KU LEUVEN

FACULTY OF SCIENCES

Knowledge Graph Construction of the Financial Regulations of the European Union

Ioannis Dikaioulias
Konstantinos Panagiotis Apostolou

Supervisor:
Prof. dr. Jan De Spiegeleer

Master thesis submitted in fulfilment
of the requirements for the degree in
Master of Science in Statistics and Data Science

Academic year 2024-2025

Preface

First and foremost, we would like to thank our supervisor Prof. dr. Jan De Spiegeleer for providing us with the initial inspiration for this assignment.

Working on this project taught us many new skills, from creating pipelines and processing data from a large database to applying various statistical and deep-learning models. This experience has significantly shaped our interest in the field, and we look forward to having the opportunity to implement these methods in our business and academic careers.

Ioannis Dikaioulias: I would like to thank my family and friends for supporting me in this endeavor. Moreover, I would to like to thank my partner, Konstantinos Panagiotis Apostolou, for his contribution.

Konstantinos Panagiotis Apostolou: I would like to express my deepest gratitude to the Eugenides Foundation for their generous support and unwavering belief in my academic journey. The scholarship awarded to me during my studies not only eased my financial burdens but also served as a constant source of motivation to strive for excellence. Their commitment to fostering education and empowering students has made a significant impact on my life, and for that, I am eternally grateful. Also, I would like to thank my partner Ioannis for his relentless work, for without him, this Thesis would be incomplete.

Abstract

Introduction: Knowledge graph construction is a multifaceted task that encapsulates many natural language processing techniques. It is closely related to the subfield of information extraction, where there is the distinction between methods that assume an open or closed domain. In this work, it is explored how rule-based, statistical, and deep learning techniques can be used to create or facilitate the process of building a knowledge graph. Also, the arrival of large language models that offer exceptional performance in language generation tasks has revolutionized the way NLP is done; therefore, we investigate their potential application to our situation and describe their supplementary connection with knowledge graphs.
Methodology: Initially, the raw text was preprocessed to exclude noise. Then the text was passed through the following pipeline: first, coreference resolution was used to match expressions referring to the same entity, then the ClausIE model was applied, and SVO triplets were extracted. We filtered and processed these triplets to produce the final head-relation-tail triplets, which formed the preliminary data graph that served as the foundation for our knowledge graph. They were used in three different ways, where each of the disciplines of knowledge graph creation were tested.
As a first step, we explored the graph manually, aiming to define a schema to attach in order to create our own knowledge base. We derived use case questions that allowed us to define a schema and also mapped our graph to a predefined ontology, which augmented the graph with outside information about its entities. Secondly, we utilized graph algorithms, specifically centrality analysis and community mining methods, to explore the data graph. These methods allowed us to narrow down the original graph to its core nodes, which made it much more manageable for extracting latent structures and formulating a possible use case. Finally, we explored the application of a deep learning model, REBEL.
Results: We constructed two knowledge graphs, each using a different technique to illustrate the advantages and disadvantages of either the rule-based or deep learning methods. Also, we saw two ways to uncover possible use cases for a potential knowledge graph, which is a critical step to define the knowledge base of a knowledge graph. In addition, we compared the ClausIE model with REBEL and showed that the latter achieves satisfactory performance without any tuning. However, what was evident in every approach was the importance of a domain expert, who would either define the schema or provide a labeled test set to tune and evaluate the models.
Keywords: Information Extraction, Knowledge Graph, ClausIE, Graph Analytics, REBEL, LLM, Ontology

List of Figures

List of Tables

Contents

1 Introduction

Ioannis Dikaioulias & Konstantinos Apostolou

Data Science is all about numbers. But, is that statement really true? When a person thinks of data, numbers are the first image in that persons mind. Numeric Data, ranging from streams of revenue on a firms' accounting statement, to fluctuations of a stock's price in the stock market, or even to evaluation metrics of a famous athlete. These data can be stored from the simple format of a simple spreadsheet, in rows and columns, to a more sophisticated relational database in the cloud, such as Amazon Aurora. These numeric data are mostly dominated by a single characteristic. They are structured. Most numeric data available for use in the world, are stored in an efficient, organized way, so that humans and most importantly computers, can easily identify, access, process and utilize.

Alas, there is another form of data, which dominates all other, and has been available since humanity first started to use written language. These are text data. Text data, can be found everywhere in the world and in abundance. Text data can range from large history books, to news articles, business reports, emails, and even to the vast every day messages in social media. They are a challenge for data scientists to handle, not only because of their existence in huge numbers, but because of the main characteristic, that separates them from numeric data, which is their unstructured format. In their natural form, text could not be stored in a spreadsheet, to be directly processed in a computational way. Text data is difficult to be processed by machines, compared to structured numeric data. This is is a challenge for data scientists who seek to extract valuable information from data, using computational methods. Text in its natural form, is characterized by the complexity of the human language, which is a challenge for machines to interpret and detect patterns. While the human eye, can easily understand the semantic information hidden in the text, this is not the case for machines. Nevertheless, the advancements in Artificial Intelligence (AI), especially with the advent of Transformers, (41), has changed how text is handled, providing machines with the ability to grasp the context hidden in human text, thus revolutionizing the field of text processing, otherwise known as Natural Language Processing (NLP).

One type of the vast amount of unstructured text data, are legal texts, such as regulations, contracts, legislation, legal codes and agreements. One of these legal texts, are the Financial Regulations of the European Union (15). More specifically, the Financial Regulations document, adopted in July 2018, as initiated by the European Commission in 2016, is applicable on the general budget of the European Union and concerns the management of EU Funds. The Regulations are broken down into 16 main chapters, or Titles as called in the document. These Titles dwell on the following subjects:

- **Title I:** Subject Matter, Definitions and General Principles
- **Title II:** Budget and Budgetary Principles
- **Title III:** Establishment and Structure of the Budget
- **Title IV:** Budget Implementation
- **Title V:** Common Rules
- **Title VI:** Indirect Management
- **Title VII:** Procurement and Concessions
- **Title VIII:** Grants
- **Title IX:** Prizes
- **Title X:** Financial Instruments, Budgetary Guarantees and Financial Assistance
- **Title XI:** Contributions to European Political Parties
- **Title XII:** Other Budget Implementation Instruments
- **Title XIII:** Annual Accounts and Other Financial Reporting
- **Title XIV:** External Audit and Discharge
- **Title XV:** Administrative Appropriations
- **Title XVI:** Information Requests and Delegated Acts

The EU's Financial Regulations is a complex document comprising of legal articles. These articles correlate to one another, not only within each Title, but in several instances, across Titles. For any interested party, ranging from either a researcher or a EU Official, the study, analysis and understanding of the Financial Regulations can be a complex and time consuming task. It would be inefficient for the interested party, both in terms of time management and received information, to search through each title, article and paragraph, to find the required information, but also to manually search for other articles across title, that contain important relevant information, to the initial search question. A solution should be provided, that transforms the financial regulations text, into a structured format, one that would offer the interested party with the tools to efficiently search for the requested information, understand the structure of the regulations, connect all related information together and thus answer the initial search question, in a way that is efficient in terms of time required and information received.

One of the most innovative ways to convert the unstructured text of the financial regulations, into an easily accessible, structured format, is through the use of a graph database management system (GDBS). A graph database, in contrast to a relational database, stores data in a dynamic way, through the use of nodes and edges. A graph database such as the Neo4j Graph Database System (30), not only stores data efficiently, but also visualizes the data in a friendly way to the human eye. A graph database, is based on graph transversal to obtain information and has its origin in graph theory, which was first initialized as a mathematical problem, namely the Königsberg bridge problem (2). A way to store unstructured text data in a graph database, is the knowledge graph.

Main Research Question: This Thesis aims, in simple words, to construct a knowledge graph from the unstructured text of the EU's Finacial Regulations and store the graph in a graph database. The following sub-research questions are directly related to the main research question:

1. What methods exist in the literature for knowledge graph construction?
2. Can the knowledge graph be constructed in a clearly automated way, without human intervention?
3. Can domain related use cases questions of a knowledge graph be drawn in an automated way? How important is the role of the domain expert in knowledge graph construction?
4. How has Artificial Intelligence (AI), changed the way knowledge graphs can be created?

Structure: The sections of this Thesis are structured as such: In section 2, certain important terms regarding knowledge graph construction are defined, such as what is a knowledge graph. Then in section 3, different methodologies in the literature for knowledge graph construction are discussed. The discussion is followed by a description in section 4 of the main semi-automated, NLP based, methodology, used to construct a knowledge graph and how it was inspired. The methodology's implementation is analytically described in section 5. Section 6, is devoted on how graph analytics can contribute on knowledge graph construction, and specifically on defining its purpose. In section 7, an automated method, in knowledge graph construction, based on Transformers, is explored and compared with the semi-automated method used in section 5. In section 8, a discussion is conducted on the role of Large-Language-Models (LLMs) in knowledge graph construction. Finally the conclusion of the Thesis, is included in section 9.

Results: This thesis includes two main results. The two results, are the product of the implementation methods used in this thesis, in sections 5 and 7. The two results consist of two knowledge graphs, which describe the semantic information hidden in Title III of the financial regulations. The knowledge graph as shown in 5.7, is considered a good representation of how a financial regulations knowledge graph should resemble, according to the work of this thesis, while the knowledge graph in 7.3 would ideally require additional processing. The results concern only Title III but the same pipeline can ideally be implemented across the Titles, to construct knowledge graphs, that would then, theoretically be merged into an overall knowledge graph of the entire financial regulations.

2 Terminology

Ioannis Dikaioulias

The following terms are important for the reader to understand in order to grasp the research and methodology surrounding knowledge graph construction, as is described in the subsequent chapters.

2.1 Knowledge Graph

In the knowledge graph construction book by (Barrasa and Webber), a knowledge graph is defined as a type of graph, a structure of nodes (vertices) connected by relationships (or edges) to create high-fidelity models of a domain. As referred to in (18), a knowledge graph is defined as such:

> "A graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities."

The knowledge graph's structure is based on a graph-based data model. The most commonly found and simpler structure is the property graph. The property graph is a subclass of the knowledge graph. The building blocks of a property graph, are the nodes and the relationships that interconnect those nodes, as well as the characteristics that those nodes and edges can have. According to Knowledge Graph Construction Book (Barrasa and Webber), nodes (or entities) in the property graph belong to a label, also known as class or category, and can have zero or more properties that represent a given nodes' characteristics. Moreover, relationships can have a type as well as zero or more properties.

An example property graph is provided below:

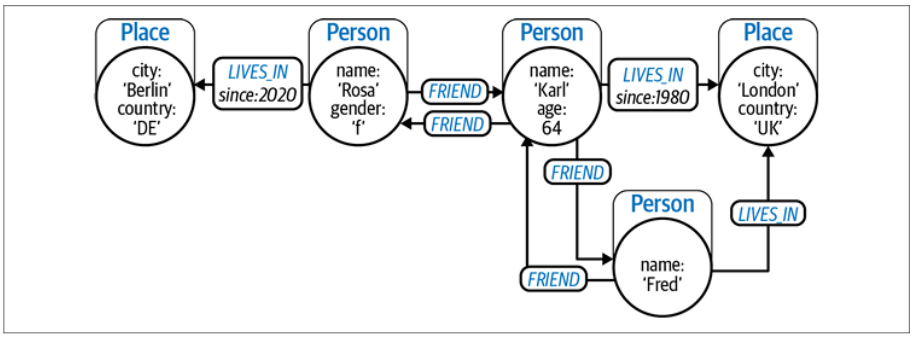


Figure 2.1: Property Graph Example
(Barrasa and Webber)

For the reader to fully grasp a knowledge graph power, the term organizing principle needs to be understood. The organizing principle of a knowledge graph is basically a "contract" that defines how the nodes and edges of a knowledge graph interact with one another. It is a set of rules, that define the elements of the graph and their characteristics. For example, in the case of the property graph the user expects to find labeled nodes and directed relations with or without properties.

2.2 Ontology

A more complex form of an organizing principle, is a taxonomy that offers hierarchy among different node labels, offering broader and narrower structure among labels in the graph. A taxonomy is expressed in the knowledge graph in the form of classes and their subsequent subclasses. A property graph with multiple hierarchies, and a taxonomy organizing principle is given in the following example:

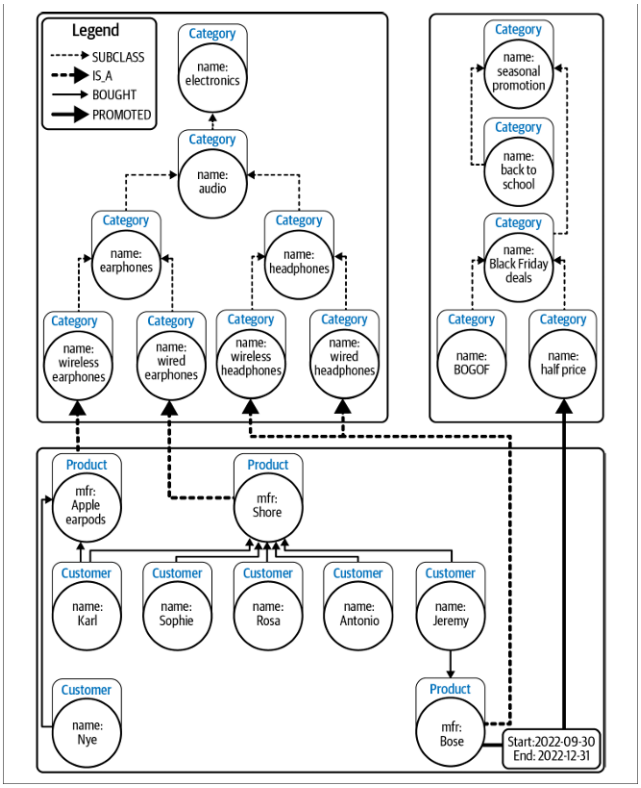


Figure 2.2: Taxonomy - Multiple Hierarchies
(Barrasa and Webber)

The SUBCLASS relation defines the hierarchy among categories. Instances of the CUSTOMER class (category) buy a product which is a subclass of the wired earphones category.The main characteristic of a taxonomy is that the hierarchy among categories, consists of "vertical" relations, of the broader-narrower = class-subclass nature. A taxonomy is a subclass of a more sophisticated organizing principle, the ontology.

An ontology allows for more complex relations among categories and thus among the instances of the categories. More specifically, an ontology allows not only "vertical" but also "horizontal" relations among categories Knowledge Graph Construction Book (Barrasa and Webber).

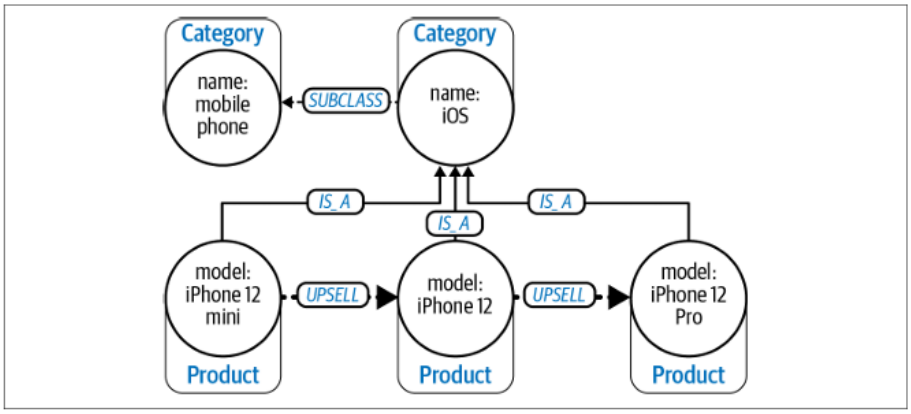


Figure 2.3: Ontology - Horizontal Relation
(Barrasa and Webber)

It is observed in the graph all three instances of i-phones belonging to the model category, are subclass (Vertical Relation) of the iOS category. What the ontology organizing principle offers are more dynamic relations such as the UPSELL relation type. The UPSELL relation type offers a "horizontal" relationship among iphone instances. The semantic information that is conveyed is that iPhone 12 Pro is superior than the rest in terms of sales. This horizontal semantic information can not be conveyed by a simpler taxonomy organizing principle.

An ontology is defined by (18) as:

> "A concrete, formal representation-a convention-on what terms mean within the scope in which they are used (e.g. a given domain)."

An domain specific knowledge graph or as referred to by (18), an enterprise knowledge graph, requires an domain specific organizing principle, thus a domain specific ontology, in order to describe the complex relations and structure of the nodes and relationships found in the domain.

The ontology of a knowledge graph can be constructed or re-used from a source on the web, as described in the seven step method for ontology learning (Noy and McGuinness). For specific domains, there are ontologies that serve as a knowledge base, and are ready to be reused. Two such ontologies, are FIBO (FIBO) and Schema.org (Schema.org).

In conclusion, an ontology of a specific domain describes which categories of entities are required, which relations are acceptable and how these entities and relations relate to each other. Thus, an ontology provides the rules that define a semantic network of information, such as a knowledge graph.

2.3 Information Extraction (IE)

In the context of knowledge graph construction from unstructured text, information extraction (IE), includes two main tasks which are Named Entity Recognition (NER) and Relation Extraction (RE). The tasks are essential to the knowledge graph construction. A knowledge graph by definition, is comprised of Nodes and Edges. NER is the task that leads to the construction of nodes, while RE is the task that leads to the construction of edges. The entities extracted from NER, will act as the corresponding nodes, while the relations extracted from RE, will act as the corresponding relations. Together, the extracted entities and relations, form the so called triplets. The main goal of IE, is to create those triplets. These triplets capture the semantic information hidden in the data, and comprise the bricks that form the knowledge graph. A knowledge graph is a combination of triplets, connected together to form a semantic network of interconnected nodes. These triplets are in the form of Head-Relation-Tail. A triplet can be called a SVO triplet (Subject-Verb-Object) or a

SPO triplet (Subject-Predicate-Object), or even called a Head-Relation-Tail, which more accurately captures the main usage of a triplet in the context of a knowledge graph.

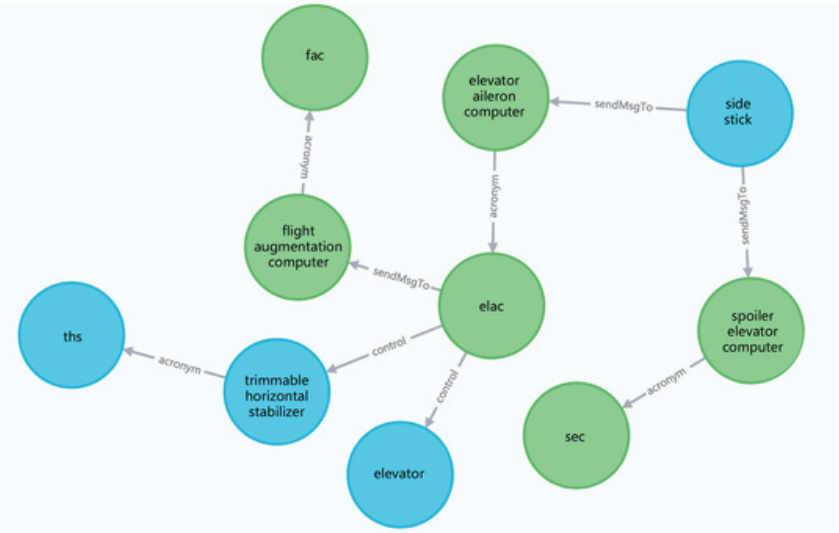


Figure 2.4: Partial Knowledge Graph Example (26)

In the above graph, the following "Head-Relation-Tail" triplets can be observed:

Head	**Relation**	**Tail**
elac	control	elevator
elac	sendMsgTo	flight augmentation computer
side stick	sendMsgTo	spoiler elevator computer

3 Literature Review

Ioannis Dikaioulias

3.1 Information Extraction From Unstructured Text

Knowledge Graph construction in the literature revolves around a certain task, which is information extraction (IE) 2.3. Independent of how IE will be performed, in all the researched papers, the authors faced either one or both of the following obstacles:

1. The absence of a predefined domain specific ontology to match their use case and to define the schema of the knowledge graph
2. The absence of a predefined domain specific Entity-Relation labeled dataset to be used for supervised or transfer learning.

Information Extraction (IE) can be further broken down into two different types. The first type is closed IE, and the second type is open IE. Open IE, from a statistical point of view, is purely unsupervised. It includes the use of open information extraction tools for NER and/or RE such as in the work of (39) or the use of rule-based methods for IE, such as in the works of (4) and (22). As the above methods are not based on a specific domain, the entities and relations extracted, are in most cases redundant/abstract and in huge numbers. This is a problem when the goal of IE is to extract entities and relations that can help answer specific domain use case questions. A knowledge graph created out of such abstract information, would not only be unfriendly to the human eye in terms of visualization, but would also not provide any meaningful insights. The solution to the vast amount of extracted information has a name. That name is ontology!

The second type of IE is closed IE. This form of IE, includes entities and relations extraction, not in an abstract way, but in a domain specific way. It involves the use of supervised or transfer learning, or a combination of both. Supervised learning can be harnessed for IE by training a model that learns to predict domain specific entities and relations such as in (46) and (die). On the other hand, transfer learning, includes the use of a pre-trained model, which is fine-tuned to extract domain specific information. In terms of supervised IE, the chosen models can either be more statistical such as in (48) where a simply Conditional Random Fields-CRF model, (40) is utilized for NER, or they can be based on deep learning architectures such as in (46) where a deep learning architecture that includes a multilayered architecture is utilized to perform NER. Evidently, closed IE, does not just require an ontology framework, but a labeled entity-relation dataset, split into training, validation and test set, to be used for training and evaluation of a model that will learn to predict domain-specific entities and relations. The role of an ontology in closed IE is not to filter out redundant entities and relations, but rather to help create or use a labeled entity-relation training and test set that conforms to the ontology.

3.2 Ontology

An ontology is required to limit the abundance of entities and relations and contain the information to the domain (use case) questions that need answering. An ontology in the literature can be created using two main approaches. The authors of (46) utilized the bottom-up approach while the authors of (die), utilized the top-down approach. In the work of (26), different kinds of ontology are described and explained. One of the main papers in the literature, that describes a popular method for ontology construction and is the basis for ontology learning in almost all searched papers, is (Noy and McGuinness). In this paper, the structure, creation and usage of an ontology is described, and is a great tool for understanding important ontology concepts. In conclusion, regarding open IE, ontology is necessary to contain the information and achieve domain-specific results.

