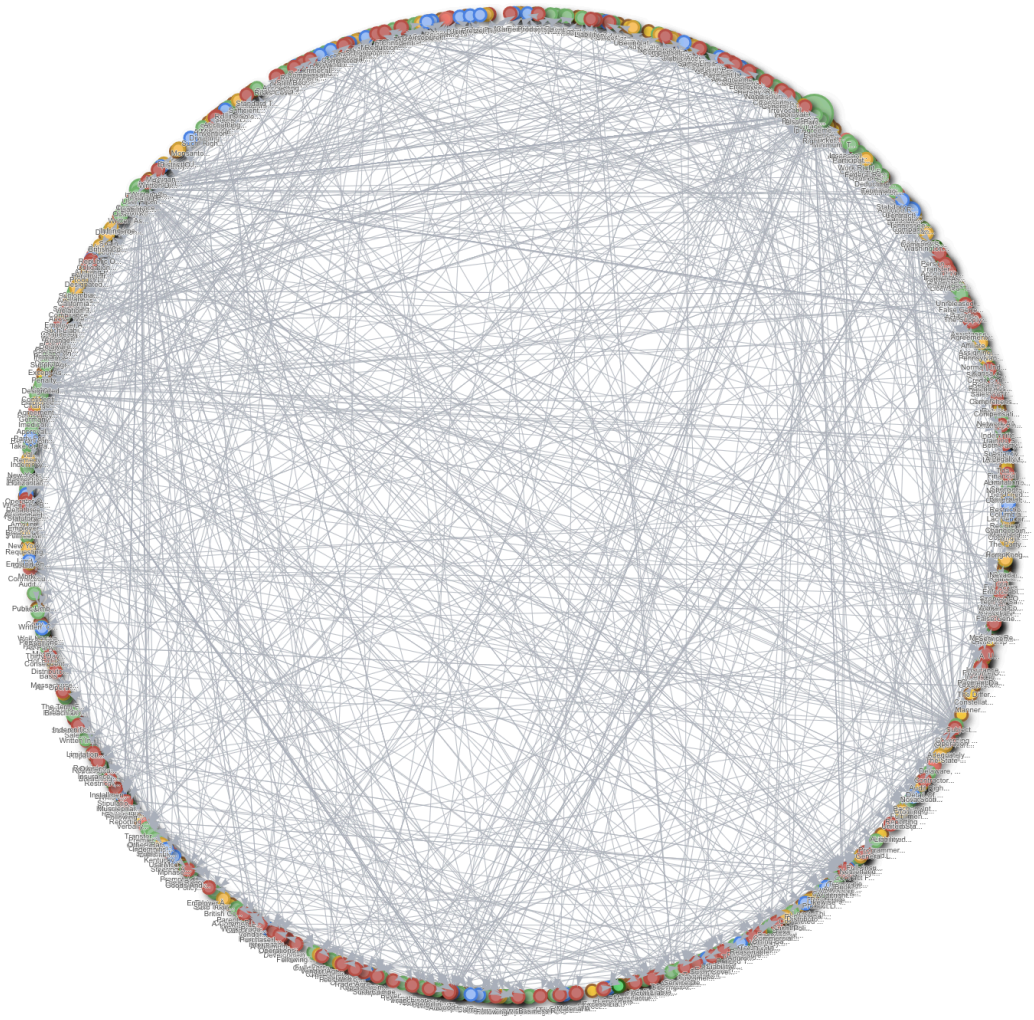

Figure 15: Circular Type visualization layout.

The Circular Type is the default visualization type. It is a concentric circle layout. All nodes are aligned around the circle and the edges connect them together. Each of node is colored based on it label and the size of node is proportional to the PageRank value. This is the default type visualization of our knowledge graph.