It is important to underline another important role of an ontology. Apart from providing the schema that maps the entities and relations to a specific framework, an ontology acts as a knowledge base, that defines the textual meaning of a domain specific entity and helps to disambiguate similar entities found in the text, depending on their meaning. In certain papers such as in (39), an additional task is involved in IE. That is the task of Entity Linking (EL) AND Entity Disambiguation. During these two tasks, an entity is linked to each textual meaning in the knowledge base. An ontology that also serves as a knowledge base, is that of FIBO (FIBO).

3.3 Closed Information Extraction - closedIE

In the literature, domain-specific NER, in the context of domain-specific IE (closedIE) is performed using supervised learning. In supervised learning, a model is trained on a training set, tuned on a validation set and evaluated on a test set, that represents the ground truth, and showcases the ability of the model to generalize efficiently when applied on new data. In older studies, a statistical model used in the literature for NER, is the CRF (40). Deep learning models, that harness the power of recurrent neural networks (RNNs), such a LSTM and more specifically a BiLSTM, improved the performance of NER. In (die), the authors use a combination of BiLSTM-CRF to achieve more accurate results in NER. The advent of transformers in 2017, (41), changed the field of NER. More specifically, the BERT Large-Language-Model (LLM) (13), which is based on the Tranformers (41) architecture, manages to capture contextual information in the text through the use of contextual word embeddings. These embeddings transform text into dense numerical representation in the form of vectors, that capture the semantic information of each token (word) in a sentence. The deep learning architecture found common in the literature includes a BERT-BiLSTM-CRF architecture where a pre-trained BERT LLM, produces embedding vectors out of sentences, that pass on to a BiLSTM neural network and then to a statistical CRF model. The architecture of a BERT-BiLSTM-CRF deep learning model is given below:

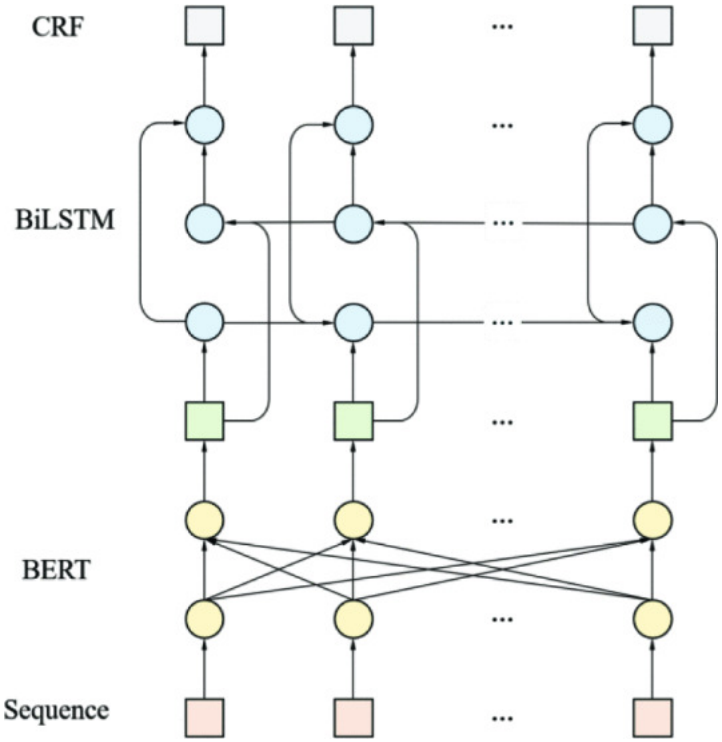


Figure 3.1: BERT-BiLSTM-CRF Architecture
(46)

To capture the power of pre-trained BERT embeddings, the authors of (die), first conducted NER by training a BiLSTM-CRF architecture and then added a BERT layer, in order to achieve higher results on a test set. The difference in performance can be observed in the following table:

Table 3.1: BiLSTM-CRF VS BERT-BiLSTM-CRF

Models	Accuracy	Precision	Recall	F1
BiLSTM-CRF	98.73%	89.36%	93.18%	91.23%
BERT-BiLSTM-CRF	**99.07%**	**92.96%**	**94.81%**	**93.88%**

In a study regarding building a knowledge graph out of flight control maintenance manuals, to help pilots and engineers, (26), utilized a transformer encoder with a final CRF layer. The model architecture used in their study is the following:

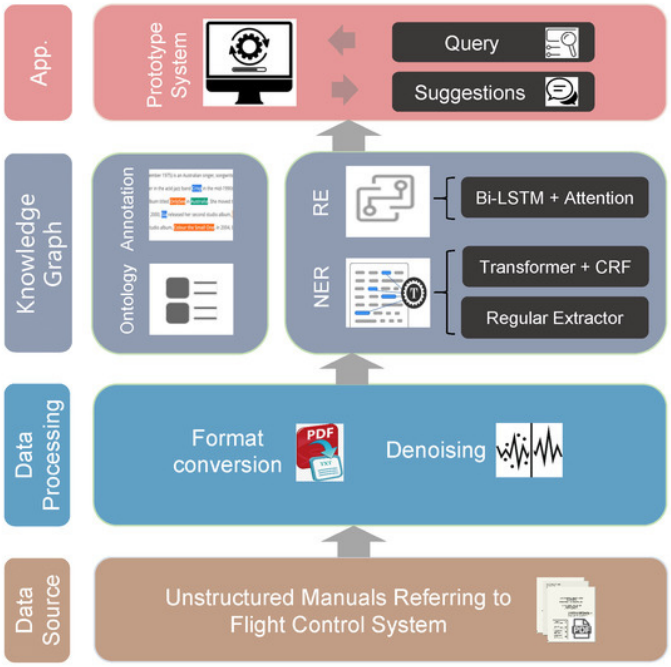


Figure 3.2: Knowledge Graph Construction Pipeline (26)

In (26) the authors transformed the tokens of a given sentence using numerical vectors. Different kinds of embeddings were calculated, such as the embedding of the Part-Of-Speech tag of each token, the embedding of whether the token has high frequency in the text or not, or the embedding of whether the token is mapped to a domain-specific phrase or keyword. Different combinations of these embeddings are combined to form a final embedding vector, which is then inputted into a transformer encoder and then to a CRF layer. The authors concluded that higher performance is achieved, when the final embedding vector, is the result of the highest possible combination of different embedding vectors. The performance of their model on a test set, can be observed in the following table:

Table 3.2: Transformer Encoder - CRF

Model	Precision	Recall	F1
Tranformer Encoder-CRF	84%	80%	82%

It is evident that transfer learning, through the use of BERT pre-trained embeddings, leads to better results compared to simply using the raw text as data. The reason is that embeddings, are able to capture semantic information of a text sequence and pass it to the forward layers in the form of numerical embedding vectors.

When Relation Extraction (RE) is considered, again transformers play a crucial role in achieving higher performance. In all papers of closedIE, RE, is performed as a different task from NER. Either entities extracted from NER as used as the input to the RE model, or ground truth labeled entities are directly used to train a model. In (26) a simpler architecture of a BiLSTM + Attention Mechanism, is trained on ground truth labelled entities and achieves good performance as indicated in the following table:

Table 3.3: BiLSTM + Attention Mechanism

Model	Precision	Recall	F1
BiLSTM + Attention Mechanism	86%	77%	81%

In (43), introduced a pipeline called FR-BERT for relation extraction out of financial related text:

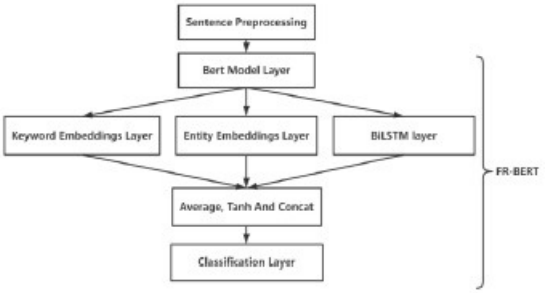


Figure 3.3: FR-BERT Architecture
(43)

The basis of this pipeline is to use pre-trained embeddings which focus on finance related entities and keywords. Thus, the basic BERT model is fine-tuned to capture finance domain entities. The authors of this paper, compared the FR-BERT, R-BERT and BiLSTM+Attention on the same test set. The results are observed in the following table:

Table 3.4: BiLSTM VS R-BERT VS FR-BERT

Models	Precision	Recall	F1
BiLSTM + Attention Mechanism	91.60%	90.59%	90.94%
R-BERT	94.15%	93.65%	93.35%
FR-BERT	**94.79**%	**95.42**%	**95.02**%

3.4 Open Information Extration - openIE

In the literature, two ways to perform open information extraction are discovered. The openIE methods for knowledege construction are summarized below:

1. Open Information Extraction Tools such as REBEL and openNRE (reference) and ReliK (reference) that extract triplets from text by linking and disambiguating entities to open domain knowledge bases such as Wikipedia.

2. Rule Based Methods that rely on text mining techniques and NLP methods, in order to extract information from text in the form of Subject-Verb-Object (SVO) triplets. This method is unsupervised as it relies on the text data alone to extract information.

In regard to the **first method**, pretrained models such REBEL (20), rely on specific pipelines that are implemented to perform IE. A pipeline for knowledge graph construction using open IE tools, (39) is presented below:

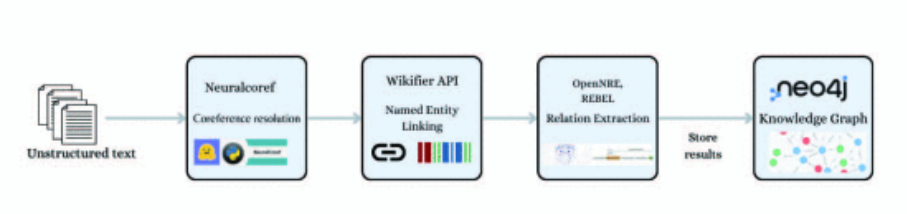


Figure 3.4: OpenIE Pipeline
(39)

From the pipeline, the openIE character is visible from the Wikifier API, were Entity Linking (EL) is performed to link entities to Wikipedia instead of a domain related knowledge base, such as FIBO (FIBO). In the last step, the extracted triplets are stored in the Neo4j Graph Database System (30). NER and RE are performed in the same pipeline by either REBEL or openNRE.

Another pretrained pipeline found in the literature is called Relik, is an innovative and quick way to perform IE as it performs both NER and RE in one forward pass of the pipeline,(34). ReliK utilizes the power of Large Language Models (LLMs), and specifically a transformers encoder. The pipeline of Relik is summarized in the following visual:

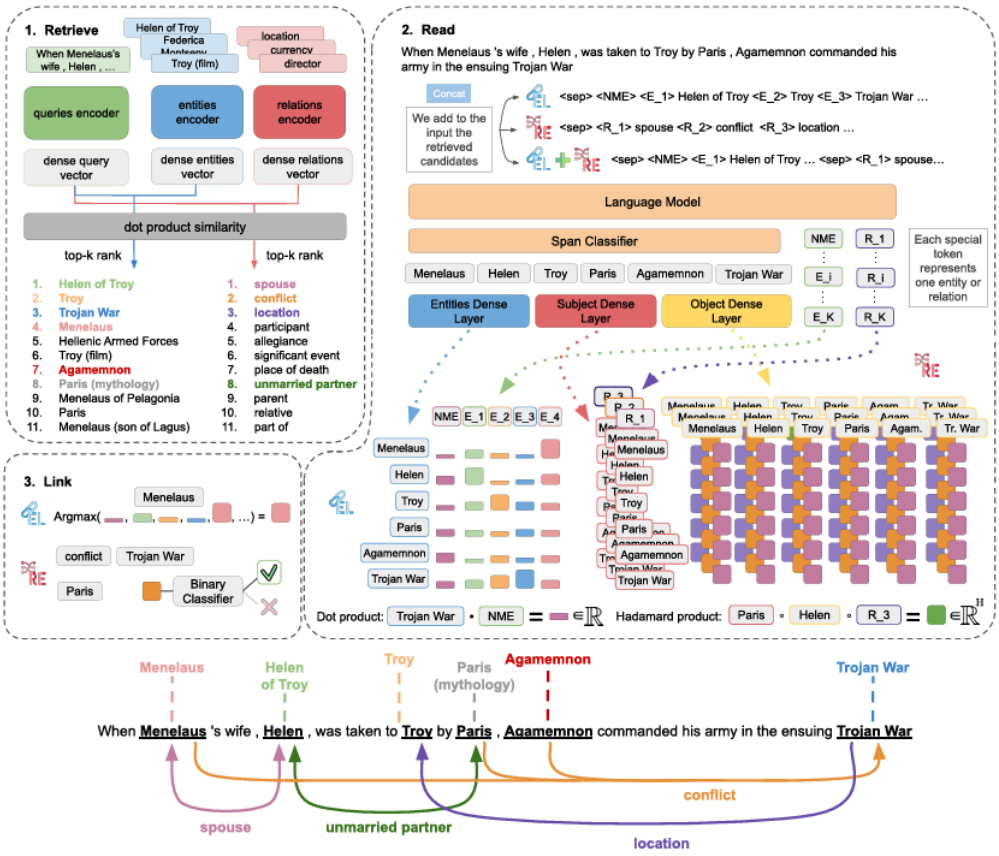


Figure 3.5: ReliK Information Extraction (34)

In regard to the **second method**, raw text data is utilized directly, in an unsupervised way, to extract information through text mining techniques, by identifying patterns in the text data or use rule-based algorithms to extract information in the form of triplets. In (4) the authors identify and extract patterns out of text to discover entities belonging to a specific class in the ontology. To extract the entities, regular expressions are used. A simpler but at the same time efficient rule based method for information extractions is proposed in (4). The authors make us of NLP methods, namely dependency parsing and pos tagging, to extract Subject-Verb-Triplets from each sentence in the text. The triplets are then visualized in a graph, called preliminary.

Figure 3.6: Preliminary Graph
(4)

The preliminary graph or graphs are comprised of a huge number of generic-open domain entities and relations, that are too redundant to be used directly for a knowledge graph, but require preprocessing to limit their number and only keep the most essential. Domain expertise is required to better understand the concepts, and contextual information hidden in the preliminary graph and thus define the domain related use cases. An ontology based schema is then created by the domain expert. The ontology can be based on various ontology learning techniques. In (26), the cyclical method, the skeletal method and the seven-step method (Noy and McGuinness), are mentioned. The usage of a preliminary graph, means that a bottom-up approach is used to create the ontology, as entities and relations are first extracted and then used to create the ontology. The final schema, defines the main categories (classes) that the entities belong and their properties, as well as the types, and properties of the main relationships.

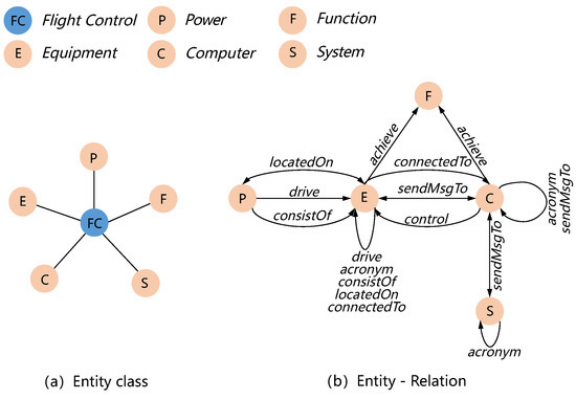


Figure 3.7: Ontology Based Schema (26)

The above graph is an example of an ontology-based schema that helps to answer specific domain questions, on flight control maintenance.

3.5 ClosedIE VS OpenIE

The main difference between closedIE and openIE is related to the ontology. In openIE, there is no clear ontology required from the start. Entities and Relations are extracted in a generic way, often linked to entities of a more generic knowledge base such as Wikipedia (34) and (39). Domain expert(s) are then used to construct an ontology in order to keep only the relevant entities and relations. Some of the openIE tools used in the literature such as ReliK and REBEL, can be fine-tuned to perform Entity Linking (EL) on a domain specific knowledge base, thus no manual filtering is required. On the other hand, when openIE is performed using purely rule-based method, the entities and relations extracted are too generic and redundant and thus require filtering. The disadvantage of openIE is that the evaluation is more difficult. There are two ways to evaluate an open Information Extraction. The first way is to create a ground truth test set of entities or relations, as in (39), then count how many of the extracted entities/relations belong in the pool of ground truth entities/relations. This can be achieved by a simple accuracy metric (see Appendix A). The second way is to simply ask the correct questions and thus evaluate the practicality of the extracted information. This can be achieved by evaluating how good the knowledge graph can answer questions related to the domain. These questions refer the use case questions that the knowledge graph is meant to answer.

Conclusively, the main advantage of closedIE is that the performance in terms of information extraction is more measurable than openIE. In closedIE, a model is trained or fine-tuned, to extract domain-specific information in the form of entities and relations. Since there is a clearly defined test set, IE can be numerically measured. In the literature the evaluation metrics of Accuracy, Precision, Recall and F1 (see Appendix A) are used to measure the performance of the IE model used. In constrast, openIE performance, is mostly measured on the practicality that the information extracted offers, in terms of the created knowledge graph's ability to answer domain specific questions. These questions are answered in the form of knowledge graph queries. These queries depend on the graph database system used to store the knowledge graph. A popular graph database system used in the literature is the Neo4j Graph Database System.

3.6 Text Preprocessing in the Literature

In the literature one of the most important step before triplet extraction, is to preprocess the text data. In most papers, text data are in the form of PDF documents. An example of raw data in PDF format, is given in (26), where the raw data are flight control maintenance manuals.

The PDF format is difficult to manage directly. Thus, **parsing** is required to transform the data hidden in the PDF, into raw text data. In (26) the authors clearly state that they used a parsing tool, namely PDFMiner (PDFMiner.six), to extract the raw text from documents.

These documents contain information that is not needed for information extraction. This noisy information includes page marks, page numbering, company details, emails, names and any other information which repeats in every page and is usually there for formal reasons, not offering any actual information. An example is given below:

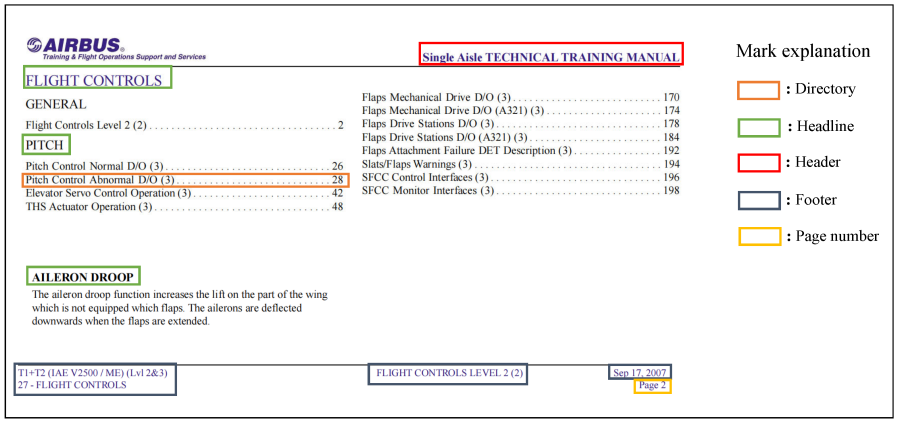


Figure 3.8: Denoising of Text Data Source (26)

The process of removing such data is called **denoising**, and involves the use of text mining methods, such as regular expressions (5), that help detect and remove noise from data.

Another important process found in papers such as (26) and (46), is **labeling**. Labeling is done manually in the literature and is related with closedIE. In (4) the authors utilize LabelStudio, which is a tool for created labeled datasets out of text. A labeling example is given below:

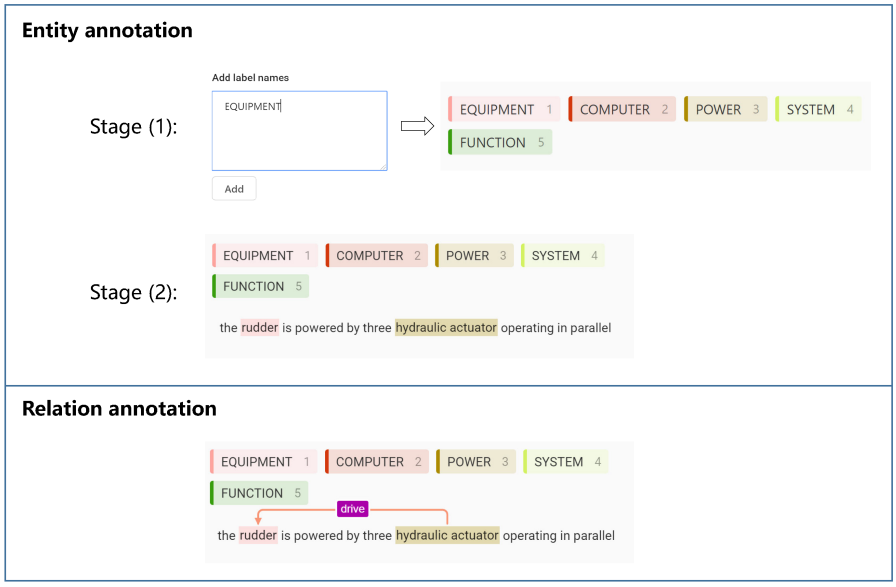


Figure 3.9: Entity-Relation Dataset Labeling
(26)

Entities and relations are manually labeled. These datasets are then used for prediction of domain specific entities and relations, in new text data.

A crucial step found is coreference resolution. This step involves replacing pronouns in text, with the subject or object that the pronouns refer to. Coreference resolution ensures that as much information as possible is preserved. If pronouns are not replaces with their references in the text, information is lost, since a pronouns such as "she" or "it", do not offer any contextual information on their own. An example of coreference resolution from (39) is given below:

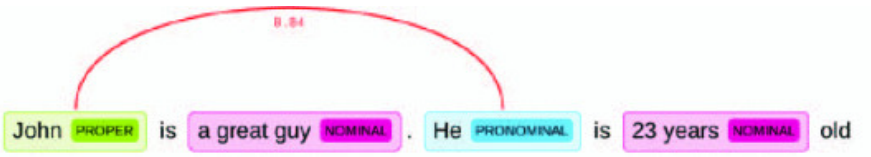


Figure 3.10: Coreference Resolution Example
(39)

3.7 Importance of the Domain Expert

In all the research papers, either the IE is closed or open, the domain expert(s), plays a crucial role in knowledge graph construction. Depending on the domain that the knowledge graph belongs, the intervention of an expert is required to create the ontology. Without the domain expert, no domain specific ontology-schema can be created to keep only the most important information from the text. At the same time, without domain expertise present, no labeled entity-relation dataset can be created to be used for supervised learning (model training) or transfer learning (fine-tuning a pre-trained model). Even when an ontology is not created, but an already constructed ontology is reused, the domain expert is required, to correctly identify the domain and knowledge base. In conclusion, the domain expert's knowledge is essential in ontology learning independent of the IE methodology used.

3.8 Conclusion

In the literature there seems to be a distinction between domain-specific IE (closedIE) and (openIE). ClosedIE appears to be superior, in the sense that the results of information extraction are more easily measurable. On the other hand, openIE methods, are more easily applied and require less time and effort, while at the same time, being able to achieve practical results, that are able to answer the use case questions at hand. In relation to the research question in this thesis, the problem of ontology absence is present. Although FIBO can be used as an knowledge base for Entity Linking (EL), the framework and mapping of FIBO is not enough to describe the complex relations present in the financial regulations. Ontology learning is thus required. As observed in the researched papers, ontology creations required human intervention through domain expertise. In the construction of a knowledge graph from financial regulations' unstructured text, there is no domain expert present. The domain expert in this case could be an individual, with expertise in the domain of financial regulations, with strong knowledge of the legal framework as well as knowledge of European Union's processes. To solve the absence of ontology and domain expert, a methodology was constructed, inspired by papers on open information extraction tools and on NLP rule-based methods for IE.

4 Main Methodology: Description

Ioannis Dikaioulias

4.1 Problem

A domain specific knowledge or as referred to in (18), an enterprise knowledge graph, requires a domain specific-ontology. As mentioned in the previous chapter, one of the two main obstacles of knowledge construction, is the absence of a clearly defined domain-specific ontology. In the case of knowledge graph construction from financial regulations, there is no predefined ontology that describes the structure of the regulations. The financial regulations domain contains a complex structure of legal and financial terms that connect with each other. There is no single ontology that can describe the complex financial regulations' structure. This obstacle is the main issue that needs to be overcome. To manually construct an ontology from scratch is a complex task that requires not only a significant amount of time and effort, but also the most important element, which is domain expertise. Without in-domain knowledge of the financial regulations, an ontology is almost impossible to be accurately constructed.

4.2 Solution-Inspiration

In all researched papers, the absence of ontology was the main problem faced, in regard to knowledge graph construction. In the case of knowledge construction from financial regulations, the same problem exists. To overcome the issue, inspiration was drawn from (4). In this paper the authors utilize an openIE methodology, where text mining methods are used to extract triplets from text, in an unsupervised way. More specifically, the authors make use of the so-called preliminary graph. The preliminary graph is constructed from triplets extracted from text. These triplets do not abide to a specific domain, but are rather abstract and generic. These triplets are the result of a rule-based Information Extraction. NER is performed to extract the heads and tails of the triplets and RE is performed to extract the relations. The authors use the following text mining methodology to extract the triplets.

- A sample text is chosen from a the collection of raw text data available. In the case of this paper, the raw text data is in the form of manuals. The text is segmented into sentences. A triplet is extracted from each sentence. The triplets are in the form of Subject-Verb-Object (SVO).
- Part-of-Speech Tagging (POS) (5) and Dependency Parsing (DP) ((5) are applied on the text. Tokens (words) that have the tags NOUNS or PROPER NOUNS are used as the Subject (Head) and Object (Tail), while tokens with VERB tags are used as the Verb (Tail) in the triplets. Dependencies are used to locate Subjects and Objects that are comprised of more than one token (phrases). For example, in the sentence "Students will write their own IDS rules", although the subject is the token "Students". the object, is not a single token, but the phrase "own IDS rules". In this example, the tokens "own" and "IDS" are dependencies and more precisely, an adjectival modifier (amod) and a compound, respectively.
- The extracted SVO triplets are then filtered to deal with redundancy of the entities and relations. Entities are manually disambiguated. Similar entities are merged to a single one, while entities that are not linked to the domain at all, are removed. Lemmatization is applied on the entities to remove redudant entities, while only the most frequent relations are not removed.
- The filtered triplets are the visualized in a graph where the subject and object of a triplet serve as the node and tail, and the verb serves as the relation. The graph is called preliminary and is utilized by the domain expert, in order to better understand the domain and build an ontology-based schema. An ontology is not constructed fully, but rather a schema is visualized. The schema is based on the most important categories and how they relate, as observed in the preliminary graph.

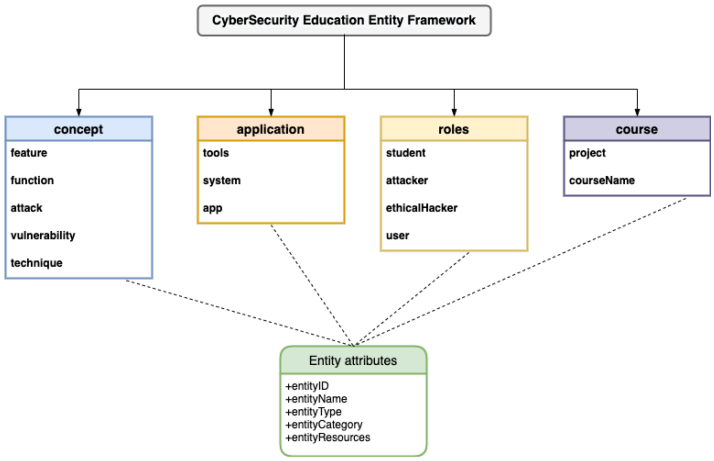


Figure 4.1: Main Categories and Entities Distinction
(4)

- The domain expert intervenes to manually assign entities and relations into categories. The filtered triplets are then used to create the knowledge graph, which is stored and visualized in the Neo4j Graph Database System.

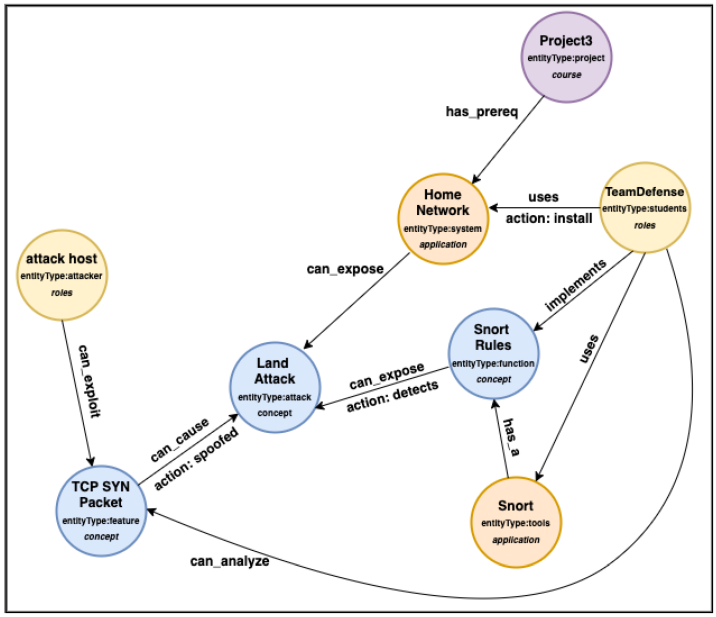


Figure 4.2: Cybersecurity Knowledge Graph
(4)

It is important to note, that the preliminary graphs created in (4), are not for the entire text data available. The authors created the preliminary graphs of a sample of text data, and then created textual rules, through spaCy's Entity Matcher (19), in order to identify in the rest of the text, entities and relations that abide with the ontology-schema derived from the sample text. The ability to generalize is then crucial to constructing a knowledge graph for the entire text data available. In the case of this paper, the text is in the form of cybersecurity related manuals.

4.3 Knowledge Graph Construction: Financial Regulations

Goal

The first goal of the main methodology behind knowledge graph construction is to create preliminary graphs. Although there is no domain expert to inspect them, the preliminary graph can still prove useful. As the authors of (4) highlight, their methodology can be reproduced in any domain, in order to create a ontology based schema, on which a knowledge graph is based on. The preliminary graph in the case of financial regulations, will be used to understand the financial regulations document and identify the main concepts on which the document is comprised of. The entities and relations which relate to those concepts will then be stored into triplets and will be the basis of the knowledge graph. This knowledge graph will be domain-specific, and more precisely, able to answer use case questions, based on the main concepts identified in the preliminary graph.

Use Case Extraction is thus the goal of the preliminary graph. Financial Regulations include 16 titles, as seen in 1. These titles are not always related to each other. For example, although Titles II,III and IV are directly related to regulations about the Budget of the European Union, the same can not be said for the rest of the Titles. Certain Titles such as Title V:Common Rules, Title VIII:Grants, Title XIV: External Audit and Discharge, are not related to the Budget of the Union, but rather concern different sub-domain within the overall domain of financial regulations. It is thus important to note, that using a sample text as done in (4), will not lead to generalization to the rest of the Titles. This is a problem, not only in the rule-based methodology of knowledge graph construction, but also in a potentially supervised methodology for knowledge graph construction.

Rule-Based openIE Pipeline

In order to extract information from the text of the financial regulations, an openIE methodology was followed. Specifically, a rule-based openIE pipeline was applied on the text. In the pipeline, NLP techniques were applied to perform NER and RE, in order to extract the SVO triplets, required for the preliminary graphs. The basis of this pipeline is a clause-based information extraction tool, found in the literature based on the paper (12). Information on the application the paper on text, can be found on the folloing Github repository, (10) as well as in the spaCy library (19). This clause-based tool is used to extract the SVO triplets needed for the preliminary graph and subsequenly for the knowledge graph.

In more detail, the following pipeline is followed for rule-based openIE knowledge graph construction from the financial regulations document:

1. PDF Parsing, Denoising and Cleaning
2. Coreference Resolution
3. Triplet Extraction: ClauseIE
4. Triplets Preprocessing
5. Preliminary Graphs Construction
6. Title III: Knowledge Graph Construction
7. Ontology Linking - FIBO
8. Use Case Questions
9. Evaluation: Use Case Queries

In the chapters that follow, an analytical implementation of the above pipeline, that leads to the knowledge graph construction, is described. The preliminary graphs are constructed for each Title of the financial regulations document, but the knowledge graph is constructed for Title III: Establishment and Structure of the Budget. The procedure followed to create the knowledge graph for a single Title can be followed for each financial regulations' title. Theoretically, the knowledge graph of each Title, can then be combined, to create an overall knowledge graph, covering the entire financial regulations document.

Furthermore, although an ontology is not constructed from scratch during the financial regulations knowledge graph construction, a predefined ontology is utilized. The Financial Instrument Business Ontology (FIBO)

(FIBO) is used as a knowledge base and less as a schema for the knowledge graph. The FIBO ontology will provide an additional layer of information for certain concepts in the knowledge graph. It will serve as a knowledge base and basically for Entity Linking (EL). Although FIBO alone, is not enough to explain the complex structure of the financial regulations, it is enough to explain certain financial domain concepts. Thus, FIBO does not serve as an ontology in the sense of providing a mapping layer for the knowledge graph structure, but rather serves as knowledge base that help to describe domain specific concepts. As will be seen in the final knowledge graph, FIBO, will help expand the knowledge graph, providing additional semantic information in the graph.

5 Knowledge Graph Construction: Rule-Based openIE

Ioannis Dikaioulias

5.1 PDF Parsing, Denoising & Cleaning

The raw data are in the form of an approximately 222 pages PDF document. In this first section of the pipeline, the goal is to extract the text from the PDF and prepare the text for the next steps of the pipeline. The first step is to parse the PDF, then apply certain NLP techniques to remove noise from the text, such as page marks, notes, numbering, included in the headers and footers of each page. The extracted text is then further segmented into Titles, Articles and paragraphs. The text is then further cleaned and prepared for the next step of the pipeline, which is coreference resolution.

From the 222 pages of the document, the first 25 are removed as they are considered unimportant to the knowledge graph. The first Title of the regulations starts at page 26 while the last Title ends at page 186. Thus, after removal of the first 25 pages, the remaining 186 comprise the document for parsing. The document is parsed using PyMuPDF Python library (Artifex). The document is thus transformed into a text file. The raw text is then splitted, per Title and per Article. During parsing, noise from the PDF is also passed into the text. Noise is defined as page marks and notes such as information of the authors about the regulations, numbers, which is visible in the headers and footers. An example of the noisy data is provided below:

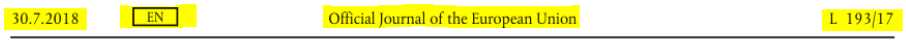


Figure 5.1: Headers: Page Marks and Notes

(1) Regulation (EU) No 1306/2013 of the European Parliament and of the Council of 17 December 2013 on the financing, management and monitoring of the common agricultural policy and repealing Council Regulations (EEC) No 352/78, (EC) No 165/94, (EC) No 2799/98, (EC) No 814/2000, (EC) No 1290/2005 and (EC) No 485/2008 (OJ L 347, 20.12.2013, p. 549).
(2) Regulation (EU) No 514/2014 of the European Parliament and of the Council of 16 April 2014 laying down general provisions on the Asylum, migration and Integration Fund and on the instrument for financial support for police cooperation, preventing and combating crime and crisis management (OJ L 150, 20.5.2014, p. 112).

Figure 5.2: Footers: Legal Details - Information

The noise is removed using regular expressions (5). Regular expressions allow to detect patterns in the text, and thus remove redundant information that matches those patterns. Noise is also removed from withing the text. Instances of (a), (b), (c)...lists are removed, while instances of redundant empty spaces are also removed. After noise is removed, the text is splitted based on detected patterns: first on the Title name of each Title, and then for each article name.

(a) the execution or both the execution and design of a work;

(b) the execution or both the execution and design of a work related to one of the activities referred to in Annex II to Directive 2014/24/EU; or

(c) the realisation, by whatever means, of a work corresponding to the requirements specified by the contracting authority exercising a decisive influence on the type or design of the work.

Figure 5.3: Alphabetical Lists Example

5.2 Coreference Resolution

Coreference resolution in Natural Language Processing (NLP), is the process of linking mentions in a text to the real world entities. For example, in the sentence "Michael's mother, Anna, is an accountant, she works in a insurance company", the token "she" refers to "Anna". This is completely understandable for the human mind to comprehend, but not for machines. There needs to be a way, that these mentions are connected to the entities they represent in the text. This is especially important, when Information Extraction, in the form of triplets extraction, is concerned. If these mentions are not replaced with their actual references in the text, then extracted triplets will not provide any contextual information. In the sentence above, the "she" token does not provide any information by itself. This problem is thus relevant in the task of preliminary graph construction. A graph consisting of pronouns as the head or tail of the graph, does not offer any semantic information. In conclusion, the reader, in this case, the domain expert, can not derive conclusions on the domain and make sense of the graph.

In regard to coreference resolution, spaCy's (19) neuralcoref module, is chosen to solve the above problem. This module is based on a neural network model, more specifically a scoring model as described (11). A description of the model is provided below:

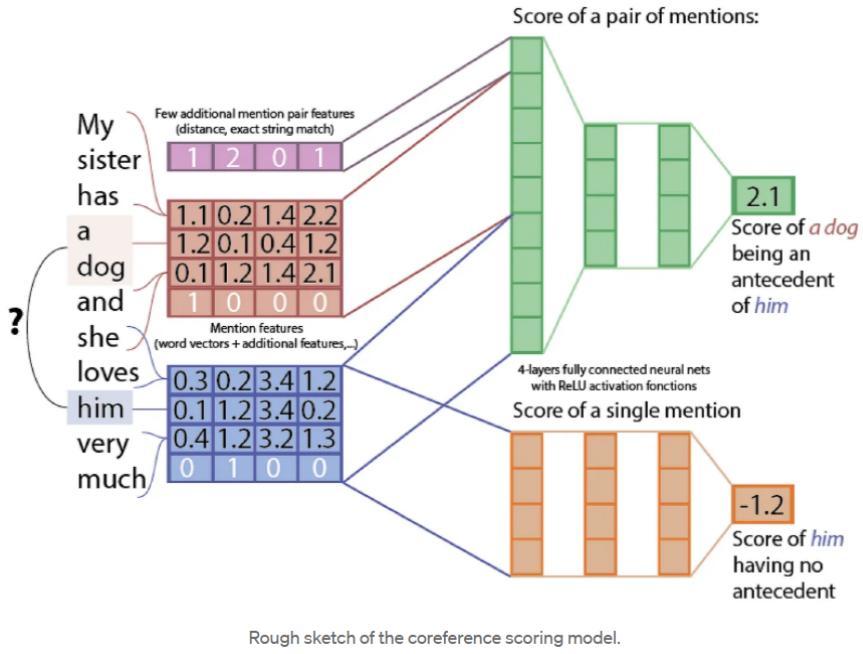


Figure 5.4: Coreference Resolution Architecture
(44)

First each mention and pair of mentions is represented by an embedding vector. These embeddings are calculated using a word2vec architecture (28). This vector is the average of certain vectors representing specific features, such as distance to the antecedent (the entity on which is the mention is linked), the word embedding of the token itself and the the word embeddings of the surrounding tokens to the mention. These embeddings are passed as an input to a first feed forward neural network, while embeddings of only the mention are passed to a second feed forward neural network. During the forwards pass, weights are optimized through a gradient descent algorithm, where back-propagation computes gradients for the weights, in order to minimize the following Loss function:

$$L(\theta) = \sum_{i=1}^{N} \max_{c \in C(m_i)} \Delta(c, m_i)\,(1 + s(c, m_i) - s(t_i^*, m_i))$$

Variables	Details
N	Number of mentions
$C(m_i)$	Antecedents per mention, m
$\Delta(c, m_i)$	Error cost
$s(c, m_i)$	Assigned Score to mention-entity pair(c, m_i).
t_i^*	True entity (antecedent)

The output of this first neural network provides a score for each mention-antecedent pair. These scores are not probabilities, but rather serve as a comparison metrics to obtain the mention that is more likely to refer to a specific entity (antecedent). The output of the second neural network concerns a score, only for a single mention. This score represents the existence or not of an antecedent.

An example of the coreference resolution scoring output is provided below:

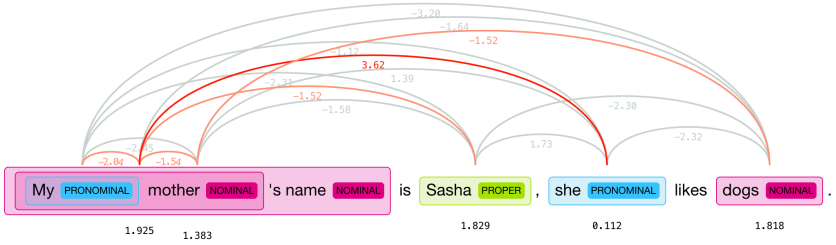


Figure 5.5: Coreference Resolution Example (3)

It is observed that pairs of mentions and entities are ranked based on scores, as to their semantic linkage. In the example above we observe that the "she" mention has a higher scoring link to "My mother", 3.62, than to "dogs", -2.32.

The neuralcoref model utilizes the above pretrained architecture to replace mentions in the text with their corresponding entity in the text.

The model is applied not on the sentence level of the financial regulations text. But is rather applied on each article content, cleaned of any noise and lower-cased. The whole content of each article is chosen, as to better capture the semantic information in the text and achieve better results.

After the pretrained architecture is applied, and the mentions in the text are replaced with the entities they represent, the following step is to extract the triplets. These coreferenced triplets would thus contain as much information as possible, for the domain expert to utilize.

Remark: The above architecture, provided by spaCy's neuralcoref module, can be fine-tuned on a specific

coreferenced dataset. This would allow to accurately measure the ability of the architecture to predict the correct mention-entity pair. There was no fine-tuning in the case of the financial regulations, and no manually constructed coreferenced ground truth dataset was used to evaluate how good the architecture captures the entities. Ideally this would allow to calculate how accurate the link between mentions and entities is.

5.3 Information Extraction: ClauseIE

Text Preprocessing

The following preprocessing pipeline is applied on the financial regulations text before the SVO triplets are extracted:

- Punctuation Removal: `\!@#$%^&*_-+{}\[\];\"\'\|<>,\?/~``'', is removed from the content of each article. Punctuation is removed to help the algorithm extract only the most important text, in the form of triplets. The "." is not removed as is required for the next step
- Sentence Segmentation: The content of each article is segmented into sentences or small text chunks. The segmentation is performed using the dot "." symbol.
- Text Retainment: After segmentation, only sentences that contain text are preserved. Sentences that contain non textual data, such as numbers or noise, are removed.

After the above preprocessing pipeline, all the sentences of the financial regulations are now collected. These sentences are lower-cased and cleaned from noise and punctuation. The sentences are now ready for SVO triplet extraction, through ClausIE.

ClausIE Algorithm

In (12) paper, the authors describe an open IE process of extracting Subject-Verb-Object triplets or propositions as referred to the paper. These triplets are extracted by utilizing NLP techniques, mainly dependency parsing. The goal of ClausIE is to identify clauses in the sentence. ClausIE can distinguish among seven different type of clauses as shown below:

Table 1: Patterns and clause types (based on [15]).