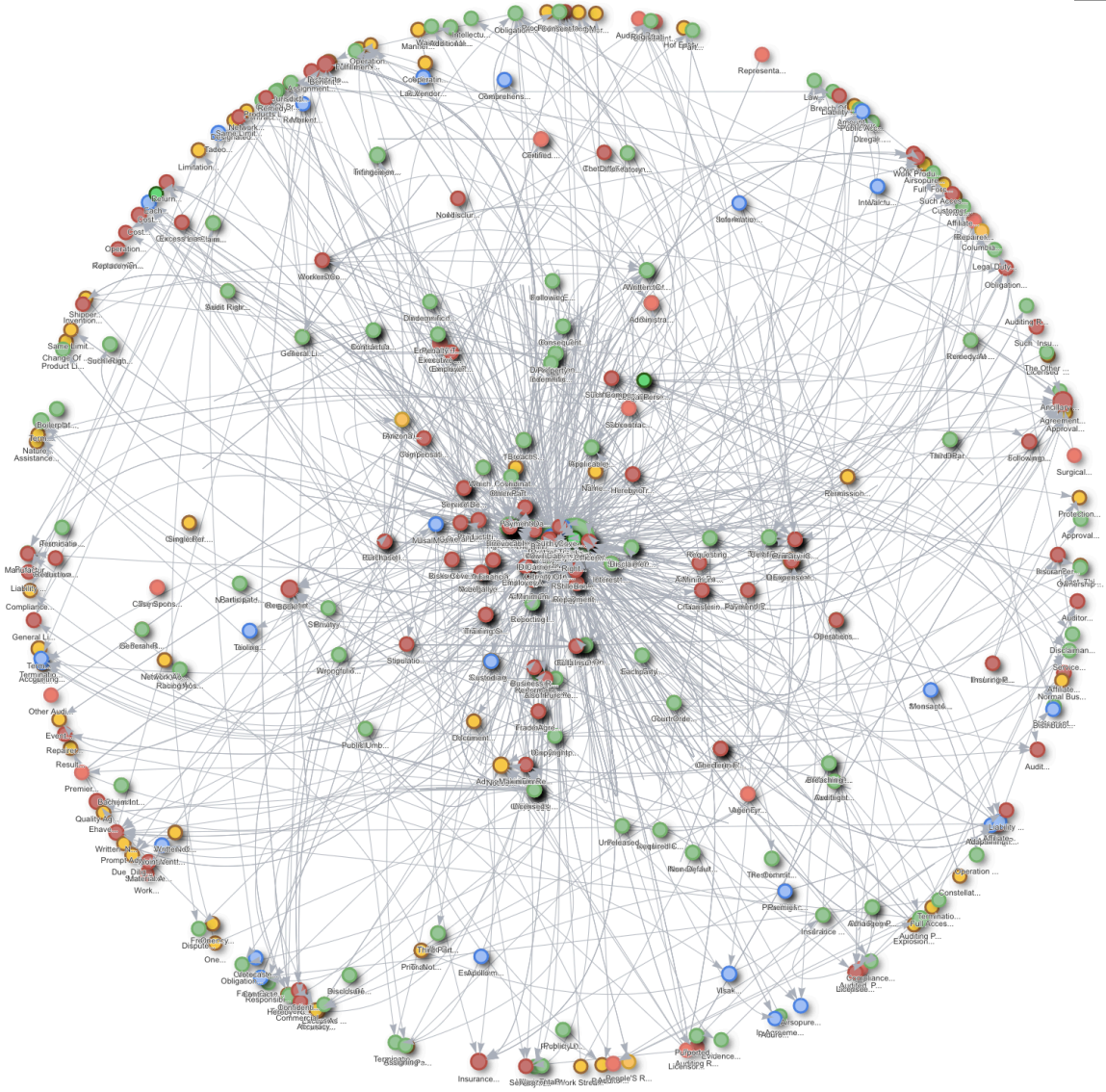

Figure 16: Force-directed Type visualization layout.

The Force-directed Type is a Force-directed graph generated from the Kamada Kawai algorithm.The spring-like attraction force based on Hooke's law is used to attract pairs of endpoints at the edge of the graph to each other, while the repulsive force of charged particles based on Coulomb's law is used to separate all pairs of nodes.

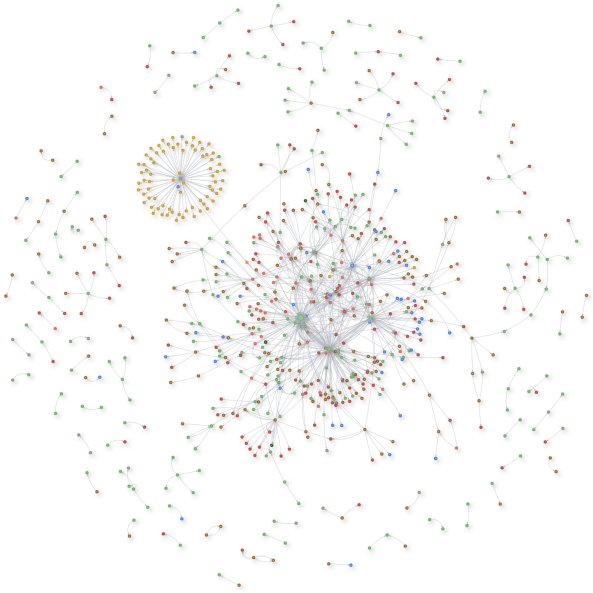

Figure 17: Dynamic Physical Type visualization layout.

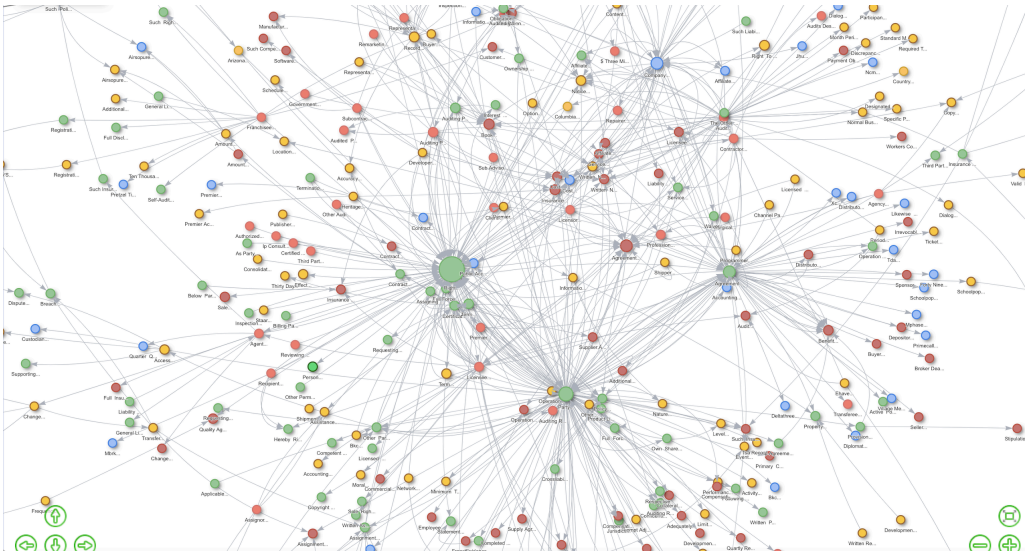

Figure 18: A closer look at Dynamic Physical visualization.

The Dynamic Physical Type is generated by Barnes Hut simulation, which is a quadtree based gravity model. Each node in this tree represents a region of the three-dimensional space. The advantage of this visualization is it automatically separates all points clearly and provides interesting interactions to the users.

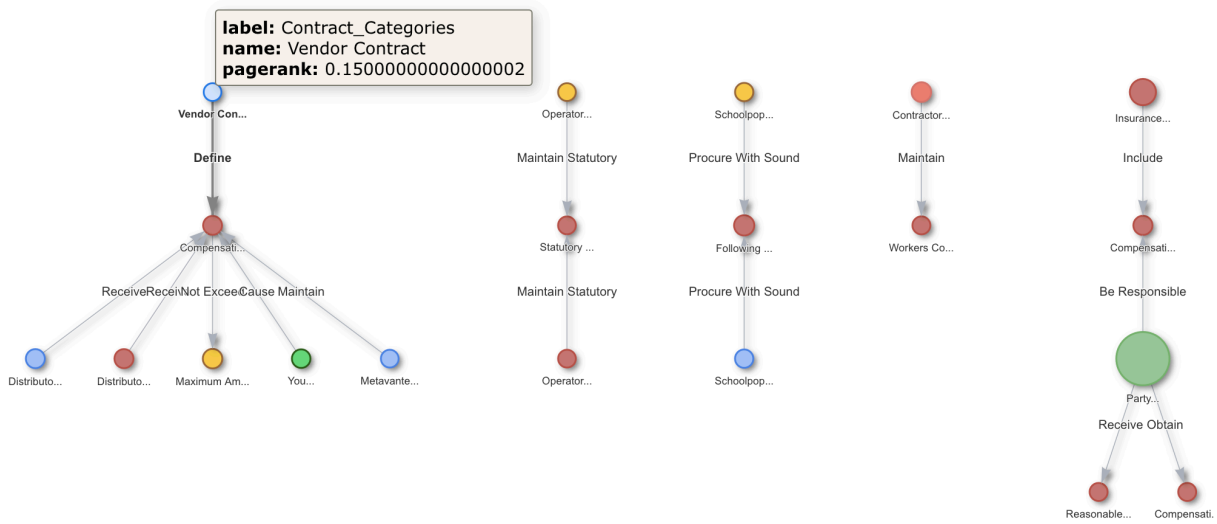

Figure 19: Hierarchical Type visualization layout.