	Pattern	Clause type	Example	Derived clauses
			Basic patterns	
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_eC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
			Some extended patterns	
S_8:	SV_iAA	SV	AE died in Princeton in 1955.	(AE, died) (AE, died, in Princeton) (AE, died, in 1955) (AE, died, in Princeton, in 1955)
S_9:	SV_eAA	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton) (AE, remained, in Princeton, until his death)
S_{10}:	SV_cCA	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist) (AE, is, a scientist, of the 20th century)
S_{11}:	$SV_{mt}OA$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)
S_{12}:	$ASV_{mt}O$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

Figure 5.6: Clause Types
(12)

These clauses are then mapped into triplets. The distinction into clauses is performed in an algorithmic way. The methodology is purely unsupervised and does not require any training data. ClausIE extracts propositions (triplets), by utilizing grammatical and syntactical rules, based on the following procedure as outlined in the paper:

1. Compute the DP of the sentence

   DP refer to the Dependency Parsing of the sentence. Based on syntactical rules, the dependency tree of the sentence is extracted as in the following example:

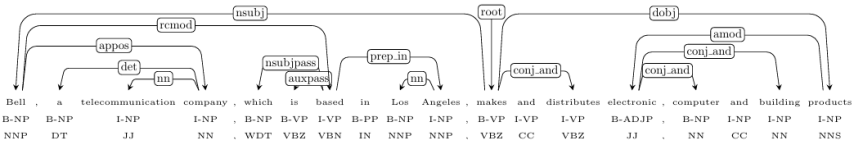


Figure 1: An example sentence with dependency parse, chunks, and POS tags (chunks by Apache OpenNLP)

Figure 5.7: Dependency Tree Example
(12)

2. Determine the set of clauses using the DP

3. For each clause, determine the set of coherent derived clauses based on the DP and small, domain-independent lexica

   The clauses are the determined and classified on their type, in an algorithmic way:

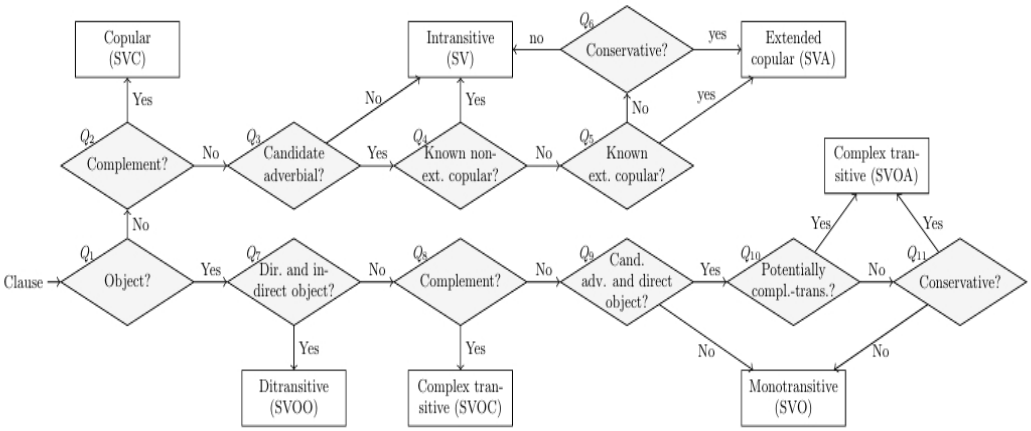


Figure 2: Flow chart for verb-type and clause-type detection

Figure 5.8: Clause Detection Algorithm
(12)

4. Generate propositions from (a subset of) the coherent clauses by mapping identified clauses to propositions. These propositions or triplets, can be multiple.

The user can choose to adjust how flexible the algorithm should be in clause detection. The default is to generate as many triplets as possible in the sentence. This behavior allows for higher recall (see Appendix A). On the other hand, there can be adjustments, such that clausIE limits the redundancy of the extracted triplets. This is achieved by either only keeping triplets of simple clause types or ignoring conjunctions during the clause identification algorithm. Conjunctions are words such as "and, or, in". This leads to a precision-recall trade off, with more limitations leading to a higher precision and less limitation, leading to a higher recall.

Triplet Extraction

The implementation of ClausIE, was performed through the claucy module from the work of (10). The module is applied through (19). For each sentence, a set of propositions are extracted. These propositions are filtered to generate full triplets. For each extracted proposition, the following triplet construction pipeline is followed:

- The Subject, if detected, is assigned as the Head in the triplet
- The Verb, if detected, is assigned as the Relation in the triplet
- The Object, if detected, serves as the Tail in the triplet
- If a Complement, an Adverbial or a combination of both is detected, they are assigned as the Tail in the triplet, in addition or not to the Object
- If a final triplet contains one or more empty values is removed. Empty values can be the result of the absence of any of the above part of speech entities, in a given proposition. A final triplet is considered valid if it contains three elements that represent the Head, Relation and Tail.

Table 5.1: Text to Propositions Example

Field	Value
Input text chunk - sentence	"basic act means a legal act, other than a recommendation or an opinion, which provides a legal basis for an action and for the implementation of the corresponding expenditure entered in the budget or of the budgetary guarantee or financial assistance backed by the budget, and which may take any of the following forms"
Output Propositions	[<SVO, basic act, means, None, a legal act, other than a recommendation or an opinion, which provides a legal basis for an action and for the implementation of the corresponding expenditure entered in the budget or of the budgetary guarantee or financial assistance backed by the budget„ None, []>, <SVO, which, provides, None, a legal basis for an action and for the implementation of the corresponding expenditure entered in the budget or of the budgetary guarantee or financial assistance backed by the budget, None, []>, <SVO, which, may take, None, any of the following forms, None, []>]
Number of Propositions	3
Final Triplet After Filtering	**Head**: {basic act}, **Relation**: {means}, **Tail**: {a legal act, other than a recommendation or an opinion, which provides a legal basis for an action and for the implementation of the corresponding expenditure entered in the budget or of the budgetary guarantee or financial assistance backed by the budget}

Heads & Tails Preprocessing

As highlighted in the literature review, the entities and relations extracted using an openIE method, especially a rule-based NLP method, such as ClausIE, leads to redundant and abstract triplets. The entities and relations are too generic and in big numbers.

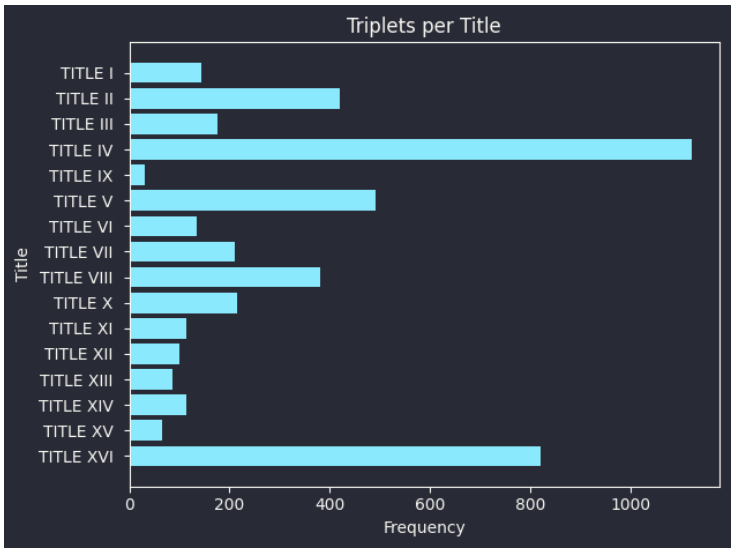


Figure 5.9: Before Refinement: Triplets per Title

There are 173 NA values detected, which are removed. There are in total, 4630 triplets predicted, with Titles IV and XVI having the most triplets and Titles IX and XV, having the lowest. The goal of this section is to filter the triplets and limit their generic and abstract nature. This will lead to more clear preliminary graphs. These clean graphs would help the assumed domain expert to understand the regulation's main concepts and thus design a ontology-based schema, describing the main classes (categories) of the entities how these entities connect with each other.

Each part of the triplets, namely the Head, Relation and Tail, are explored and preprocessed accordingly. The required preprocessing of each element is provided in the below sub-sections:

Relations

In order to limit the number of relations, first the relation element of each triplet was lemmatized. **Lemmatization** (5), converts a word into its canonical form. With reducing every relation to each lemma, relations that are different but are originated from the same lemma, are now same in terms of their meaning. This helps reduce the number of relations per Title.

Another step, which is performed simply for visualization reasons, is **2nd word removal**. Words that are prepositions, such as "to" or stopwords such as "be" are removed. This is performed by simply removing the second element in a relation, without any specific NLP method.

Heads

Stopwords are words in the English language, that appear common in the text. These include words such as "about, was, same, else, due, am, always...". Due to their high frequency, these words do not provide any significant meaning. In the effort to limit the redundancy of the entities, these type of words are detected and removed. The removal is performed through (19). Before stopwords are treated, **punctuation** is also removed from the Object items of the triplets. SpaCy provides a built-in list of stopwords of the English language. It

appears that in the Object part of the triplets across all Titles of the financial regulations, there are exactly 541 stopwords, or 11.68%. These stopwords are removed, as they provide no semantic meaning.

Lemmatization is also applied on each Object item of each triplet. Thus each Object (Head) of each triplet, is now cleaned of punctuation, is not a stopword and is further reduced to each lemma.

In addition to the above cleaning methods, spaCy's **Part-of-Speech (POS)** Tagger was utilized, in order to identify and keep only Object items, that contain nouns in their text, as far as their POS tagging is concerned. If a Object item, is more than one token, then the item is tokenized (5) and each token is inspected on each tag. Only Objects that contain nouns are thus preserved as Heads. The same noun check procedure is also applied on the Tail items.

A problem observed is that 636 object items, corresponding to 18.85% of the triplets, are comprised of **more than 6 tokens each**. For example, object items such 'criterion point subparagraph paragraph 2' or 'outcome build civil engineering work', or 'continuity union action management need', do not offer any semantic information. Ideally a human inspection of each individual object phrase from the 636 items, would be required to replace each of these phrases with the actual meaning they might actually infer. These 636 items are removed.

In conclusion, after the refinement of the Head and Tail parts of the triplets, the total triplets are limited from a total of 4630 triplets, to 3373 triplets. A drop of approximately 27%.

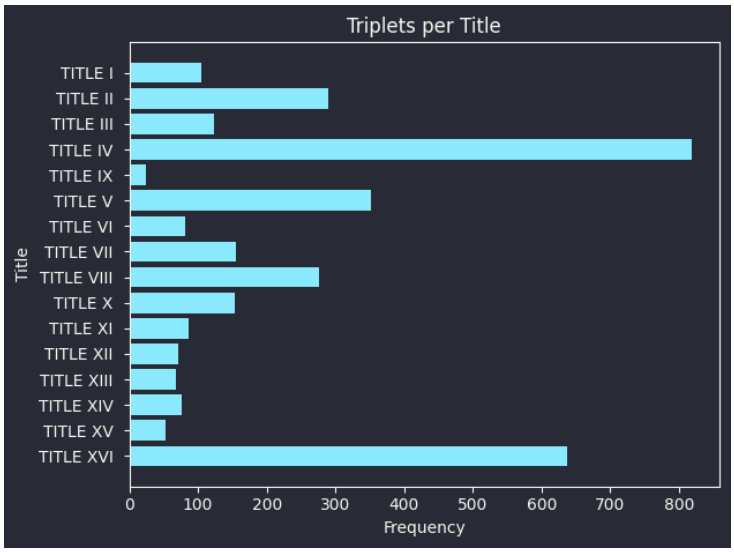


Figure 5.10: After Refinement: Triplets per Title

Tails Preprocessing

Tails Exploration - Concerns

Although the preprocessing of Heads and Tails was included in the previous section, together for both elements, this is not the case for the preprocessing of Tails. The significance of Tails in the preliminary graphs, as well as the more complex preprocessing needed, required a section of its own. A characteristic that the ClausIE produced Tail items, is that they are not comprised of a few tokens. In contrast, the majority of the Tail items, consist of long phrases or even large text chunks. This characteristic is both advantageous and disadvantageous. The advantage is that the domain expert, inspecting the preliminary graphs, can better understand the semantics behind the graph. The large textual information of the Tails, can help the assumed expert, to better grasp the contextual meaning and the content, by simple reading the Tail's text.

The disadvantages concern two main obstacles of having Tails with long text. The first obstacle is about the reproducibility of the preliminary graph. Visualizing the triplets in a preliminary graph, opens the way to

the power that graphs can offer. Graph algorithms exist, that offer transverse through a graph and filter out the most important concepts. Most of these algorithms can not be applied to elements that contain large noun chunks, as they are based on counting or similarity methods. Tails that consist of large chunks of text, are informative but at the same time unique and thus do not offer any information to ranking algorithms or pattern detection algorithms. The second obstacle is also around the uniqueness. Since almost each Tail will be unique, most of these Tails will have low connectivity with other Nodes (Heads or Tails). The main reason for the low connectivity is that these Tails will never have a Head duplicate in another triplet. For example, given an A-B-C triplet, where C is a long text unique Tail, there will not be another triplet, D-E-F, where C=A. This means that if there is semantic relationship between elements A and F, this will remain hidden and not visible to the domain expert.

In conclusion, due to the above two disadvantages, a solution is required to increase the semantic connectivity of the preliminary graphs, as well as, to offer a higher dynamic of the graph, in terms of graph application abilities.

Solution: TF-IDF Algorithm

A solution is proposed and applied, in order to transform the long text chunks into few tokens, and more precisely, into small noun chunks / noun phrases. The solution is inspired by (43), where they authors made use of the **TF-IDF algorithm** (see Appendix B), as a ranking mechanism, to keep only the most important tokens. The inspired solution in this thesis, uses the TF-IDF algorithm, as a ranking method, not on single tokens, but on calculated noun chunks. For each Tail text, the most important noun chunk, based on the TF-IDF algorithm, is chosen to represent the Tail. The importance is calculated both within the specific Tail text, and in relevance to all the other Tails.

The TF-IDF algorithm is a way to measure the importance of a token in a document, given a collection of documents. A document can be any form of textual data, such as news articles, paragraphs, sentences, or in the case at hand, text chunks-phrases, that represent the Tails in the extracted triplets.

The TF-IDF algorithm is applied on the collection of Tails. Stopwords are not taken into account during the application of the TF-IDF algorithm. Thus the extracted tf-idf weighted DTM matrix, does not contain any stopwords. Across the Titles of the financial regulations, the total collection of available Tails, T is considered:

$$T = \{t_1, t_2, t_3, \ldots, t_n\}$$

Each Tail, t, is considered an individual document. A weighted Document-Term Matrix (DTM) is calculated where rows correspond to the Tails collection and columns correspond to the Tails Vocabulary, meaning the unique collection of tokens that comprise the Tails. Each cell of the DTM matrix contains the TF-IDF score of the corresponding token, given the Tail.

For each token w in:

$$W = \{w_1, w_2, w_3, \ldots, w_n\}$$

where W, is the unique collection of tokens (Vocabulary), the sum of all the TF-IDF scores, across all tails, T, is calculated. For a given token w the following sum is calculated:

$$\text{Aggregated TF-IDF Importance Score} = \sum_{t \in T} \text{TF-IDF}(w, t)$$

This sum is used to compare its token with others in terms of its importance in the entire corpus (collection of Tails). Each token is thus mapped to an TF-IDF based, importance score, as such:

Table 5.2: Extracted Tail Tokens - Aggregated TF-IDF Importance Scores

Tail Tokens	Aggregated TF-IDF Importance Score
article	107.1837
union	72.2838
financial	71.5598
paragraph	56.6723
following	53.9887
accordance	49.1715
point	46.7656
subparagraph	44.5331
referred	43.5970
commission	40.5536
officer	39.5320
budget	36.5869
european	35.4283
year	34.4690
regulation	32.8489

The above importance scores are related to the tokens. The goal is to choose the most important noun chunk hidden in each Tail. Through the help of spaCy's noun chunk extraction procedure as found (19), the noun chunks of each Tail are extracted. The noun chunks extraction is based on the dependency tree of each Tail text. An example is provided below, where the dependency tree of the sentence "the expenditure of the european agricultural guarantee fund (eagf)", is provided:

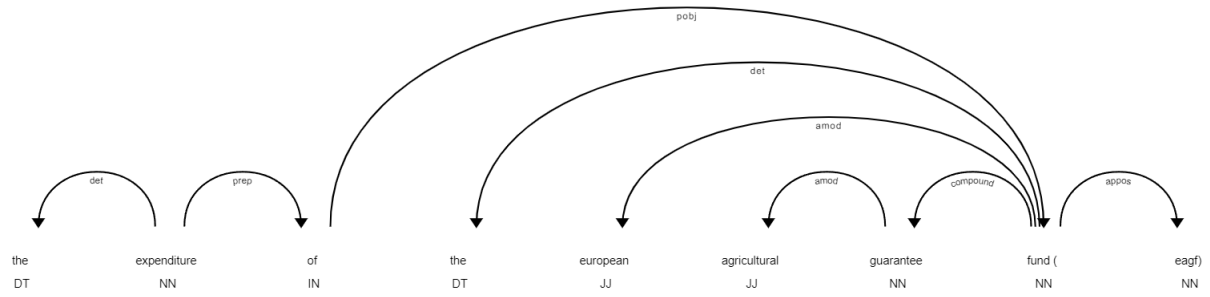


Figure 5.11: Dependency Tree - Noun Chunks Extraction

Table 5.3: Extracted Noun Chunks from DP Tree Example

Extracted Noun Chunks - spaCy
the expenditure the european agricultural guarantee fund eagf

The above noun chunks contain stopwords such as "the". These stopwords are removed from the extracted noun chunks, in order to keep only the most important words.

Given the collection of all Tails:

$$T = \{t_1, t_2, t_3, \ldots, t_n\}$$

each Tail, t, is broken down into a set of noun chunks:

$$C = \{c_1, c_2, c_3, \ldots, c_n\}$$

For each noun chunk, c, the noun chunk is broken down into tokens W:

$$W = \{w_1, w_2, w_3, \ldots, w_n\}$$

Each token, w, is mapped to the corresponding aggregated TF-IDF importance score. For a given noun chunk c, belonging to a certain Tail, t, the following sum is calculated:

$$\text{Noun Chunk Importance Score} = \sum_{w \in W} \text{Aggregated TF-IDF Importance}(w, c)$$

This sum, serves as the importance score for each noun chunk in a given Tail. The noun chunk with the highest importance score, is selected to represent the Tail. The extracted noun chunks are cleaned of their punctuation and any noun chunk which is a stopword is removed.

The following table illustrates an example of the top noun chunk selection:

Table 5.4: Top Noun Chunk Selection Example

Original Extracted Tail	**Extracted Clean Noun Chunks** (Punctuation & Stopwords Cleaned)	**Top Noun Chunk** (Highest Importance Score)
the rules for the establishment and the implementation of the general budget of the european union and of the european atomic energy community (the budget) and the presentation and auditing of the european union and of the european atomic energy community accounts	['rule', 'establishment', 'implementation', 'general budget', 'european union', 'european atomic energy community', 'budget', 'presentation', 'auditing', 'european union']	european union

The "european union" obtains the highest Noun Chunk Importance Score, compared to the other noun chunks of the Tail's text.

Remark: The TF-IDF algorithm as a ranking method, has the main disadvantage that it does not capture contextual information. For instance, in the example provided above, the "european union" noun chunk does not capture the semantic information of the original extracted tail. It is evident, that for the domain expert, the text in the original extracted Tail, offers more information, in regard to the context. Nevertheless, as outlined in the beginning of this section, a single token Tail, offers graph transversal capabilities and open the pathway for other possible automated methods for knowledge graph construction, as is described in a different chapter of this thesis. Ideally, word embeddings, could be used to obtain the most important noun chunk. For example, word embeddings, would have the ability to capture the semantic information hidden in the noun chunks, in context of the whole Tail's text. An example of such method, would be to vectorize the noun chunks using word embeddings. Then compare each noun chunk's embedding representation to the embedding representation of the whole Tail. The noun chunk or a combination of noun chunks, with the highest similarity in terms of a distance measure, such as euclidean distance, to the whole Tail text, would be

selected to represent the whole Tail.

Conclusion

The overall cleaning of the triplets, specifically of the Heads, Relations and Tails, led to a decrease of the triplets, from 2737 to 4630, translating to a decrease of approximately 41%.

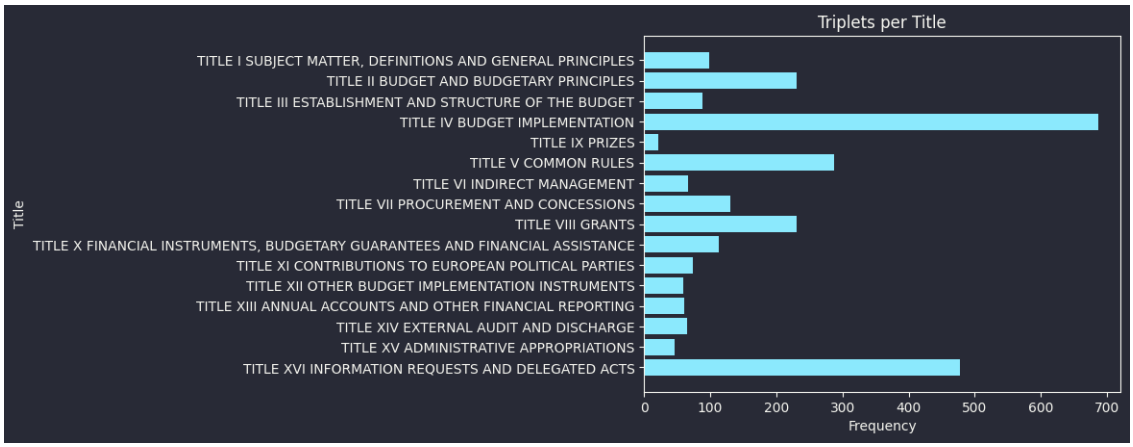


Figure 5.12: Reduced Triplets per Title

5.4 Preliminary Graphs

There are 3 possible versions of preliminary graphs, depending on how the Tail element of the triplets is handled:

- **Version A:** The whole text of the Tail is used as the Tail
- **Version B:** Each extracted noun chunk is used as a new Tail
- **Version C:** Only the highest importance noun chunk is used as a new Tail

Version A and B are implemented in this section to create preliminary graphs for each Title of the financial regulations. Version C is not utilized, as the choice of a single noun chunk does not offer a high amount of semantic information, in terms of representing the entire Tail text. To illustrate the different versions, the Tail example from 5.12 is used to give an example of the possible triplets:

Table 5.5: Triplet Versions Example

Version	Head	Relation	Tail
A	regulation	lay	the rules for the establishment and the implementation of the general budget of the european union and of the european atomic energy community (the budget) and the presentation and auditing of the european union and of the european atomic energy community accounts
B	regulation	lay	rule
B	regulation	lay	establishment
B	regulation	lay	implementation
B	regulation	lay	general budget
B	regulation	lay	european union
B	regulation	lay	european atomic energy community
B	regulation	lay	budget
B	regulation	lay	presentation
B	regulation	lay	auditing
B	regulation	lay	european union
C	regulation	lay	european union

It is visible from the example, that Version A triplets contain the most contextual information for the domain expert. Version A triplets use the long text content, as the Tail. This does not have an effect on the size of the triplets. On the other hand, Version B triplets, have an effect on the size, since as seen in the above example, a single triplet is converted into ten triplets. This leads to a significant increase in the size of the triplets for each Title, and subsequently leads to large preliminary graphs.

Version A and Version B triplets are subsequently preprocessed. Each Version is preprocessed in order to filter and limit the size of the triplets for each Title. The reasons behind the refinement of the size, is again to limit the redundancy and abstractness of the produced preliminary graphs. The goal of the preliminary graphs, especially of the Version A graphs, is provide semantic information to the domain expert, and thus need to be easy to read and understand for the human eye.

Version A - Whole Tail Preliminary Graphs

For Version A triplets, there are three different variations of preliminary graphs created. The **first variation** concerns the creation of preliminary graphs that contain all the available triplets. The **second variation** and **third variation** concern the construction of smaller preliminary graphs, where triplets are removed based on the following preprocessing:

Preprocessing - Filtering

The focus of the filtering of each Title, is on the Relation term of the triplets. For each Title the following pipeline is followed:

1. A descriptive table is created providing statistics, namely the frequency, relative frequency, cumulative frequency and relative cumulative frequency, of each unique relation in the triplets. Visualizations are also provided, regarding the exploration of the Relations.

2. Relations are filtered out, based on a frequency threshold, which is manually decided, depending on the capture cumulative frequency of the triplets and the preliminary graph appearance in terms of interpretability

3. The preserved Relations and their corresponding triplets are used to construct a preliminary graph for the relevant Title

4. The constructed preliminary graph is stored and visualized using the Neo4j Graph Database System, and more specifically the Neo4j AuraDB (29)

The above procedure is repeated for each Title of the Financial Regulations document, thus a preliminary graph exists for each Title. An example of the pipeline implementation and the construction and visualization of the preliminary graph, is provided in the below example. The preliminary graph construction in the example, concerns Title III: Establishment and Structure of the Budget.

Preliminary Graph - Title III

Title III contains in total, 89 triplets, after duplicates of exactly equal triplets are removed. A descriptive table regarding the unique Relation terms, of these triplets, is created along with certain exploratory plots. The goal is to explore the unique Relations and then decide on a frequency threshold. Triplets that have Relations with frequency falling below that threshold, will not be taken into account.

Table 5.6: Descriptive Table - Relations: Title III

Relation	Frequency (n)	Cumulative Frequency (N)	Relative Frequency (f)	Cumulative Relative Frequency (F)
include	7	7	0.078652	0.078652
refer	5	12	0.056180	0.134831
contain	4	16	0.044944	0.179775
show	4	20	0.044944	0.224719
make	3	23	0.033708	0.258427
present	3	26	0.033708	0.292135
be	3	29	0.033708	0.325843
comprise	2	31	0.022472	0.348315
provide	2	33	0.022472	0.370787
exceed	2	35	0.022472	0.393258
enter	2	37	0.022472	0.415730
accompany	2	39	0.022472	0.438202
correspond	2	41	0.022472	0.460674
attach	2	43	0.022472	0.483146
realise	1	44	0.011236	0.494382
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
group	1	87	0.011236	0.977528
act	1	88	0.011236	0.988764
use	1	89	0.011236	1.000000

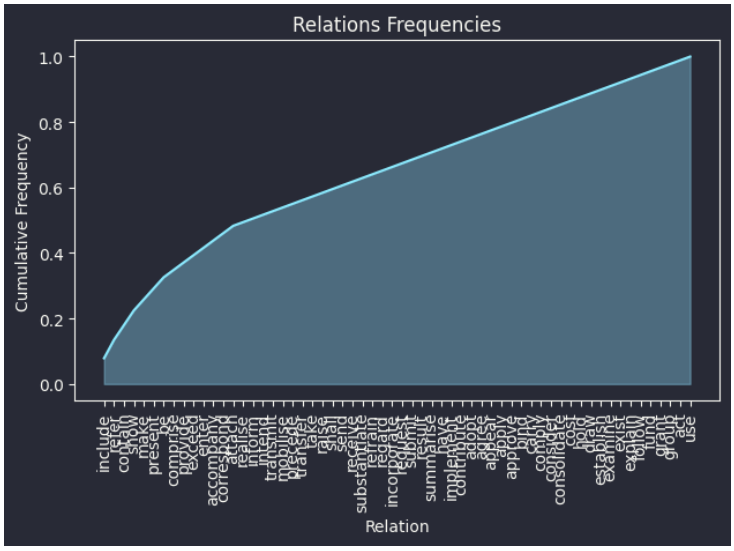


Figure 5.13: Cumulative Frequency of Relations

From 5.6 and 5.13, it is observed that most Relations, appear only once. More than 50% of Relations appear once across the triplets, meaning that there are triples which do not repeat more than once in terms of their Relations. Triplets, more thaN n=43, only appear once.

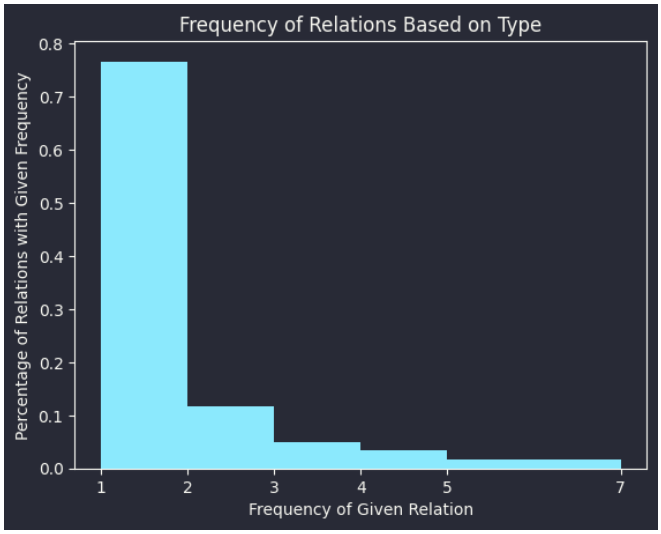


Figure 5.14: Relations Per Frequency Type

In the cumulative frequency line of 5.13, there appears to be a change in the slope, at a value of F=0.4, which approximately corresponds to the "attach" Relation. In regard to Title III, it is decided that the cut-off will be n=2 where F=0.483146 and N=43. Above this cumulative frequency level, all Relations will be removed.

The final Relations captured are the following:

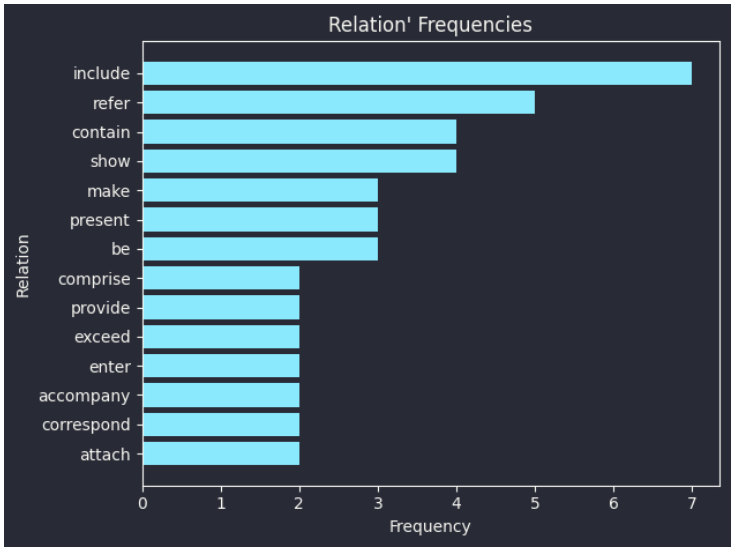


Figure 5.15: Final Relations: Title III

These Relations correspond to a total of N=43 triplets, while around 48% of the triplets is captured.

The same procedure of manually deciding a threshold was repeated for all the Titles. In certain Titles, namely Titles II, IV, V, VII, VIII, XVI, in addition to the initial cut-offs, additional smaller versions, of stricter filtering of the triplets, was conducted. In conlusion, for each Title, large unfiltered preliminary graphs and smaller filtered preliminary graphs were created. For certain Titles, even smaller preliminary graphs were created.

The following table showcases, for each Title, the different frequency thresholds and the captured percentage of triplets:

Table 5.7: Captured Triplets Per Preliminary Graph - Title

Title	**Frequency (n) Threshold**(Small Prelim.Graph)	**Captured Triplets (%)**(Small Prelim.Graph)	**Frequency (n) Threshold (%)**(Smaller Prelim.Graph)	**Captured Triplets (%)**(Smaller Prelim.Graph)
I	65	68.04	-	-
II	2	67.69	9	26.10
III	1	48.31	-	-
IV	6	47.53	48	10.31
V	2	55.87	9	26.33
VI	1	54.54	-	-
VII	1	71.87	2	48.43
VIII	2	75.55	7	29.33
IX	0	100	-	-
X	1	56.63	-	-
XI	1	62.50	-	-
XII	0	100	-	-
XIII	1	68.85	-	-
XIV	1	61.90	-	-
XV	1	65.21	-	-
XVI	9	42.60	54	15

From 5.7 a trade-off can be easily observed between a stricter, higher threshold and a lower captured information, as seen in the lower captured amount of triplets. Although more captured triplets offer more information to the preliminary graph, certain Titles such as Title XVI and Title IV, contain such a high amount of triplets, as seen in 5.12, that the produced preliminary graphs are difficult to interpret by a domain expert. Thus, especially under the shortage of time, the domain expert can choose from smaller versions of the preliminary graphs of certain Titles.

Preliminary Graph - Neo4j GDBS

In this section, the different variations of the Version A preliminary graphs are stored and visualized using the Neo4j Graph Database system. The three variations, depend on the existence or not of a frequency threshold:

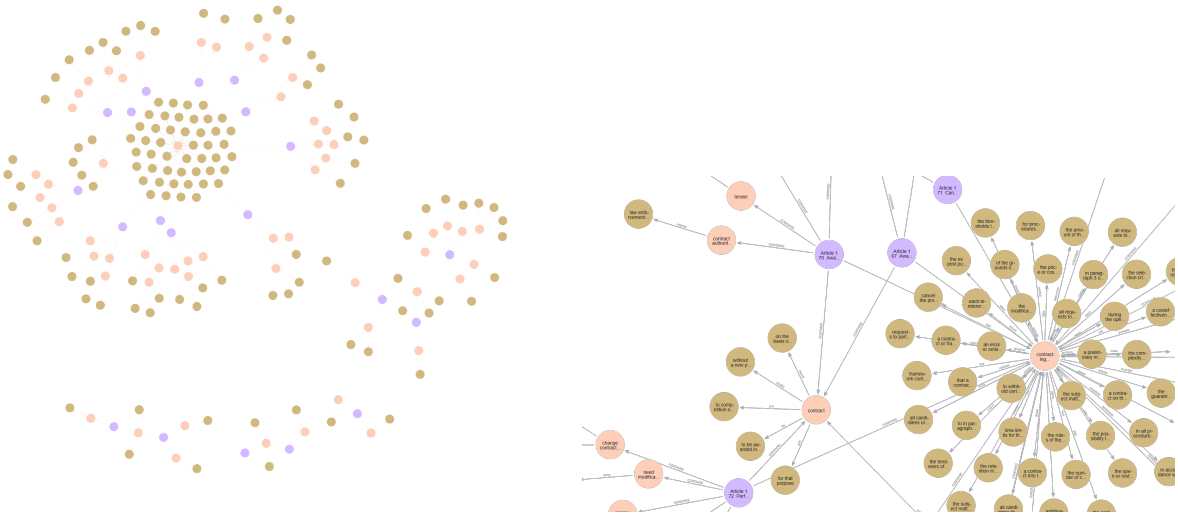
- No relations' frequency threshold applied to the triplets, thus producing preliminary graphs with 100% capture triplets.
- A moderate relations' frequency threshold applied, thus producing smaller version of preliminary graphs with less than 100% captured triplets
- A stricter-higher relations' frequency threshold applied to certain Titles, thus producing even smaller preliminary graphs, with a low value of captured triplets.

For each Title, each preliminary graph, contains Article Nodes, Head Nodes and Tail Nodes. An example of the created preliminary graphs for Title VII regarding procurement and concessions, is provided below.

Table 5.8: Preliminary Graphs Nodes, Edges And Properties

Nodes	Node Properties	Edges
Head	{Name: Head Text}, Article	{To Tail: Verb}, {From Article: "CONTAINS"}
Tail	{Name: Tail Text}, Article, Contained Noun Chunks	{From Head: Verb}
Article	{Name: Article Title}, Content, Regulations Title	{To Head: "CONTAINS"}

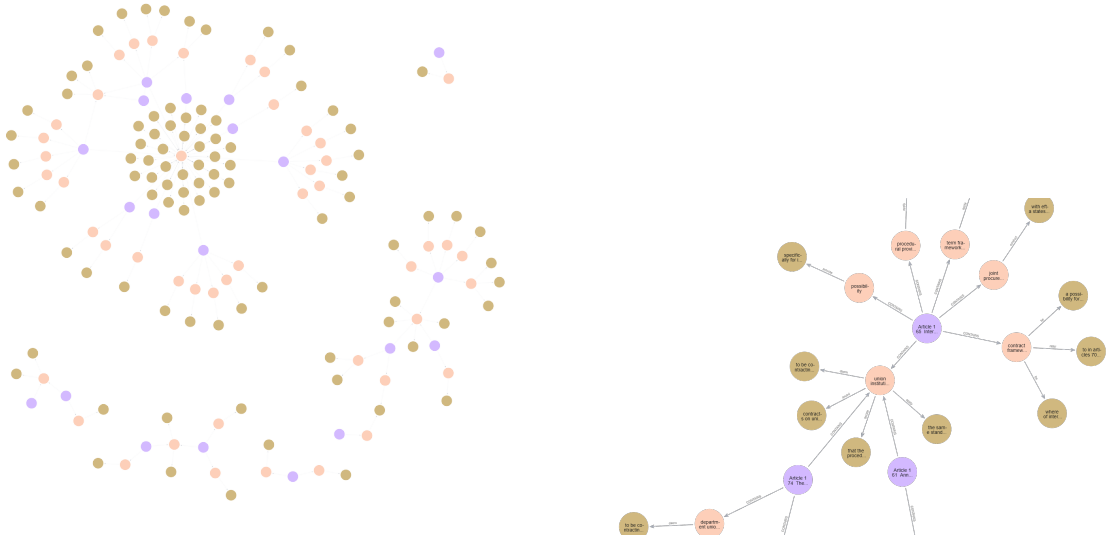
Figure 5.16: Full Whole Tail Preliminary Graph - Title VII



(a) Graph Overview

(b) Inside the Graph

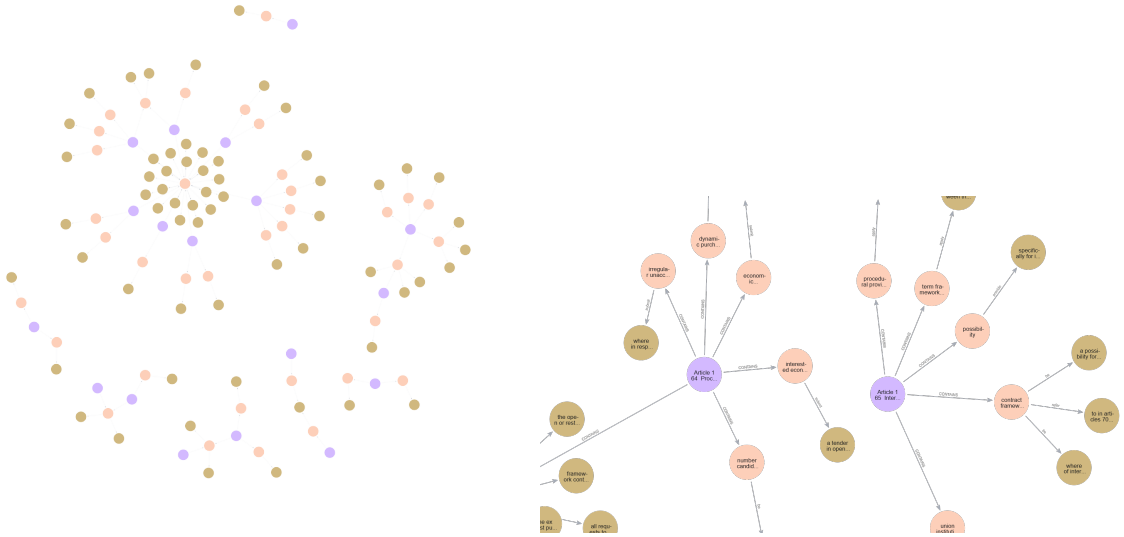
Figure 5.17: Moderate Whole Tail Preliminary Graph - Title VII



(a) Graph Overview

(b) Inside the Graph

Figure 5.18: Small Whole Tail Preliminary Graph - Title VII



(a) Graph Overview

(b) Inside the Graph

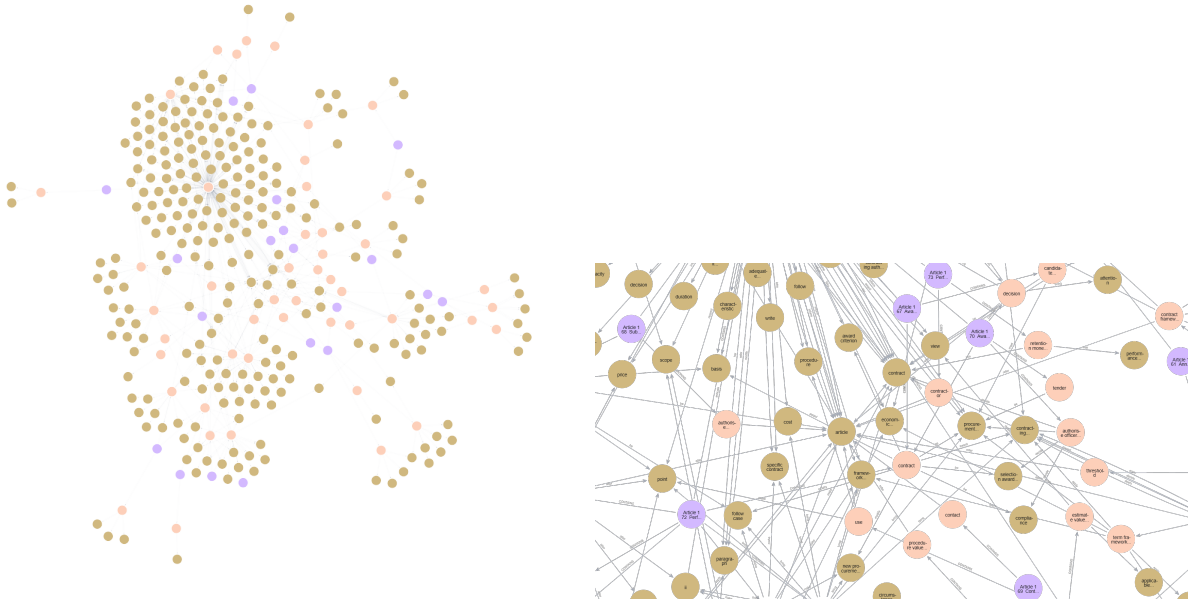
Version B - Expanded Tail Preliminary Graphs

The Version B preliminary graphs are produced by expanding each Triplet into a set of sub-Triplets based on the same extracted noun chunks from each Tail. So given an $A - B - C$ triplet, the Tail C is broken down into individual noun chunks $\{c_1, c_2, c_3, \ldots, c_n\}$, where n is the total extracted noun chunks. Thus, a set of $\{A - B - c_1, A - B - c_2, A - B - c_3, \ldots, A - B - c_n\}$ triplets are produced out of every given $A - B - C$ triplet.

Version B triplets are broken down into two variations. The first variation derives by using the previously produced moderate preliminary graphs and expanding the Tails. The second variation, derives by using the previously produced small preliminary graphs (for the relevant Titles), and also expanding the Tail.

For each Title, each preliminary graph, contains Article Nodes, Head Nodes and Tail Nodes. An example of the created preliminary graphs for Title VII regarding procurement and consessions, is provided below.