The Hierarchical Type positions the nodes in a hierarchical fashion. This visualization type is designed to help users obtain information more effectively in search results. Thus, we ascertain the levels of the nodes based on the data: The nodes with the most edges are put at the top. The rest levels of the hierarchy is evaluated in the same method. Since this visualization is designed for visualize user query result, the demo above is the visualization of searching “Compensation” in the search box.

In the next section, we introduce and demo the interactive functionalities that all our visualizations support.

Interactions in Visualizations For all above visualizations, we implemented many useful interactions for users, which includes hover, dragging, selection, multiple selections and note delete. In the following section, we demo each of them in the default Circular type visualization.

Uses can hover over the nodes and edges. When a node or an edge is hovered, it will be highlighted and its properties will be shown. A infoBox will pop up and show the node's name, label and PageRank score.

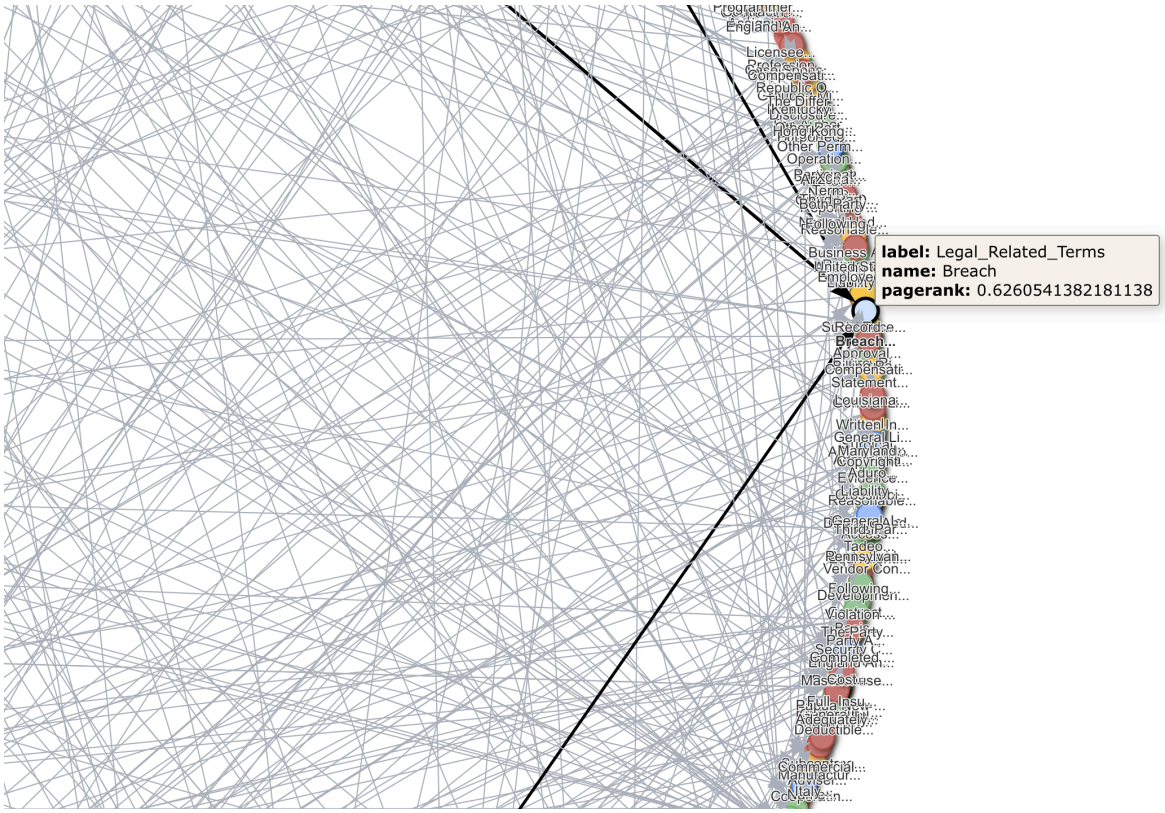

Figure 20: Demo for hovering node in Circular Type

The hover function provide the properties information of the selected node to the users. We planned to add the original sentence of the selected node at first. However, due to the length of clause sentences, we have removed this feature.

Users can drag nodes to anywhere they want. By doing this, users can generate custom layouts. Meanwhile, users can select the nodes and edges by simply click. A longheld click as well as a control-click will add a node/edge to the multiple selection. The selected nodes and edges can be deleted from the visualization by simply double click. These functions also allow the users to generate custom layouts.