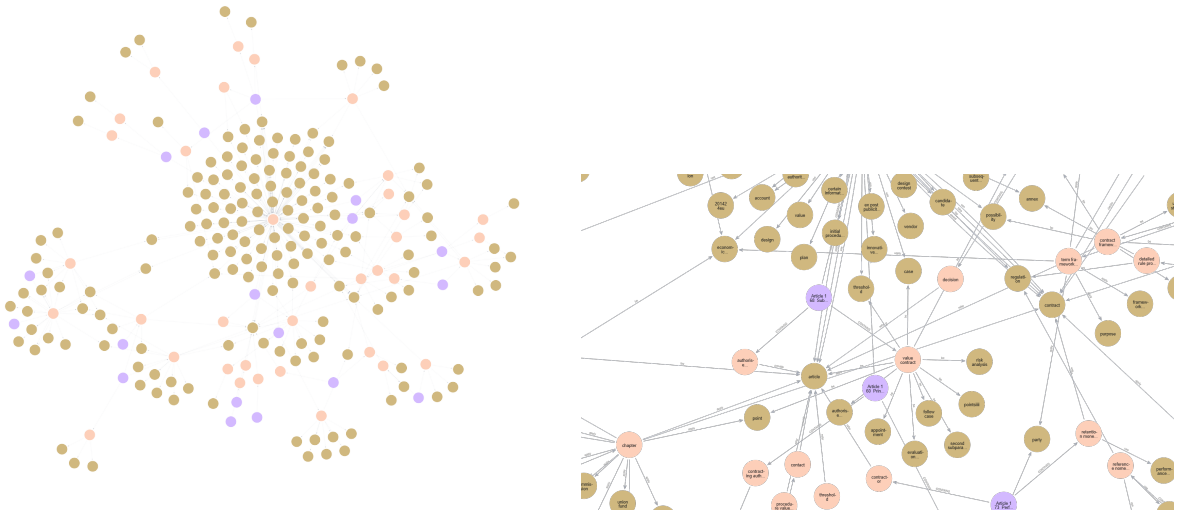
Figure 5.19: Moderate Expanded Tail Preliminary Graph - Title VII



(a) Graph Overview

(b) Inside the Graph

Figure 5.20: Small Expanded Tail Preliminary Graph - Title VII



(a) Graph Overview

(b) Inside the Graph

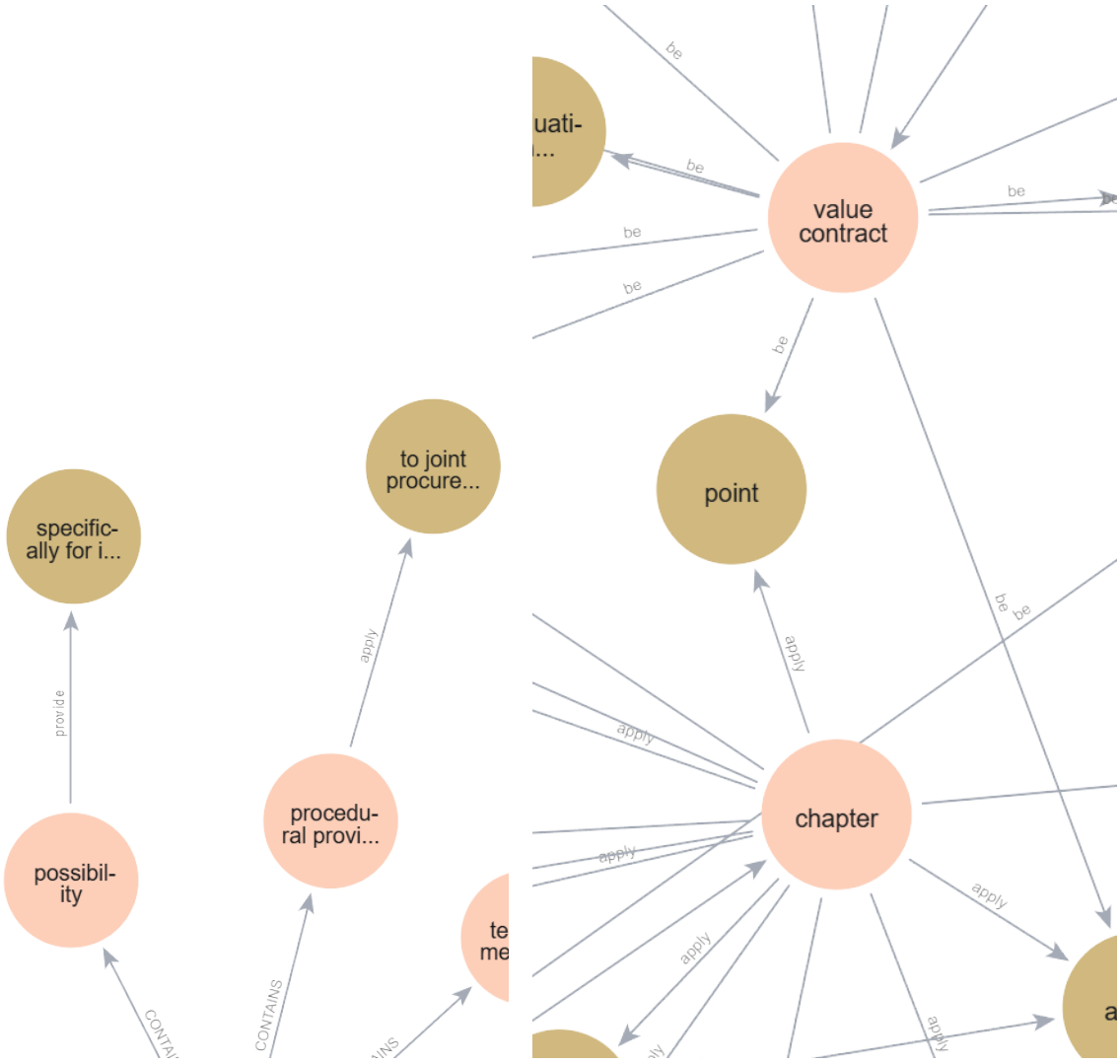
Version A VS Version B Preliminary Graphs

The two main differences between the two Versions, is the interconnection of the Nodes, the human intervention capability and the further usage.

In terms of nodes' interconnection, Version A preliminary graphs, offer fewer connections, but with higher semantic information, compared to Version B preliminary graphs. Version A graphs, contain Head-Relation-

Tail triplets, where the Tail nodes are isolated and only act as the end node of a triplet, with connections only leading to them and never from them to other nodes. This is not true for Version B Tails, where Tails, can have outwards connections with other nodes, thus providing higher connectivity in the graph.

Figure 5.21: Interconnectability Difference Between Versions - Title VII



(a) Version A - Lower Interconnection

(b) Version B - Higher Interconnection

In the Version B example, it can observed that the Tail node "point" is connected with both the "value contract" and "chapter" Head nodes. This is not the same with the Version A example, where the Tail node containing the long "subject" text of the SVO triplet, only connects with a single Head node.

5.5 Knowledge Graph Construction

A domain-specific knowledge graph, or as referred in (18), an enterprise knowledge graph, needs to be able to answer certain questions. These questions are directly related with the domain at hand, and subsequently with the ontology of the knowledge graph. In (Noy and McGuinness), the questions that the domain-specific

ontology should be able to answer are called **competency questions**. These questions define the purpose of the knowledge graph, in other words, define the domain use cases that the knowledge graph aims to answer. The formulation of these questions is the responsibility of the domain expert. One should have a solid knowledge of a domain, in order to derive the use case questions. The use case questions are the foundation of a domain-specific knowledge graph, as they link it with the domain specific ontology that the knowledge graph is based, and most importantly, they define its main purpose.

In order for the reader to better understand the concept of the **use cases** behind a knowledge graph, the authors of (37), provide the following example:

Name: Use Case 1 - Real Time Parking

Description: Finding available parking spaces in certain areas and times of day is an almost impossible task in certain cities. Faced with this problem, a city council has deployed a network of sensors in public car parks to obtain data in real time and to provide citizens with information about free and occupied places. In addition, it has placed display panels of available parking spaces in the streets and has made available to the public a mobile application for guided parking. Thanks to this system, the citizens can know in real time in which public car park there are free places to leave their car.

Actors: citizen, application

Flow: Mike, while driving his car to the city center, activates through a voice command the parking application. The application requests the destination address. Mike specifies the destination address to the application. The application, which has access to the GPS of his mobile, inspects the data emitted by the parking sensors, identifies the parking places closer to Mike's destination from its current location, and recommends him a route to park, however the spot is not reserved for him. Mike follows the route suggested by the application, arrives at the place and parks.

Figure 5.22: Use Case - Real Time Parking
(37)

The goal of the knowledge graph is to describe a use case flow such as in 5.22, but for the financial regulations domain. The competency or use case question, not only help define such a use case, but also help define the ontology-based schema on which the knowledge graph is based.

The goal of the preliminary graph is to help the domain expert, define the use case questions and define the ontology-based schema of the knowledge graph. As seen in 3, knowledge graph construction main obstacle are the absence of a domain-specific ontology and/or the absence of entity-relation domain specific labeled dataset. In this thesis, another main obstacle is the absence of a domain expert. Ideally a domain expert would inspect the preliminary graphs, understand the context and define the ontology-based schema. In the absence of the domain expert, the author of this section assumed that role, in order to showcase an ontology-schema construction out of a preliminary graph. Title III: Establishment and Structure of the Budget, was chosen to derive the knowledge graph.

As seen in the 3, there are many ways to construct an ontology. The approach of utilizing the preliminary graph to derive the most important entities and categories and how they relate, is directly related to the bottom-up approach, where the ontology is constructed, starting from the collection of entities and then the most important entities-categories are chosen to comprise the schema.

Use Case Extraction

Ideally, a domain expert, would first derive the competency questions and define the use cases, and thus build an a schema. Contrary to that procedure, the role of the domain expert is assumed by the author of this section and a bottom-up approach is followed, through the help of the preliminary graph. In order to derive the main concepts - categories of the entities and how they relate to each other, the ontology construction methodology of the seven-step method, as described in (Noy and McGuinness) was used as a guide. The seven-step method was not followed step by step to construct an ontology from scratch but rather served as a guide to define the main concepts and thus construct the use case/competency questions. Therefore, the goal is to define the main classes and subclasses that the entities in the preliminary graph belong, as well as the main relations that connect those classes together. This procedure, requires human intervention, in order to manually inspect the preliminary graph, understand the semantic connections and contextual information hidden in the graph, and then derive the most important concepts related to the corresponding Title of the financial regulations, linked to that preliminary graph.

The following pipeline was followed in order to transform the preliminary graph of Title III, to a knowledge graph:

1. Inspection of the Title III, full (no frequency threshold triplet reduction), whole tail, preliminary graph. Manual inspection of each triplet in the preliminary graph.

2. Manual cleaning of the entities in the preliminary graph: Removal of entities with no visible contextual information. Merging of differently written but similar entities in terms of meaning. Rewriting of entities in order to better grasp their semantic meaning.

3. Definition of main classes and of the classes' hierarchy, visualized through class-subclass relations. Assignment of each entity to its corresponding class.

4. Preservation, removal, or adjustment of the relations of the preliminary graph to better match the relations among the entities and the classes the represent.

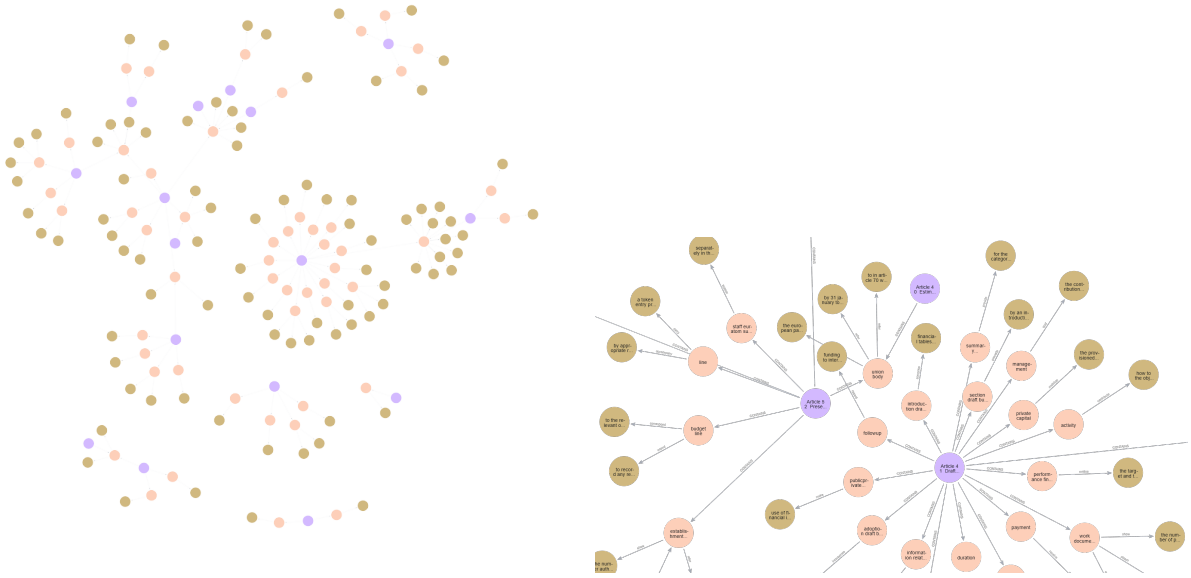
5. Preservation of the node properties of the preliminary graph, as well as the inclusion for each node, of the article name, that concerns the node, and of the content of the article.

6. Storage of the knowledge graph in the Neo4j Graph Database and visualization

Manual Exploration of the Preliminary Graph

The following is the full, no frequency threshold applied, whole Tail preliminary graph for Title III of the Financial Regulations:

Figure 5.23: Full Whole Tail Preliminary Graph: Title III

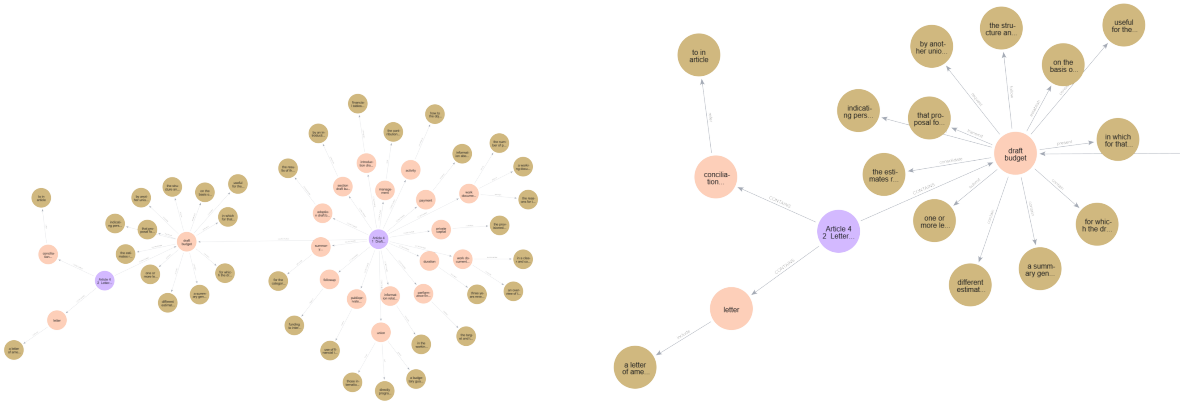


(a) Graph Overview (b) Inside the Graph

From the manual exploration of the graph, the semantic information of Title III was explored, by analyzing the clusters observed in the graph and studying the properties of nodes and the connections offered by the relations. Especially the content of each Tail and the content of each Article, were essential in understanding the contextual information of the graph and derive the main classes of the entities of the graph and the use cases around those classes.

An example is provided, on the manual preprocessing of a specific part of the graph, more specifically, of the part of the graph regarding the procedure of the Budget Amendment:

Figure 5.24: Budget Amendment Use Case - Preliminary Graph



(a) Budget Amendment - Overall Cluster (b) Budget Amendment

In 5.24a each triplet is explored in terms of the conveyed meaning. It can be observed that Budget Amendment concerns "Article 42 Letter of Amendment to the Draft Budget" as well as the overall main

article, namely "Article 41 Draft Budget". The Nodes of this cluster are studied in terms of their content, by looking at the properties of the nodes, more specifically:

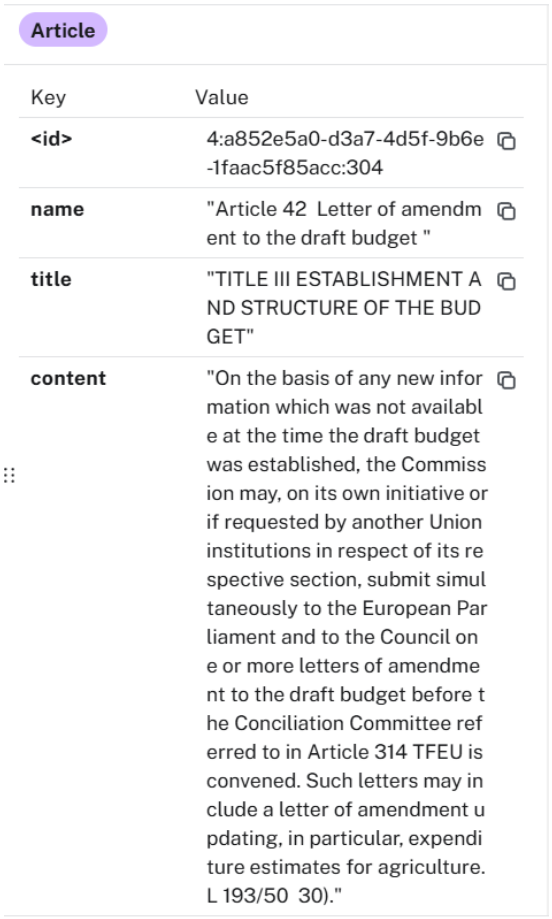


Figure 5.25: Article Node: Properties

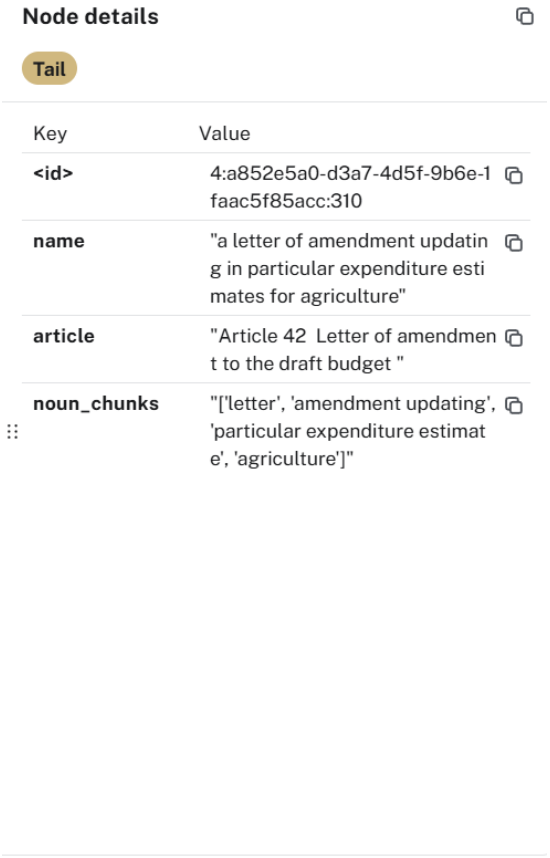


Figure 5.26: Tail Node: Properties

From analyzing this particular cluster of the preliminary graph, the following entities, relations and classes were derived:

Table 5.9: Budget Amendment: Schema - Main Entities-Relations & Classes

Entities	Relations	Classes
Letter of Amendment Amendment Updating Budget Amendment European Commission European Council European Parliament Expenditure Estimates Draft Budget	CONCERNS INCLUDED_IN CREATED_BY SEND_TO	Budget Amendment European Union Estimates Draft Budget

A use case based on these entities was extracted. A format similar to 5.22, as derived from (37), could be used to create a use case for Budget Amendment:

Table 5.10: Use Case: Budget Amendment of the Draft Budget

Description	As described in Article 42 of the ...reference financial regulations, there can be a request for the Draft Budget of the European Union, to be amended, meaning to be altered or adjusted. The amendment procedure is requested by other a specific Union Institution or is initiated by the European Commission, by its own decision. The amendment procedure, is performed, through one or more letters of amendment, sent by the Commission to the European Parliament and to the Council. In most cases, the Draft Budget Amendment concerns agricultural expenditure estimates of the Budget.
Actors	European Union, Union Institution, Commission, European Parliament, Letter, Expenditure, Amendment
Flow	The Commission after careful analysis of the expenditure estimates described in the Draft Budget, decides to perform a budget update regarding the estimates for agriculture. Thus the Commission, submits a letter of amendment to the European Parliament and the Council, requesting the amendment updating, before the Conciliation Committee as refered to in Article 314 TFEU of the Finacial Regulations.

The use case is linked to the following example **use case questions** regarding the Budget Amendment use case:

1. Who submits the Letter of Amendment and to whom?

2. What is the purpose of the Draft Budget's Amendment?

3. What type of expenditures does the Amendment mostly concerns?

4. What is time frame of the Letter's submission?

Main Schema Concepts - Title III

A similar pipeline of use case extraction by thorough examination of the preliminary graph was across all the triplets and identified clusters. The following classes were derived from the preliminary graph of Title III: Establishment and Structure of the Budget:

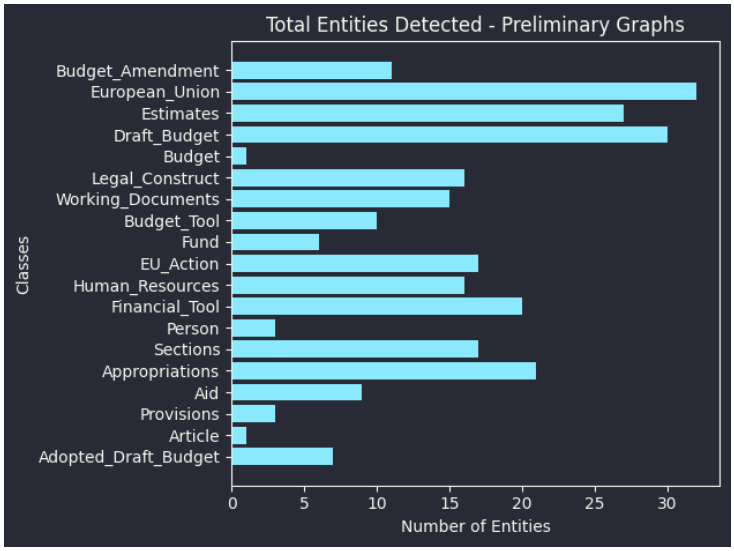


Figure 5.27: Total Entities per Class

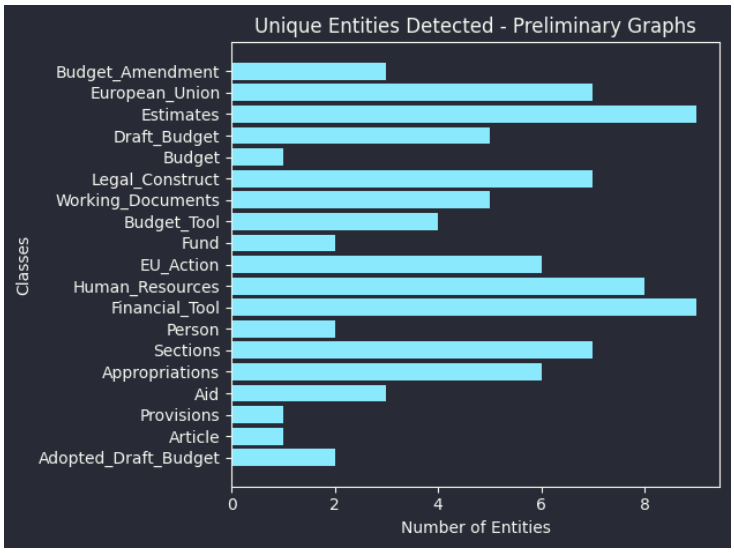


Figure 5.28: Unique Entities per Class

These are the classes that comprise the schema of the Title III Knowledge Graph. The classes are hierarchical related, with certain classes being sub-classes.