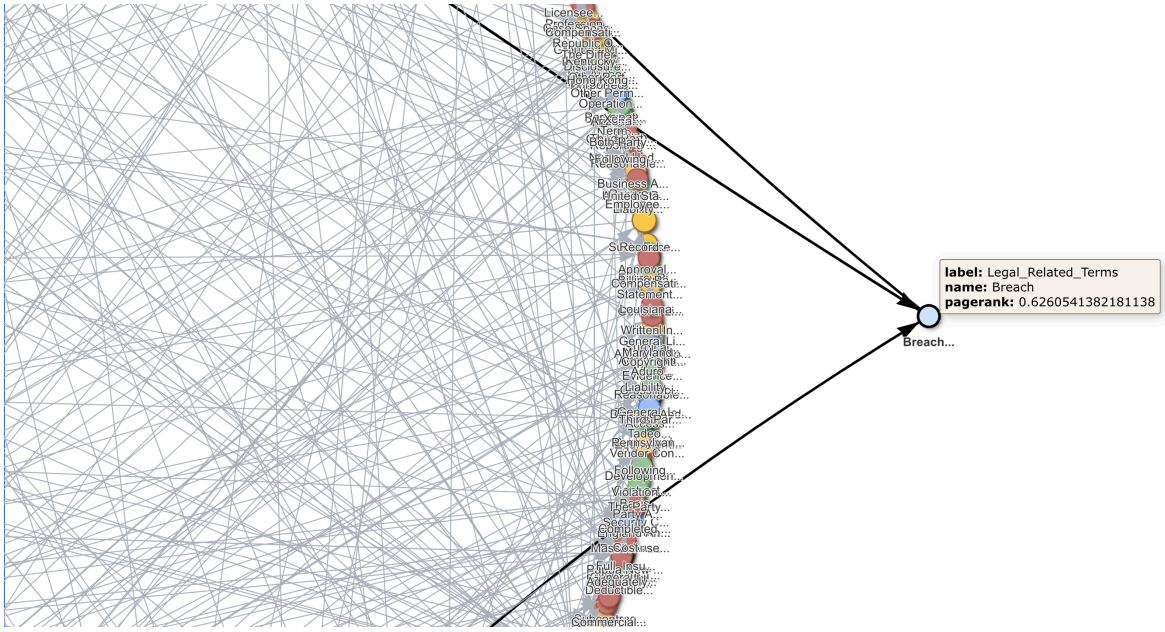

Figure 21: Demo for dragging node in Circular Type

The main purpose of these interactions is to aid user's information query and support customized graph layout in the analysis report.

2.3.9 Pipeline for Contract Review

To further aid the users review their contract, we built a pipeline for contract review.

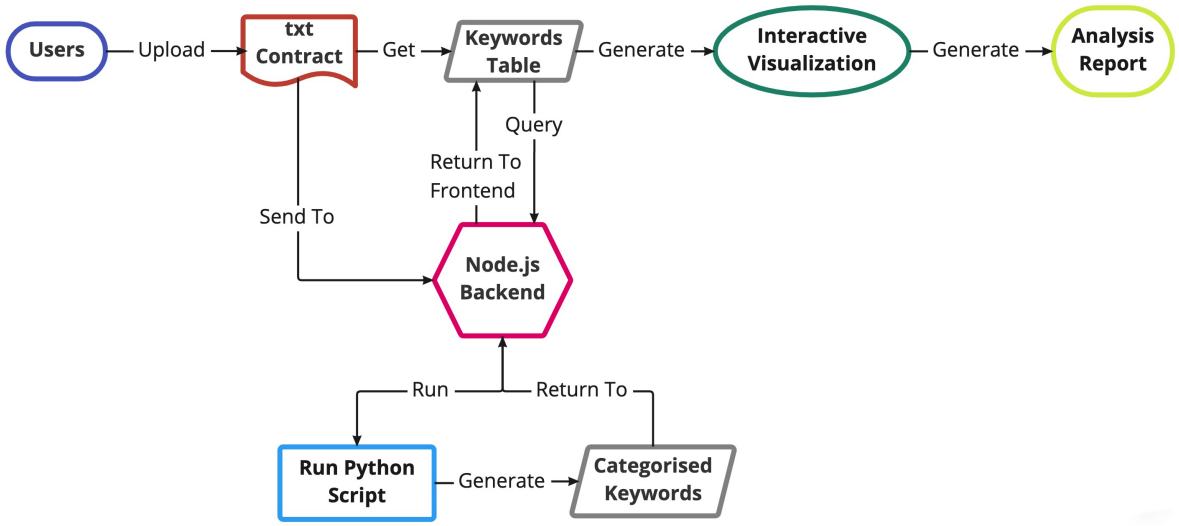

Figure 22: Pipeline workflow for contract review

According to the reference article *6 Key Clauses in Commercial Contracts* and *Keywords Studio*, we defined 6 lists of words, each of them summarized the most common words appear in one essential clause. The main idea of contract review is: **To search, locate and highlight the keywords that belongs to these 6 essential clauses and alert user to potential omissions in their contract if there are undetected categories. Meanwhile, users can search the keywords to gain more information in the knowledge graph. The interaction functions in previous section allow uses to check the retrieved information, select and retain the useful knowledge. Finally, we generate an in-depth review report for this contract and allow users to download it.**

To run python scripts from backend Node.js, we utilized *"spawn"* method of *"children process"*, which spawns the child process asynchronously. In the Python script, we utilized **Pandas** library

A	B	C	D	E	F
Confidential Terms	**Payment Terms**	**Disclaimers**	**Warranties**	**Dispute Resolution Terms**	**Force Majeure**
Private	Sale	Income	Warranty	Dispute	Force majeure
Secret	Vendor	Loss	Defects warranty	Governing law	Circumstance
Confidential information	Pay	Revenue	Deliverables warranty	Jurisdiction	Beyond
Provision	Payment	Termination	Claim	Dispute Notice	Flood
Confidentiality	Discount	Liquidation	Customer warranty	Dispute resolution	Drought
Obligation	Consumer	Disclaimer	Implied warranty	Resolve	Earthquake
Disclose	Wholesale	Indemnity	Promise	Clause	Disaster
Confidentiality exception	Cash	Compensation		Court	Terrorist attack
Disclosure	Credit	Indemnification			Civil war
Security	Bill	Infringement			Force
Restriction	Coupon	Limitation			Explosion
Penalty	Payment method	Obligation			
Remedy	Trade	Breach			
Confidential information	Rebate				
Disclose					
Disclosure					
Privity					
Exception					
Confidentiality					
Breaching					

Figure 23: The keywords for each of 6 essential clauses

to spilt contracts sentences into word lists. Then we used *"Stopwords"* in **NLTK** package to remove the common words from the list. We also used **Inflect** package to eliminate duplicated words and unify their singular, plural and capitalization. Finally we matched the list with our keywords table and the entity list of our knowledge graph. The backend returns each matching keyword by category. Words not in our keyword list but in our knowledge graph will also be returned as "other". The whole process was bonded with a button in our front end, thus the users could simply start the analysis by a click.

Confidential Terms	Payment Terms	Disclaimers	Warranties	Dispute Resolution Terms	Force Majeure	Others
Confidentiality	Credit	Compensation	Claim	Dispute	Force	Unit
Exception	Discount	Infringement	Warranty	Court	Beyond	Record
Private	Payment	Breach		Clause		Report
Obligation	Trade	Limitation				It
Security	Pay	Indemnity				Schedule

Figure 24: Example of analysis result: A keywords table with categories.

Users can search each keywords by clicking, and a hierarchical searching result will be returned. After checking, selecting and generating the desired graph, users can add the visualization to their analysis report.

CReKG

Analysis Report for:

SITE DEVELOPMENT AND HOSTING AGREEMENT

Keywords Table:

Confidential Terms	Payment Terms	Disclaimers	Warranties	Dispute Resolution Terms	Force Majeure	Others
Obligation	Consumer	Revenue	Promise		Force	Portion
Disclosure	Payment	Breach				Record
Secret	Sale					Balance
Disclose	Pay					Termination
Provision	Trade					Customer

Notice:

This contract may lack of "Dispute Resolution Terms".
Dispute resolution includes approaches that prevent disputes or enable parties to manage and resolve their disputes without intervention. It's an important clause in a contract. Double check is recommended.

Information Graphs:

Figure 25: Example for the generated Report

We designed a template for analysis report. Basically it lists the categorised keywords and the retrieved graphs. Also, if the contract lacks of some of the six essential clauses, a warning message will be included. We generate and show the report in a separate webpage in our system and allow the users to download. This function is realized by the **DocRaptor**, a document conversion API that converts HTML to PDF and Excel format.

2.4 Testing

During the implementation, we performed unit testing on functions and modules to make sure that each of them were bug-free. After we finished building the whole system, integration testing was conducted to ensure the functionality of our system. We tested the model used for NER and relation extraction, the database, the visualization and the user interface.

2.4.1 Test the Validity of Manual Labeling

To test the validity of manual labeling, we calculated the Cohen's kappa of our label results. Cohen's kappa coefficient (κ) is a statistic that is used to measure intra-rater reliability for categorical items. Below is the definition of Cohen's kappa from Wikipedia:

Cohen's kappa measures the agreement between two raters who each classify *N* items into *C* mutually exclusive categories. The definition of κ is

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e},$$

where p_o is the relative observed agreement among raters, and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly seeing each category. If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters other than what would be expected by chance (as given by p_e), $\kappa = 0$. It is possible for the statistic to be negative,[6] which can occur by chance if there is no relationship between the ratings of the two raters, or it may reflect a real tendency of the raters to give differing ratings.

Figure 26: Definition and formula for Cohen's kappa coefficient

According to our label results, the overall Cohen's kappa coefficient is around 0.833 and could be defined as **Almost Perfect** (0.81-1).

2.4.2 Test the Model for NER and Relation Extraction

To test whether our model could successfully retrieve entities and relations from the sentence, we randomly generated a subset of 200 sentences and manually checked whether the retrieved entities and relations were meaningful and followed the pipeline stated in Section 2.3.3.

2.4.3 Test the Database

We first checked the data integrity, whether all extracted entities and relations were stored in the database.

Besides, we also conducted functional database testing. This is used to test whether the operations performed by the end-users which are related to the database work as expected. We tested the fuzzy search function with different test cases, including null values, full name of entities and relations, substrings of entity names and relation names, arbitrary strings that were not included in entity and relation. We checked whether the retrieved nodes and edges all satisfied the condition and whether all nodes and edges satisfying the condition were retrieved.

2.4.4 Test the Visualization of the Knowledge Graph

We checked functionalities including hover, dragging, selection, and node deletion for all visualization style. We also tested the visualization with randomly generated data to check the color of nodes and functionalities under all visualization style.

2.4.5 Test the User Interface

We tested all the functions on the website, including uploading, downloading and all the visualization of knowledge graphs. We also tested the compatibility of our system with different browsers, including Internet Explorer, Google Chrome, Firefox and Safari.

To conduct the user study, we invited 20 volunteers in total to use CreKG and collected their feedback, 10 of which had legal backgrounds. They were asked to complete the questionnaires and give comments on the user interface design.

Question (5 for excellent, 1 for poor)	**Agv. Score**
How do you rate the user interface design in general?	3.8
How do you rate the circular type visualization of the knowledge graph?	3.7
How do you rate the dynamic physical type visualization of the knowledge graph?	4.0
How do you rate the force-directed type visualization of the knowledge graph?	3.2
How do you rate the hierarchical type visualization of the knowledge graph?	4.2
How do you rate the functionality of the system in terms of performance?	4.2
How do you rate the design of contract review process?	3.8
To what extent, do you think the system is user-friendly?	4.4
To what extent, do you think the system help you do contract review?	4.1
To what extent, do you think the searching result and keywords table help you do contract review?	4.4
To what extent, do you think the generated report help you do contract review?	4.0

Table 1: Result of User Study

Volunteers also mentioned that detailed user guide could be added to explain the functions on the website. Another suggestion was that more information could be added to the nodes, for example the original sentences in the contracts.

2.5 Evaluation

In this section, we demonstrate evaluation methods we used to show the effectiveness of our CReKG system and how well we achieved our objectives listed in section 1.2.

2.5.1 Evaluation for the knowledge graph

For the construction of knowledge graph, every step is verified during the development and the testing section. The Cohen's kappa coefficient **(0.833)** could be defined as **Almost Perfect**, which demonstrates that our manual labelling is highly effective. Different labelers follow a same labelling standard and the labelling result is valid.

Meanwhile, our NER and relation extraction methods are proved to be effective on the testing data. A brief statistic is as follows:

Table 2: Statistic of valid relation triples

Source	Valid triples	Total triples	Pass rate
CUAD	1709	2094	0.8161
External legal doc.	418	488	0.8565
Testing set	141	183	0.7705

The query testing was run 30 times and we got a 100 percent pass rate. In each run, our database successfully returned the relation triples contain the searched keyword.

The functionalities test on visualizations also shown that our system is robust. In total, 26 out of 30 tests successful returned the desired visualizations. Four test failed to show the visualizations due to insufficient resource for websocket connection. This is due to the returned graphs had

to many information. If the user repeatedly shows the full knowledge graph, there is a small probability of failure. One easy solution to this problem is to refresh the webpage and show the visualization again.

The processes in our knowledge graph construction, and the KG we built are verified to be valid and robust. According to the above evaluation, we successfully fulfilled the objectives 1, 2, 3, 4 and 5.

2.5.2 Evaluation for the contract review pipeline

According to the results and feedback from user study, most of the users agreed that our contract review pipeline is valid and helpful. We got a **4.1 out of 5** for the functionalities of contract review pipeline, a **3.8 out of 5** for the pipeline design and a **4.4 out of 5** for the helpful searching result and final report. Since half of the participants had a legal-related background and half of them were general users, the above feedback could prove that our contract review pipeline is helpful and effective.