Figure 5.29: Extracted Entity Classes

Title III Knowledge Graph

After manual extraction of the main use cases and classes, entities and relations, such as in 5.9, that comprise the schema of each identified use case in the preliminary graph, the following knowledge graph was constructed based on the extracted classes shown 5.29 or 5.28:

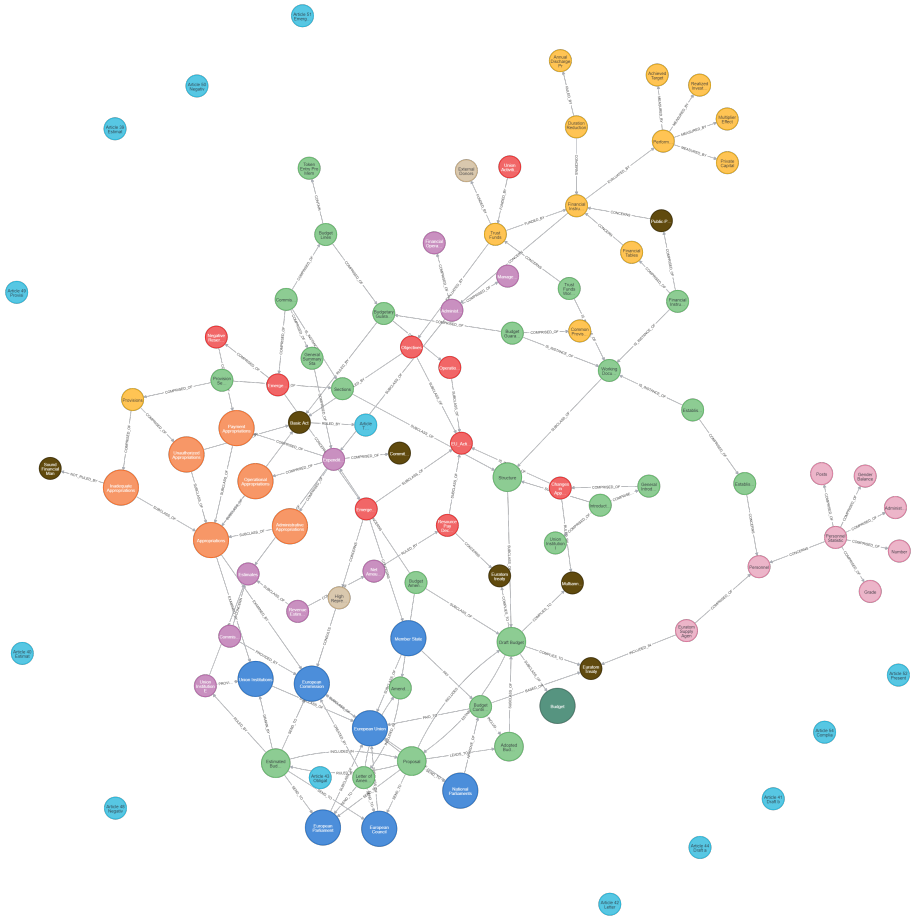


Figure 5.30: Knowledge Graph: Title III

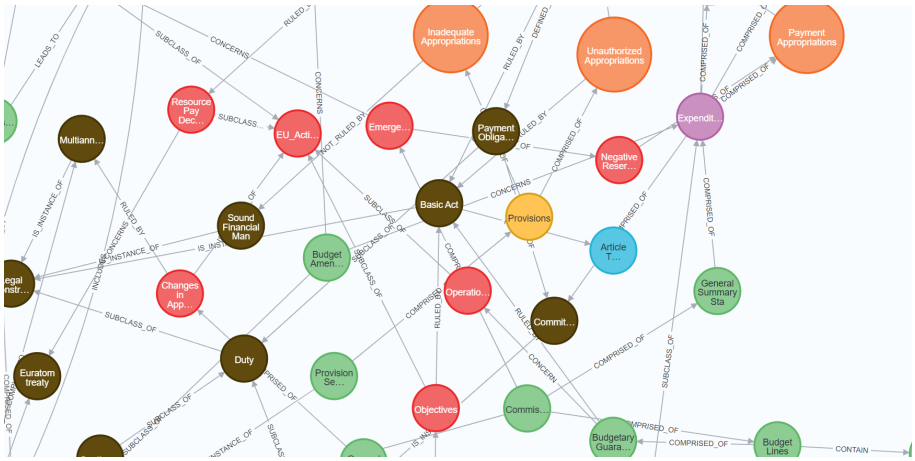


Figure 5.31: Inside the Graph

Figure 5.32: Node Properties

The hierarchy of the classes is visualized in the graph through the use of relations. In addition, instances of a class are also depicted in the graph in the form of relations. Instances according to (Noy and McGuinness), are actual realizations of a class. For example:

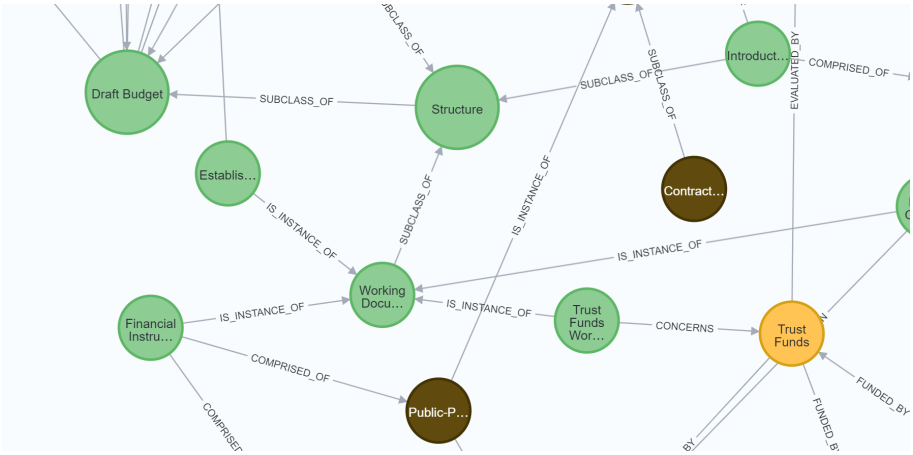


Figure 5.33: Class Hierarchy and Class Instances

From 5.33, the "Working Documents" class is a subclass of the "Structure" class, which in turn is a subclass of the "Draft Budget". At the same time, it is visible that the "Working Documents" class is realized through three instances of actual working documents found in the draft budget, namely, the "Establishment Plan Working Document", the "Financial Instrument Working Document", and the "Trust Funds Working Document".

The created Knowledge Graph, is based on a manual extraction of use cases from the Title III chapter of the Financial Regulations document. The schema of the graph is directly linked with the extracted classes from each use case and is not based on a predifined ontology. In order to add additional contextual information and structure to the graph, in the next section, the FIBO ontology is used to add additional semantic layers to the Knowledge graph.

5.6 Ontology Linking - FIBO

The Financial Industry Business Ontology (FIBO), is used to add additional layers of semantic information to the Title III Knowledge Graph. It should be underlined, that the ontology's role is this case, is not to define the overall schema of the graph, but instead has two main roles. **First** to act as an external knowledge that links certain entities of the graph to their textual and semantic meaning in the knowledge base. **Second**, to expand the graph by offering additional context, in the form of new layers of nodes in the graph.

From RDF to Neo4j - Neosemantics (n10s)

Resource Description Framework or RDF is a standard model, for data interchange on the Web, (W3C). It is a way for data exchange in the web. RDF datasets store data in the form of triplets, but without any enforced schema. The RDF datasets, comprised of triplets, can be translated into directed labeled property graphs. RDF datasets can be serialized in different formats, such as XML or Turtle.

FIBO includes various domains, with their corresponding ontologies. The domains can be explored using the FIBO Viewer (FIBO). Each domain contains the corresponding ontology. The ontology is stored using the RDF standard model. For example the European Government Entities and Jurisdictions Ontology, is stored in an RDF file, which contains the triplets in a XML format.

```
<owl:NamedIndividual rdf:about="&fibo-be-ge-euj;EuropeanUnionEntity">
        <rdf:type rdf:resource="&fibo-be-ge-ge;SupranationalEntity"/>
        <rdfs:label>European Union entity</rdfs:label>
        <rdfs:isDefinedBy rdf:resource="https://spec.edmcouncil.org/fibo/ontology/BE/GovernmentEntities,
        <rdfs:seeAlso rdf:resource="https://europa.eu/european-union/about-eu/countries_en"/>
        <skos:definition>individual representing the federated sovereignty and polity that is the Europ
        <fibo-be-ge-ge:hasSharedSovereigntyOver rdf:resource="&lcc-3166-1;Austria"/>
        <fibo-be-ge-ge:hasSharedSovereigntyOver rdf:resource="&lcc-3166-1;Belgium"/>
        <fibo-be-ge-ge:hasSharedSovereigntyOver rdf:resource="&lcc-3166-1;Bulgaria"/>
        <fibo-be-ge-ge:hasSharedSovereigntyOver rdf:resource="&lcc-3166-1;Croatia"/>
```

Figure 5.34: RDF Triplets Example - Government Entities and Jurisdictions Ontology

In 5.34, a triplet in XML format is described. The example, describes the {NamedIndividual:EuropeanUnionEntity} node. This node has a "type", "label", "isDefinedBy", "seeAlso" and "definition" properties, while it has a "hasSharedSovereigntyOver" relation with the "Austria", "Belgium", "Bulgaria" and "Croatia" nodes. In graph terms, this translates to a triplet of a node with label="European Union Entity" and the above properties, that has a directed relation of type="hasSharedSovereigntyOver" to four country nodes.

In regard to the Title III Knowledge graph, the following RDF stored ontologies, were utilized:

- European Government Entities and Jurisdictions Ontology[1]
- Legal Capacity Ontology [2]
- Financial Instruments Ontology [3]
- Payments and Schedules Ontology [4]

The RDF datasets, had to be converted into graphs and imported into the Neo4j Graph Database. Firstly the RDF datasets, were manually adjusted to remove any redudant information from the triplets. Secondly, Neo4j's Neosemantics extension (Neo4j Graph Data Platform), and the corresponding guides (**?**), were used to import the RDF datasets into Neo4j. For example after the ontology "European Government Entities and Jurisdictions Ontology", is imported into Neo4j Graph Database, it can be visualized as such:

[1] https://spec.edmcouncil.org/fibo/ontology/BE/GovernmentEntities/EuropeanJurisdiction/EUGovernmentEntitiesAndJurisdictions/

[2] https://spec.edmcouncil.org/fibo/ontology/FND/Law/LegalCapacity/

[3] https://spec.edmcouncil.org/fibo/ontology/FBC/FinancialInstruments/FinancialInstruments/

[4] https://spec.edmcouncil.org/fibo/ontology/FND/ProductsAndServices/PaymentsAndSchedules/

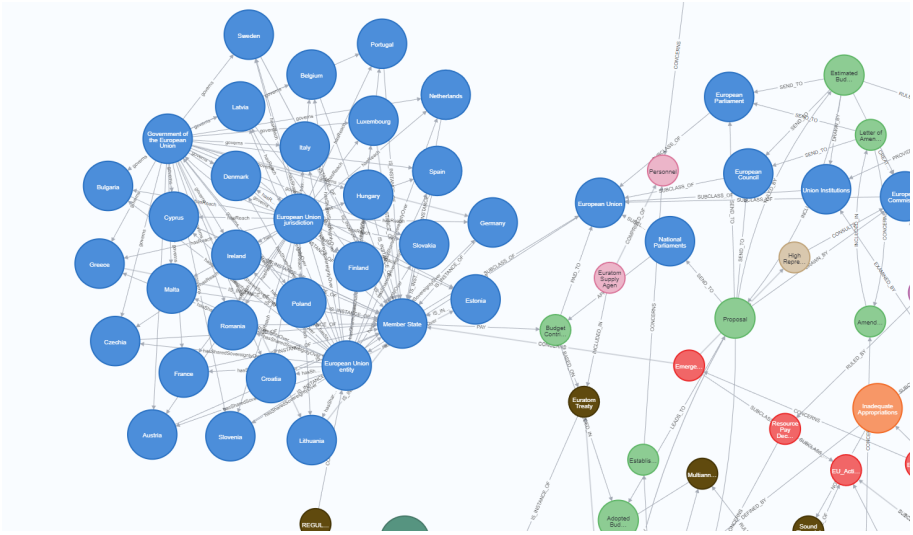


Figure 5.35: European Government Entities and Jurisdictions Ontology - Graph

Through the use of Neo4j's cypher query language, the ontology graphs of each domain, are connected to the 5.30, thus resulting to **additional semantic layer** in the graph, providing context for certain entities, such as for the financial instruments, included in the European Unions' Budget, and for the entities related to the European Union, such as for the countries and government institutions.

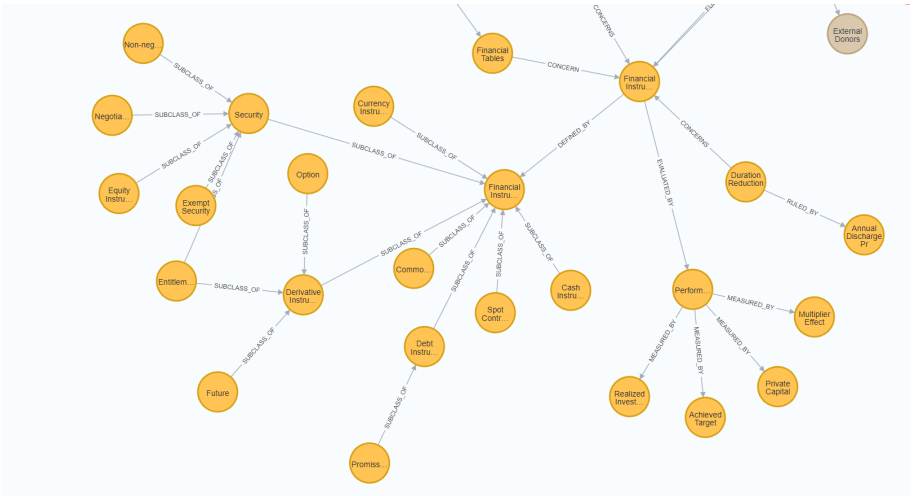


Figure 5.36: Financial Instruments Ontology - Graph

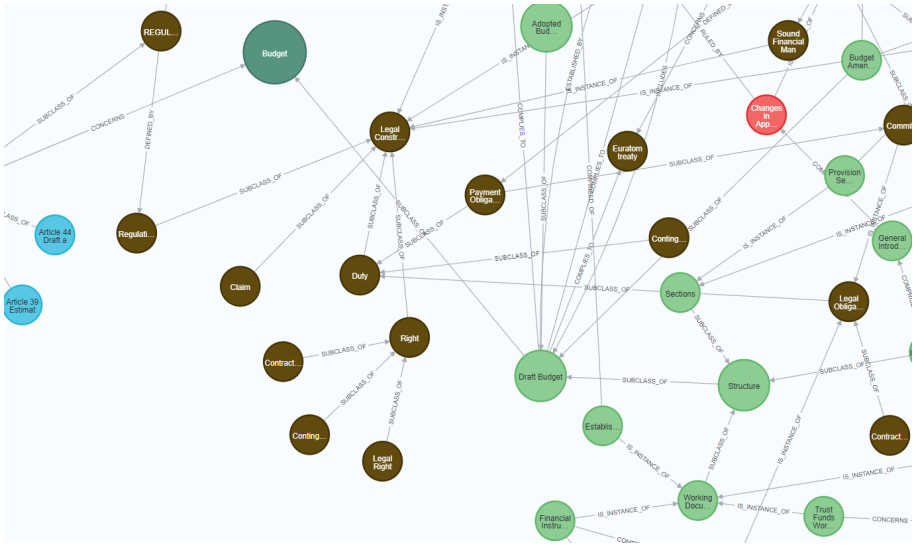
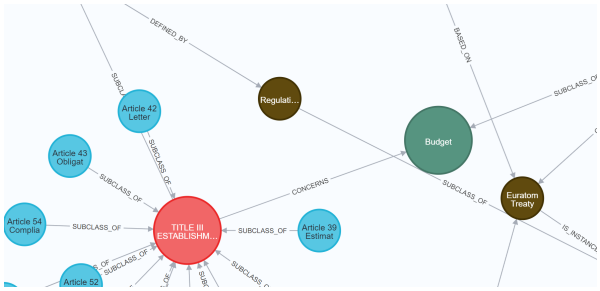


Figure 5.37: Legal Capacity Ontology - Graph

Additionally, to the expansion of the knowledge graph, FIBO provides entity disambiguation capabilities, through **entity linking (EL)**, by providing a direct link of a certain entity to each textual meaning in the knowledge base. For example, the within the properties of the European Government Node, the reader can follow the link and gain access to the official meaning of the European Government as described in the knowledge base of FIBO:

Figure 5.38: Knowledge Graph - Regulation Entity



(a) Regulation Entity Node

name	Regulation
uri	https://spec.edmcouncil.org/fibo/ontology/FND/Law/LegalCapacity/Regulation

(b) Regulation Properties - FIBO URL

The link, 5.38b ,found in the Regulation Entity, belonging to the Legal Construct Class, `https://spec.edmcouncil.org/fibo/ontology/FND/Law/LegalCapacity/Regulation`, leads to the FIBO knowledge base and provides the meaning of the entity as well as the domain on which it corresponds. All the entity nodes that are part of the FIBO domain ontologies, contain a url link to the corresponding meaning of the entity.

5.7 Final Knowledge Graph

After the FIBO ontologies are added to the knowledge graph, the final knowledge graph for Title III: Establishment and Structure of the Budget is completed.

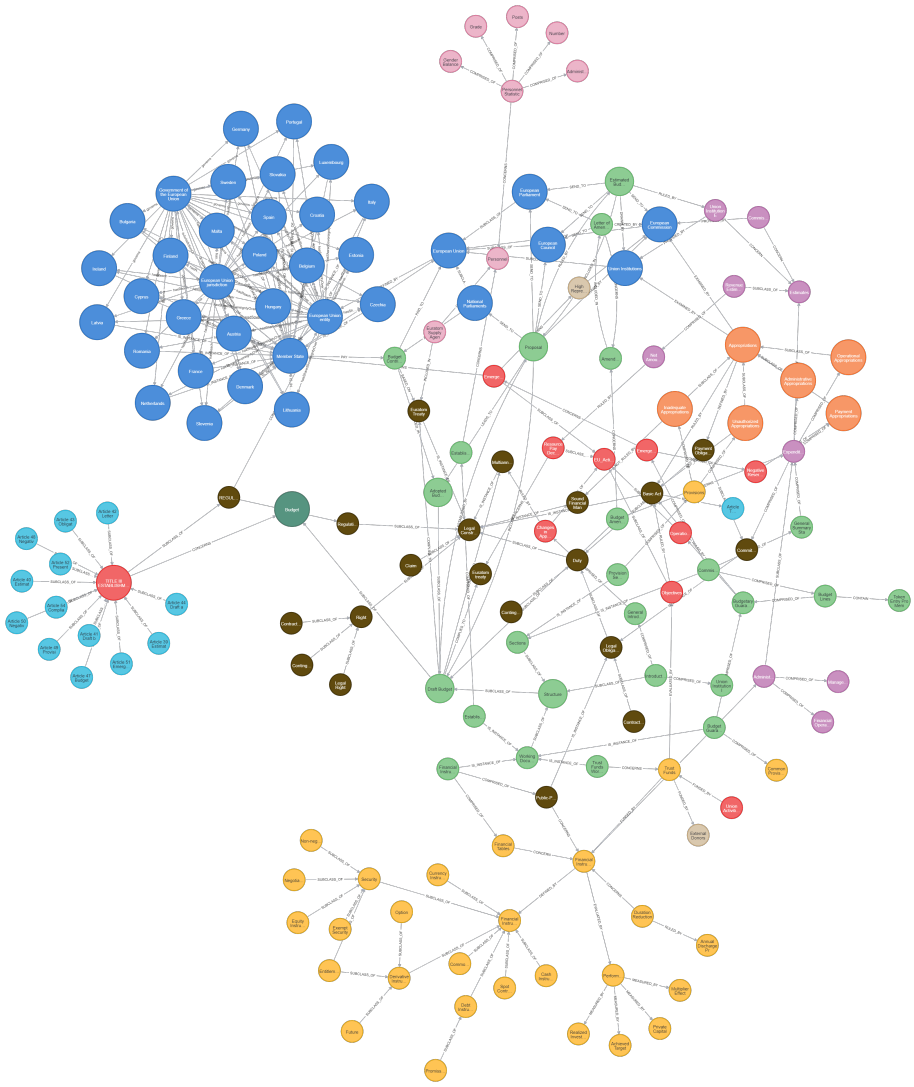


Figure 5.39: Final Title III Knowledge Graph

The knowledge graph's main purpose is to be able to answer use case questions related to the establishment and structure of the budget. The following are the main use cases on which use case questions can be derived.

1. Budget Establishment Pipeline within the European Union - Key Players and Responsibilities
2. Budget Amendment Procedure
3. Main Components of the Draft Budget's Structure
4. Role of Provisions within the Budget
5. Role of Reserves within the Budget
6. Presentation of the Budget
7. Emergency Aid towards European countries in need

The evaluation of the knowledge graph, lies in its practicality. A knowledge graph is considered successful if it manages to offer answer to domain specific use case questions. An expert of financial regulations, or any interested user, could query the knowledge graph through the use of the Neo4j Graph Database system, in order to find information regarding the establishment of the budget.

Implementation

The following link, is a path to the GitHub repository, that contains the source code, on which the implementation of the current chapter, 5, is achieved:

GitHub Link: `https://github.com/Yiannis7918/Final_Thesis_KG_Construction.git`

6 Graph Analytics & Use Case Extraction

Konstantinos Panagiotis Apostolou

Graph analytics entail the use of algorithms and methodologies to examine the links and structures inherent in graph data. We, as humans, have the innate ability to recognize patterns, whereas computers lack the reasoning to do so; therefore, in the context of knowledge graphs, which depict information as a network of entities and their relationships, graph algorithms can reveal patterns, trends, and insights that are not readily apparent through conventional data analysis techniques. Utilizing algorithms such as centrality analysis, community detection, and connectivity analysis, data scientists can discover the most important nodes or edges, identify groups of nodes that form clusters inside the graph, or measure the graph's resilience to information dissipation in case links were severed.