According to the above evaluation, we successfully fulfilled the objectives 6 and 8.

2.5.3 Evaluation for the system

Based on the user study result, we got a **4.2 out of 5** for system performance. Meanwhile, the tests about database and visualizations also demonstrated the validity of our system. The insufficient resource problem may require further improvement and testing. Since no participants reported this issue during their experiments, we could conclude it did not affect the normal usage.

According to the above evaluation, we successfully fulfilled the objective 7.

2.5.4 Summary of evaluation

As shown above, we fulfilled all our initial objectives, built up a helpful knowledge graph, implemented a valid contract review pipeline and realized a useful and relatively robust system.

3 Discussions

The main objective of our project was to implement a system to assist the contract reviewing process with the help of knowledge graph. Our system allows users to extract keywords from the uploaded contract and search the keywords retrieved from the contract in our knowledge graph. Users are also allowed to explore the knowledge graph themselves, looking for the relationship and interactions between legal terms and clauses. In this section, we illustrate the challenges we met, our solutions to the problems, limitations of our system and possible improvements.

3.1 Challenges and Solutions

The first challenge we met was the choice of dataset. We proposed to use the US supreme court documents and focused on the contract-related cases. However, we found a better dataset with annotations from professionals, the CUAD. Compared with US supreme court documents, the CUAD contains fewer errors in sentences, which facilitates the data preprocessing. Another advantage of the CUAD is that sentences in the contracts are categorized by legal professionals. Thus, the knowledge graph constructed can be more meaningful in terms of contract reviewing.

We also changed our design of database. We proposed to use the relational database to store the extracted entities and relations. With further research, we decided to use Neo4j, a graph database management system. Compared with relational database, Neo4j stores data as nodes, edges and attributes, which is more suitable for the storage of the knowledge graph.

We also had difficulties in customizing our knowledge graph. Popular front-end frameworks, like Vue.js is powerful in customization, but they cannot connect to our Neo4j database directly

and the format of data should be transformed before use. On the other hand, most visualization libraries that support Neo4j are lack of customization functionalities. To solve this problem, we finally found an alpha-version of Neovis.js. Even though the documentation of this library is not yet completed, we learned how to use it through trial and error and utilized it to solve our problem.

3.2 Limitations and Improvements

Due to the large number of entities and relations, the visualization of the full knowledge graph (Dynamic Physic type) may take more than 10 seconds(mean = 12.4 in our testing), which affects the user experience. For the future development, GPU rendering can be considered as a possible solution to this problem.

Another possible improvement is the design of the knowledge graph. One of the features of legal documents is that they are carefully worded to ensure accuracy and avoid misunderstandings. As a result, one sentence may contain several clauses, such as attributive clauses and adverbial clauses. The entities we extracted may not include the information contained in these clauses. To solve this issue, we would like to suggest a new design of the knowledge graph with clauses included as supplementary information of the entities.

4 Conclusions

In this project, we aimed to propose a legal Knowledge Graph-based automated contract review system to improve the contract review efficiency of law firms and contract parties, enabling lawyers and contract parties to focus more on other high-value areas. We constructed a knowledge graph of contracts and implemented a website for contract reviewing and knowledge graph visualization. Overall, the system successfully allows user to upload their contracts and review it by exploring the Knowledge graph and finally get a detailed report.

Looking back over the past year, we have tried a lot of different plans which may waste us a lot of time. Given none of us is an expert in front end development, we spent much time on learning Neovis.js package and react framework etc. Given the time constraint, we could only try our best to provide the basic functionality. Therefore, some improvements on the user interface can still be made to optimize our system in the future. Also, we have made changes in the way to store our Knowledge Graph data during the developing due to our inexperience in storing graph data which affected our schedule.

However, we finally completed all of our objectives for this interesting and meaningful topic. We are very glad to have the chance to research and develop this system which can truly help users improve their contract review efficiency. And we are excited that our full-stack system can solve the lack of legal knowledge problem to fill the corresponding gap by applying Knowledge Graph to contract review for the first time. If we are given more time, we could definitely provide a more exquisite and user-friendly web user interface to let lawyers better review their contracts.

5 References

[1] Manual rule-based automated provision extraction software case study: Mumboe, Mar 2022.

[2] No rules: Problems with rules-based contract provision extraction, Feb 2022.

[3] WONG Pui Ying CHAN Tsz Ho, TANG Yutian and ZHANG Liangwei. A knowledge graph to better understand medical text about covid-19. 2021.

[4] Angus Chudleigh. What is automated contract review? Jun 2020.

[5] clio. 2018 legal trends report. 2018.

[6] Jaspreet Singh Dhani, Ruchika Bhatt, Balaji Ganesan, Parikshet Sirohi, and Vasudha Bhatnagar. Similar cases recommendation using legal knowledge graphs. *CoRR*, abs/2107.04771, 2021.

[7] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. Cuad: An expert-annotated nlp dataset for legal contract review, 2021.

[8] Beverly Rich. How ai is changing contracts, Feb 2018.

[9] Kira Systems. Machine learning based automated contract provision extraction. 2022.

A Appendix A: Project Planning

A.1 Distribution of Work

Task	Zixin	KaShun	Bingying
Do the Literature Survey	✓	✓	✓
Analyze Detection Methods		✓	
Design the KG-related Algorithm		✓	
Design the Database		✓	
Design the User Interface	✓		
Design the System	✓		
Develop the KG-related Algorithm		✓	
Develop the Database		✓	
Develop the User Interface	✓		
Develop the System	✓		
Test the KG-related Algorithm			✓
Test the Database			✓
Test the User Interface			✓
Test the System			✓
Perform Integration Testing			✓
Evaluate the KG		✓	
Evaluate other components		✓	
Write the Proposal	✓	✓	✓
Write the Monthly Reports	✓	✓	✓
Write the Progress Report	✓	✓	✓

Write the Final Report	✓	✓	✓
Prepare for the Presentation	✓	✓	✓
Make the Project Video Trailer	✓	✓	✓

Table 3: Distribution of our Work

A.2 GANTT Chart

Task	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Do the Literature Survey	■	■	■							
Analyze Detection Methods		■	■	■						
Design the KG related Algorithm				■	■					
Design the Database					■	■				
Design the User Interface					■	■				
Design the System					■	■	■			
Develop the KG related Algorithm				■	■	■				
Develop the Database						■	■	■		
Develop the User Interface						■	■	■		
Develop the System						■	■	■		
Test the KG related Algorithm								■	■	
Test the Database								■	■	
Test the User Interface								■	■	
Test the System								■	■	
Perform Integration Testing								■	■	

Evaluate the KG										
Evaluate other components										
Write the Proposal										
Write the Monthly Reports										
Write the Progress Report										
Write the Final Report										
Prepare for the Presentation										
Make the Project Video Trailer										

Table 4: GANTT Chart for our project

B Appendix B: Required Hardware and Software

B.0.1 Hardware

Hardware List	Detail and Purpose
Development PC	PC with macOs 10.14 or later
Development PC	PC with Windows 10 or later for compatibility test
RTX 2080 * 2	For Model Training
Server PC	Linux with 1TB hard drive

Table 5: List of Hardware we will use in the project

B.0.2 Software

Software List	Detail and Purpose
HTML, Neovis.js (2.0.0-alpha.9)	For our Front End
Neo4j	For our Database and Back End
Python, spaCy	For Model Implementation
Github	Development Platform for Team Work
Overleaf	For Report Writing
VSCode	IDE for code development

Table 6: List of Software we will use in the project

C Appendix C: Meeting Minutes

C.1 Minutes of the 1st Project Meeting

Date: Aug. 21st, 2021

Time: 20:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: TU Bingying

1 Discussion Items

1.1 We shared our understanding of knowledge graphs and the project. Since most of us have not had much experience in building knowledge graphs, we planned to study the basics of knowledge graphs during the summer.

1.2 We listed all the possible topics that the project may involve.

1.3 We discussed the presentation of our project. All of us agreed that it was better to come up with a product as the final result, for example a website or an app on a smartphone.

1.4 We discussed the timeline of our future work. We will start to do a literature survey in the following week.

2 Goals before the Next Meeting

2.1 We will start the literature survey and share helpful articles among team members.

2.2 We will begin to think about the distribution of work.

2.3 We will learn some possible techniques or knowledge to use in the project.

3 Meeting Adjournment

The meeting was adjourned at 20:42 p.m.

C.2 Minutes of the 2nd Project Meeting

Date: Sep. 14th, 2021

Time: 10:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: CHEN Zixin

1 Discussion Items

1.1 We discussed the new datasets shared by Professor.

1.2 We discussed detailed design of our project including the type of entities and relations.

1.3 We discussed the timeline of our future work.

2 Goals before the Next Meeting

2.1 We will discuss with Professor on the inter- and intra- document knowledge graphs.

2.2 We will start to write our proposal.

2.3 We will search for more datasets that can be used as our corpus.

2.4 We start the process the data.

3 Meeting Adjournment

The meeting was adjourned at 10:55 p.m.

C.3 Minutes of the 3rd Project Meeting

Date: Oct. 26th, 2021

Time: 9:30 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: SHUM Ka Shun

1 Discussion Items

1.1 We shared the papers and models we found on NER.

1.2 We discussed the dataset on hands and evaluated them.

1.3 We discussed our future work before meeting with Professor.

2 Goals before the Next Meeting

2.1 We will start to construct the framework of our system.

2.2 We will try to perform NER and relation extraction algorithms on the dataset.

3 Meeting Adjournment

The meeting was adjourned at 10.28 p.m.

C.4 Minutes of the 4th Project Meeting

Date: Nov.16th, 2021

Time: 11:00 a.m.

Present: Professor SONG Yangqiu, CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: TU Bingying

1 Discussion Items

1.1 We presented a subgraph we had constructed using the sentences in the category governing law.

1.2 Professor Song answered our questions about NER models in legal domain and gave us suggestions on performing NER and relation extractions.

1.3 Professor Song gave us suggestions on the objectives of our project.

1.4 We discussed the timeline of our future work.

2 Goals before the Next Meeting

2.1 We will start the build the framework of our system.

2.2 We will explore more NER models and evaluate their accuracy.

2.3 We will start to develop our database.

2.4 We will modify our design of user interface.

3 Meeting Adjournment

The meeting was adjourned at 11:31 a.m.

C.5 Minutes of the 5th Project Meeting

Date: Dec.22nd, 2021

Time: 10:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: CHEN Zixin

1 Discussion Items

1.1 We discussed methods to improve the performance of NER models.

1.2 We discussed the graph algorithms we would apply.

2 Goals before the Next Meeting

2.1 We will start to visualize the subgraphs we already have.

2.2 We will think about the functions we can develop based on the knowledge graph we construct.

2.3 We start to develop the user interface.

3 Meeting Adjournment

The meeting was adjourned at 10:28 p.m.

C.6 Minutes of the 6th Project Meeting

Date: Feb.7th, 2022

Time: 21:30 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: TU Bingying

1 Discussion Items

1.1 We discussed the services we could provide based on the knowledge graph we constructed.

1.2 We discussed the design of our user interface and the visualization of the knowledge graph.

2 Goals before the Next Meeting

2.1 We will start to implement the functions we discussed.

2.2 We will start to modify our user-interface to make it more user-friendly.

2.3 We will begin to write the progress report.

3 Meeting Adjournment

The meeting was adjourned at 10:00 p.m.

C.7 Minutes of the 7th Project Meeting

Date: Mar.13th, 2022

Time: 20:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: TU Bingying

1 Discussion Items

1.1 We discussed the ways to join our subgraphs together.

1.2 We discussed the improvements we could made on the extraction of entities and relations.

1.3 We discussed the adjustments to the contract review pipeline.

2 Goals before the Next Meeting

2.1 We will start to improve the performance of the rule-based models.

2.2 We will start to manually label the entities.

2.3 We will begin to write the final report based on the progress report.

3 Meeting Adjournment

The meeting was adjourned at 21:12 p.m.

C.8 Minutes of the 8th Project Meeting

Date: Mar.28th, 2022

Time: 20:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: SHUM Ka Shun

1 Discussion Items

1.1 We shared our progress on improving the performance of our model and other possible improvements.

2 Goals before the Next Meeting

2.1 We will continue to work on improving the performance of our model.

3 Meeting Adjournment

The meeting was adjourned at 20:33 p.m.

C.9 Minutes of the 9th Project Meeting

Date: Mar.31st, 2022

Time: 10:00 a.m.

Present: Professor SONG Yangqiu, CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: TU Bingying

1 Discussion Items

1.1 We shared with Professor our current progress.

1.2 Professor SONG gave us suggestions on the contract reviewing pipeline.

1.3 Professor SONG gave us suggestions on the user interface design.

1.4 Professor SONG gave us suggestions on writing the final report and preparation of the oral presentation.

2 Goals before the Next Meeting

2.1 We will modify our pipeline of contract review.

2.2 We will add triples to our knowledge graph serve as the knowledge enrichment.

2.3 We will modify our design of the user interface to make it more user-friendly.

3 Meeting Adjournment

The meeting was adjourned at 10:34 a.m.

C.10 Minutes of the 10th Project Meeting

Date: Apr.13th, 2022

Time: 22:00 p.m.

Present: CHEN Zixin, SHUM Ka Shun, TU Bingying

Recorder: CHEN Zixin

1 Discussion Items

1.1 We discussed the distribution of work of writing the final report.

1.2 We discussed the adjustments of our user interface.

1.3 We discussed the logic of the final report.

2 Goals before the Next Meeting

2.1 We will continue to write the final report.

3 Meeting Adjournment

The meeting was adjourned at 22:33 p.m.

D Appendix D

This is Appendix D.

```
def get_entities(sent):
  ent1 = ""
  ent2 = ""
  prv_tok_dep = ""    # dependency tag of previous token in the sentence
  prv_tok_text = ""   # previous token in the sentence

  prefix = ""         # prefix and modifier will hold the text that is associated with the subject or the object.
  modifier = ""
  # As and when we come across a subject or an object in the sentence, we will add this prefix to it.
  # We will do the same thing with the modifier words, such as “nice shirt”, “big house”, etc.

  nsubj_list = []
  dobj_list = []
  root_posi = 0 # the position of root verb
  source = "" # subject will be the source node in KG
  target = "" # object will be the target node in KG

  # Rule 01: We first do NER and label each entity directly with the pre-trained models in spaCy
  entities = nlp_spacy(sent).ents
  entity_list_sp = []
  for i in range(len(entities)):
    entity_list_sp.append([entities[i].text.title(),entities[i].label_])

  # Rule 02: for each token and its dependency parsing result, first check if the current token is a punctuation mark
  for idx, tok in enumerate(nlp_spacy(sent)):
    if tok.dep_ != "punct":   # if token is a punctuation mark then move on to the next token

    # Rule 03: if the current token is the root verb, record its position in the sentence
      if tok.dep_ =="ROOT":
            root_posi = idx

    # Rule 04: # if the current token is a compound word, and its previous word was also a 'compound'
    # then add them together and add them to the prefix
      if tok.dep_ == "compound":
        prefix = tok.text
        if prv_tok_dep == "compound":
          prefix = prv_tok_text + " "+ tok.text
```

Figure 27: Code for NER(part-1)

```
    # Rule 05: # if the current token is a modifier, and its previous word was a 'compound'
    # then add them together and add them to the modifier
    if tok.dep_.endswith("mod") == True:
      modifier = tok.text
      if prv_tok_dep == "compound":
        modifier = prv_tok_text + " "+ tok.text

    # Rule 06: # if the current token is a nominal subject, then prefix and modifier will
    # be added and then it will be captured as a candidate of the first entity.
    # Variables such as prefix, modifier, prv_tok_dep, and prv_tok_text will be reset
    if tok.dep_.find("subj") == True:
      if tok.dep_ != "nsubj" :
          prefix = ""
          modifier = ""
          prv_tok_dep = ""
          prv_tok_text = ""
      else: # only extract the nominal subject
          ent1 = modifier +" "+ prefix + " "+ tok.text
          prefix = ""
          modifier = ""
          prv_tok_dep = ""
          prv_tok_text = ""
          nsubj_list.append([idx,ent1])
```

Figure 28: Code for NER(part-2)

```
    # Rule 07: # if the current token is a direct object, then prefix and modifier will
    # be added and then it will be captured as a candidate of the second entity
    # Variables such as prefix, modifier, prv_tok_dep, and prv_tok_text will again be reset
    if tok.dep_.find("obj") == True:
      if tok.dep_ != "dobj" :
          prefix = ""
          modifier = ""
          prv_tok_dep = ""
          prv_tok_text = ""
      else: # only extract the direct subject
          ent2 = modifier +" "+ prefix + " "+ tok.text
          prefix = ""
          modifier = ""
          prv_tok_dep = ""
          prv_tok_text = ""
          dobj_list.append([idx,ent2])

    # update variables
    prv_tok_dep = tok.dep_
    prv_tok_text = tok.text
############################################################
```

Figure 29: Code for NER(part-3)

```
  # Rule 08: Return the subject and object that are nearest to the root,
  # if two subj/obj have the same distance to the root, keep the former one

  if len(nsubj_list) > 1: # easy sorting to find the nsubj that is the nearest to the ROOT
    subj_pos = 0
    curr_subj_min = len(nlp_spacy(sent))
    for idx, ele in enumerate(nsubj_list):
        nsubj_list[idx][0] = abs(nsubj_list[idx][0]-root_posi)
        if nsubj_list[idx][0] < curr_subj_min: # if two subj have the same distance to the root, keep the former one
            curr_subj_min = nsubj_list[idx][0]
            subj_pos = idx
    source = nsubj_list[subj_pos][1].strip()
  elif len(nsubj_list)==1:
    source = nsubj_list[0][1].strip()
# same way for object
  if len(dobj_list) > 1: # easy sorting to find the nsubj that is the nearest to the ROOT
    obj_pos = 0
    curr_obj_min = len(nlp_spacy(sent))
    for idx, ele in enumerate(dobj_list):
        dobj_list[idx][0] = abs(dobj_list[idx][0]-root_posi)
        if dobj_list[idx][0] < curr_obj_min: # if two subj have the same distance to the root, keep the former one
            curr_obj_min = dobj_list[idx][0]
            obj_pos = idx
    target = dobj_list[obj_pos][1].strip()
  elif len(dobj_list) ==1 :
    target = dobj_list[0][1].strip()

      # see if this entity has already in the labelled list

  output1 = [source,'Undefined']
  output2 = [target,'Undefined']
  for idx,entity in enumerate(entity_list_sp):
    if source == entity_list_sp[idx][0]:
        output1 = [source,entity_list_sp[idx][1]]
    if target == entity_list_sp[idx][0]:
        output2 = [target,entity_list_sp[idx][1]]

  return [output1, output2]
```

Figure 30: Code for NER(part-4)