The use of graph algorithms aids the user in comprehending the fundamental structure and dynamics of the knowledge graph, resulting in enhanced decision-making and the discovery of new insights. In our case, we explored the notion of exploiting graph analytics as a means to identify possible use cases for our data graph. The idea is that by uncovering the latent characteristics of the data, the original large network (6.1) will be summarised and more manageable. We started from Chapter III which is one of the longer ones, but the framework can be applied to the other chapters.

Firstly, centrality analysis algorithms are used to find the most critically positioned nodes. The conventional centrality indexes like Degree (DC), Closeness (CC) and Betweenness (BC) are based on the topological characteristics of the network (16), whereas Eigenvector centrality and its successors, such as HITS and PageRank, measure the transitive influence of nodes (17). Studies have shown that the traditional centrality measures are correlated (47) and since the aim is to find what questions our data graph is able to answer we want to find the nodes that connect different communities, therefore we will keep Betweeness out of the three.

Eigenvector centrality, HITS and Pagerank are all based on the adjacency matrix of the graph and their differences lie in the assumptions. Eigenvector centrality measures the centrality of each node as a function of the centralities of the neighbors. Mathematically this is modelled as a linear algebra equation regarding the eigenvalue matrix of the adjacency matrix (17) and requires that the graph is strongly connected. Since we have a directed graph, Eigenvector centrality is unsuitable but we will include it for comparison reasons.

Pagerank (9) tries to measure the relative importance of each node in the network by introducing a rank to each node and works on the premise that a node is important if it linked to other important nodes. In essence, it imposes a probability distribution on the node set of the data graph, therefore the rank of each node conveys the chance a random walk has to reach the particular node. Pagerank achieves a balanced representation of node importance, making it efficient and effective for general use.

On the other hand, HITS (Hyperlink - Induced Topic Search) (23) produces two indexes that split the nodes into two groups, *hubs* which are nodes that have a lot of outgoing connections and *authorites* which are nodes that receive a lot of links. The core concept is that authoritative nodes $\mathcal{A} = \{n \in \mathcal{V} | n : authority\}$ hold the largest amount of the desired knowledge and logically, these nodes will have a considerable amount of incoming relations, while the nodes that point towards the authorities work as directories or hubs that control the flow of information.

6.1 Centrality Analysis

Our rationale is to use centrality as a filter in order to obtain a sub-graph that holds the essence of the information Chapter III holds. To this end, starting from the preliminary graph (version B) for Chapter III, we computed the aforementioned metrics and their Speraman's correlation matrix 6.1 in order to understand the graph's anatomy. We reach the following conclusions:

- The traditional centrality metrics BC, DC, CC and the Hub Scores are highly positively correlated.

- The eigenvalue centrality and the Authority scores are highly positively correlated and negatively correlated with the other metrics except Pagerank.
- The Authority and Hub scores are highest for tail and head nodes respectively.
- The Pagerank index does not favor any type of nodes, illustrating it's generalist nature.

Table 6.1: Correlation Table

	Centrality Measures' Correlation Matrix						
	BC	CC	DC	EigenC	HitsAuth	HitsHub	PageRank
BC	1.00	.	.	.	.	.	.
CC	0.69	1.00	.	.	.	.	.
DC	0.83	0.35	1.000	.	.	.	.
EigenC	-0.55	-0.30	-0.760	1.00	.	.	.
HitsAuth	-0.40	-0.19	-0.610	0.72	1.00	.	.
HitsHub	0.84	0.38	0.990	-0.76	-0.61	1.000	.
PageRank	0.22	0.51	-0.073	0.24	-0.22	-0.057	1.000

Due to the dual scores produced by HITS and the fact that it is able to capture the importance of heads and tails independently and collaboratively at the same time, we used this metric to filter the data graph. We started with the fifteen most important head and tail nodes and their originating articles and after discarding some nodes that clearly are mistakes of the text mining process, we acquire a subgraph for Chapter 3 (Figure 6.2).

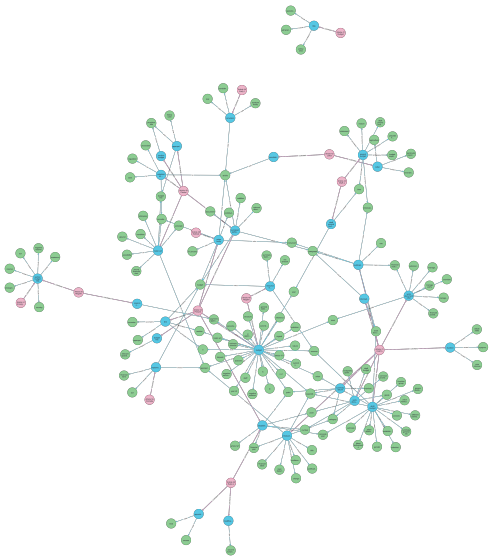


Figure 6.1: Full Graph

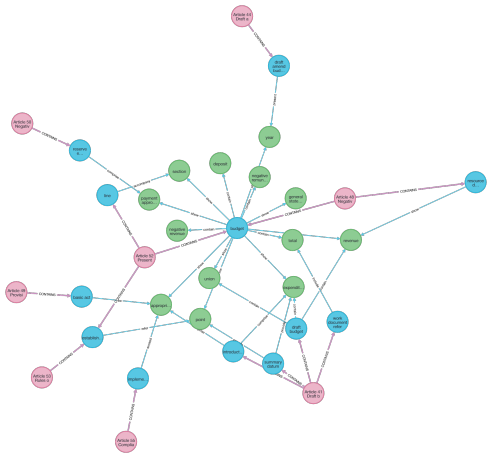


Figure 6.2: Sub Graph

Having obtained a sub-graph that is more manageable compared to the full graph of Chapter III, we have at hand the task of categorizing the nodes regarding the information they convey. The idea behind this decision is that by clustering the nodes, we will be able to discern possible topics that the data graph can provide answers for.

6.2 Community Detection

In contrast to centrality analysis, community detection revolves around data analysis methods used to discover clusters of nodes where the links between them are stronger compared to nodes outside the cluster. These techniques help us better grasp the structure of the data and might lead us to uncover latent information. It is worth mentioning that the aforementioned structure is inherent in the network, therefore the methods are unsupervised, and the number and size of the communities is not under our control.

There are four schools of thought to tackle the task of community detection, modularity optimisation methods, deep learning techniques. probabilistic models and spectral clustering algorithms. To discover the community structure of the graph, the modularity-based Louvain algorithm (8) was used due to its efficacy on the task and its similarity to hierarchical clustering.

Modularity (32) is a measure that quantifies the strength of a graph division into communities and can be used as an identifier to whether a network can be decomposed clusters. Modularity conveys whether the structure observed in the graph is similar to a randomly connecting the nodes or an architecture lies within. Thus, given a graph $\mathcal{G} = \{(v, e)|v \in \mathcal{V}, e \in \mathcal{E}\}$ modularity $\mathcal{Q}$ is proportional to the number of edges e falling within clusters minus the expected number of edges in a random graph. Mathematically, this can be formulated using the adjacency matrix of the graph and some know results of graph theory. Imagine that we have a graph with n nodes and we dichotomize it into two groups. Then for every node n_i we have $s_i = 1 \vee s_i = -1$ if the node belongs into cluster 1 or 2 respectively. It's important to remember that every element A_{ij} of the adjacency matrix shows the number of edges linking nodes i and j, also the total number of edges in the graph is $m = \frac{1}{2}\sum_i k_i$ where k_i is the degree of node i and lastly, in case of a random graph the expected count of edges connecting nodes i and j is $\frac{k_i \dot{k}_j}{2 \times m}$. Then, the modularity index of graph $\mathcal{G}$ is given by:

$$Q = \sum_{ij} \left(A_{ij} - \frac{k_i \dot{k}_j}{2 \times m}\right) \tag{6.1}$$

for $s_i = s_j$

Louvain works in two steps to optimise the aforementioned modularity. Initially, each node is allocated to a distinct cluster and then iteretivelly the nodes are moved through clusters trying to maximise modularity. The first step concludes when no futher optimization is possible. During the second step, a new reduced network is formed by aggregating nodes allocated in the same community during the first phase and then the first phase starts again. Through the iteration of the two steps, the algorithm constructs a hierarchy of communities, with each level signifying a distinct resolution of the network's community structure. The process is visualized in figures 6.3, 6.4, 6.5.

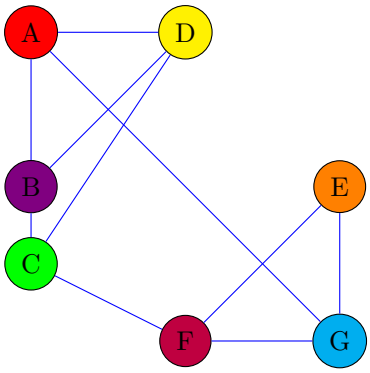


Figure 6.3: Step 1 - Every node is allocated into its own cluster

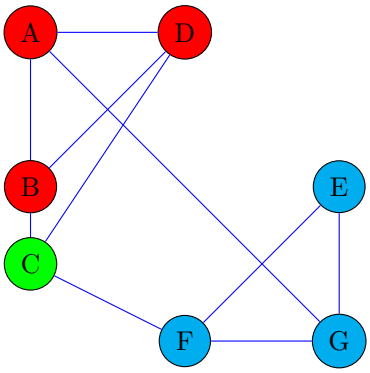


Figure 6.4: Step 2 - Nodes are assigned to clusters that maximize the modularity

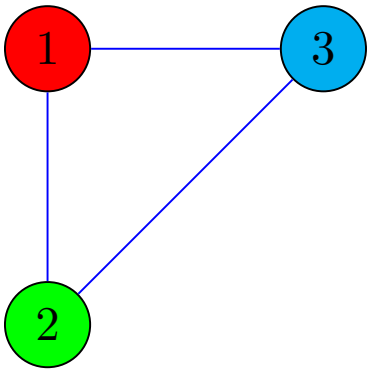


Figure 6.5: Step 3 - Clusters are reduced to a single node

Applying the Louvain method to sub graph of Chapter III results in a clustering solution as seen in figures

6.6 through 6.10, with an achieved modularity score of 0.4. The identified communities can be characterized by their contents. The yellow community pertains to the draft budget submission, financial planning for subsequent years based on the budget, and regulations concerning the establishment plans for personnel and temporary reserves. The red community possesses knowledge about to the document's template. The purple cluster has to do with budget amendments, the cyan cluster addresses negative reserves within the budget, and the orange community outlines the conditions under which negative revenue may be recorded in the budget.

Figure 6.6: Community A

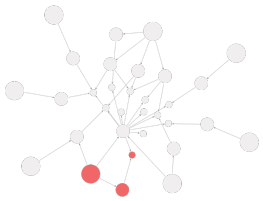


Figure 6.7: Community B

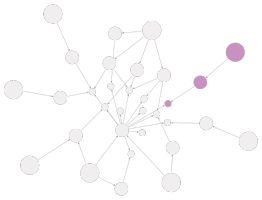


Figure 6.8: Community C

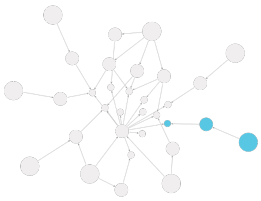


Figure 6.9: Community D

Figure 6.10: Community E

6.3 Use Case Extraction

To sum up, knowledge graph construction is a twofold task that starts with th creation of the data graph which contains the extracted information from the raw text and then the connection to an ontology that will connect the terms to real entities. We use the FIBO ontology which is very expansive. Our method is to use preliminary graphs in order to uncover use cases for our knowledge graph that will allow us to filter the FIBO ontology to make it more compatible with our data graph. In this section, it was explored how graph analytics can be used to complete this task and the methods described can be generalised to other chapters of the regulations. Here we present a possible use case, following the template presented in (37).

Use Case 1: In-House Search Engine
Description: Jonas, employed by an EU organization, is advised by his supervisor that additional personnel is necessary due to a new project. The supervisor delegates to Jonas the responsibilities of composing a paper that initially informs the relevant parties, subsequently presents the proposed budget for the institution, and ultimately includes the required attachments.
Actors: EU employee, Application
Flow: Jonas accesses the program, proceeds to the budget section, chooses the amend option, and finally picks "change in staff appointments" from a drop-down menu. The application gathers data and detects related articles from the regulations. Subsequently, it categorizes the returned articles according to the components of the query and displays them on the screen for Jonas. Additionally, data tags accompany each article to encapsulate the information presented inside the page. Jonas uses them to identify the necessary recipients for his document, the deadline, and the supporting documents he must attach.

Reproducability All the code used to generate the results discussed in this section can be found on "https://github.com/KonApos/KonApos/tree/main". For the correlation analysis R programming language was used. The Neo4j (version 4.4.38) graph database and Cypher were used for building the data graph and applying the graph algorithms.

7 Knowledge Graph Construction: REBEL

Ioannis Dikaioulias & Konstantinos Apostolou

7.1 REBEL Summary

Konstantinos Apostolou

In order to construct a knowledge graph, we need to extract the relational information contained in the raw text. Ideally, we would need to acquire a characterization for every word in the text that assigns a label from a predefined set. In other words, we need to complete the tasks of Named Entity Recognition and Relationship Extraction, in order to obtain the necessary triplets that will be fed into the graph model.

One of the ways to complete these tasks is to use a pre trained model that will receive the raw text as input and provide the triplets as output. In the following section the **REBEL** (20) model will be described.

With REBEL, relationship extraction (RE) is seen as a generation task, where given the raw text as input the model returns the entity relation triplets. The underlying model, *BART-large* is an autoregressive transformer, with an bidirectional encoder and left-to-right autoregressive decoder as architecture (25). BART is follows the architecture described in (41), but uses GELU instead of RELU for the activation function of the feed-forward network sub-layer. Alongside the transformer model, a linearization technique is used on the triplets in order to make the decoding more efficient. To linearize the triplets, new tokens are used that label each word and make it easier for the decoder. The token *<triplet>* is used when a new triplet with a new head entity is introduced, *<subj>* denotes the end of te head entity and the start of the tail entity and *<obj>* marks the start the relation between head and tail.

Therefore, if x is the input text and y the triplets, then the learning task concretely stated is:

$$P_{BART}(y|x) = \prod_{i=1}^{len(y)} P_{BART}(y_i|y_{<i}, x) \tag{7.1}$$

where BART is fine tuned for sequence generation using an already labeled (RE) dataset.

7.2 REBEL Architecture - Transformer Model

Konstantinos Apostolou

For numerous years, sequence modeling and creation were accomplished utilizing standard recurrent neural networks (RNNs). Theoretically, information from a single token can disseminate extensively across the sequence; however, the vanishing-gradient problem results in the model's state at the conclusion of a lengthy phrase lacking exact, retrievable knowledge from prior tokens.

A significant advancement was LSTM, a recurrent neural network that employed numerous improvements to address the vanishing gradient problem, facilitating effective learning of long-sequence modeling. A significant breakthrough was the implementation of an attention mechanism that utilized neurons to multiply the outputs of other neurons, referred to as multiplicative units. LSTM established itself as the standard architecture for long sequence modeling until the publication of Transformers in 2017. Nonetheless, LSTM continues to employ sequential processing, similar to the majority of other RNNs. RNNs function sequentially, processing one token at a time from beginning to end, but they are unable to do parallel processing across all tokens in a sequence.

Currently, attention methods are essential in sequence-to-sequence modeling applications since they facilitate the modeling of relationships with arbitrary places inside the input or output sequences. The Transformer model (41) relies exclusively on self-attention to model the dependency structure of input and output.

The Transformer model architecture combines an encoder (left on figure 7.1) with a decoder structure (right on figure 7.1). Each of these contains additional substructures, namely the attention mechanism and

a feed-forward multilayer perceptron. Additionally, embeddings are utilized to convert the symbolic input to vectors on the continuous model space. It's worth noting that the model produces each output symbol in an autoregressive manner, meaning that the previously generated output belongs in the information space for each prediction. Next we will elaborate on the key elements of the Transformer architecture.

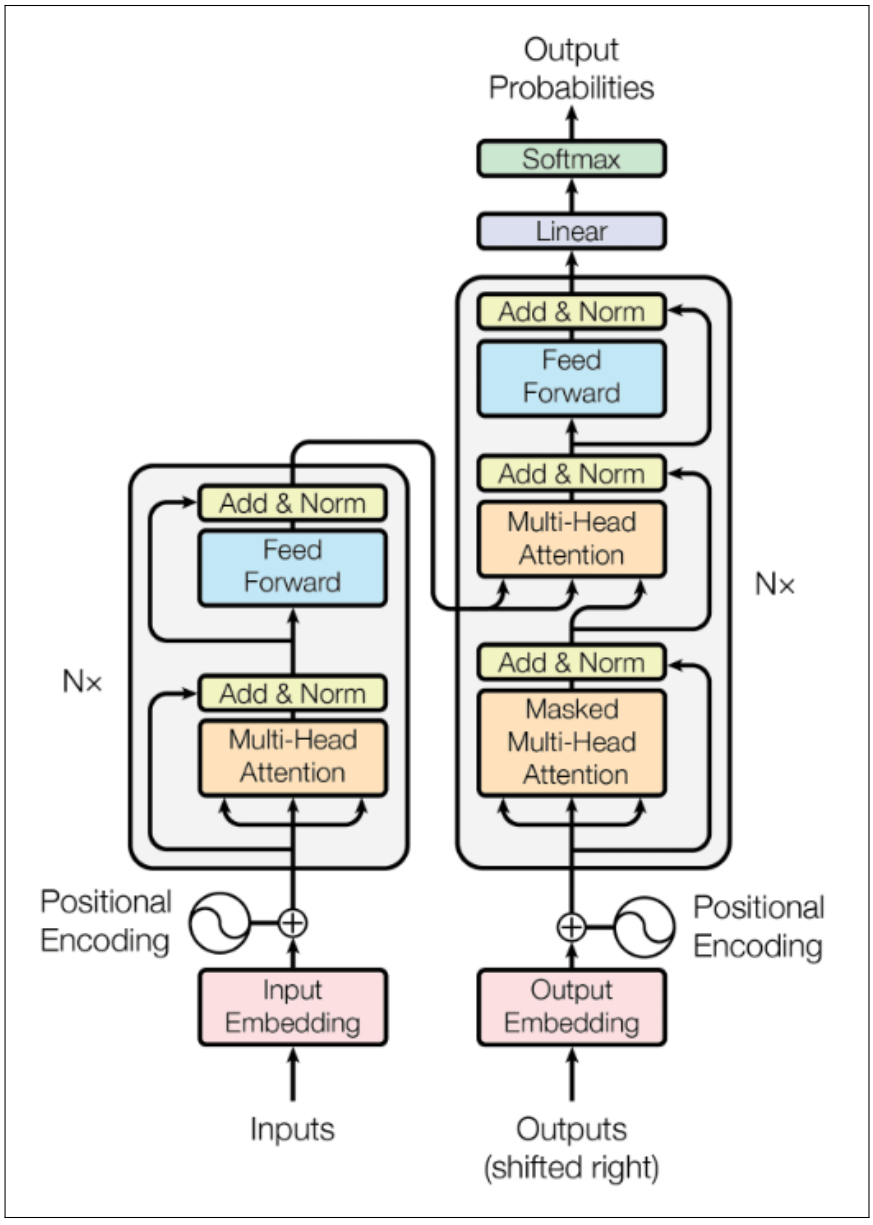


Figure 7.1: The Transformer Architecture (41)

Embeddings

The data interact with two embeddings in the model. First is the input embedding where the tokenized input sequence in transformed to vectors of a continuous space of dimension d_{model}. This is done by the matrix multiplication of a one-hot encoded vector for each token and a weight matrix M. The penultimate

step of the Transformer is a linear and softmax function composition layer called unembedding where the predicted output vector $\hat{v}$ is converted into a probability distribution on the token space. Mathematically we have, $Embed(x) = [0, \ldots, 1, \ldots, 0] \cdot M$ and $Unembed(x) = Softmax(M \cdot x + b)$. In this work the input embeddings produce outputs of dimension $d_{model} = 512$.

Positional Encoding

A positional encoding is a fixed-size vector that represents the relative positions of tokens in a sequence, supplying the transformer model with information regarding the location of words inside the input sequence. This will introduce a bias about the order of the input sequence. The functions used in the original paper are:

$$f(t)_{2k} = sin(\theta), \theta = \frac{t}{N^{\frac{2k}{d_model}}}$$

$$f(t)_{2k+1} = pos(\theta), \theta = \frac{t}{N^{\frac{2k}{d_model}}}$$

with $\forall k \in \{0, 1, \ldots, \frac{d_{model}}{2} - 1\}$

Attention Mechanism

Attention is a method to measure the importance of each component of a sequence in relation to other components. In a Transformer we encounter the scaled-dot product attention split into heads. The base of the attention function is a triplet of *query*, *key* and *value* vectors. In this context, queries and keys are different representations of the input ssequence, and a value is the output of a vector function composition on them. The model has to learn three weight matrices, W^Q, W^K, W^V respectively, and with them each vector x_i of the input sequence is expressed as a $q_i = W_Q \cdot x_i$ query, key or value vector in the same fashion. Stacking these vectors, we obtain the query, key, and value matrices Q, K, V.

The attention matrix is given by

$$Attention(Q, K, V) = Softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V$$

The Transformer model innovated by calculating the attention matrix over multiple different projections of the model space into smaller sub-spaces, called heads. This method, called multi-head attention, enables the model to concurrently focus on input from many representation sub-spaces across multiple positions. This comes at no additional computational cost compared to calculating attention on a single head with full dimensionality.

Mathematically, we have the multi-head attention function by joining the head attentions row-wise and projecting them into the model space.

$$MultiHead(Q, K, V) = Concat_{i \in n_{heads}}(Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V)) \cdot W_O \tag{7.2}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{n_{heads} \cdot d_v \times d_{model}}$.

It is worth noting that in the decoder layer the first sub layered attention mechanism needs to be adjusted. The Transformer model needs to simulate an auto-regressive behaviour, therefore the computation for token t_i needs to be able to access the output for tokens $t_j, j \leq i$. To ensure this a masked matrix M is introduced in the Attention function, which for now becomes:

$$MaskedAttention(Q, K, V) = Softmax(M + \frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V$$

where M is a matrix with $M_{ij} = 0$ for $i \geq j$ and $-\infty$ otherwise.

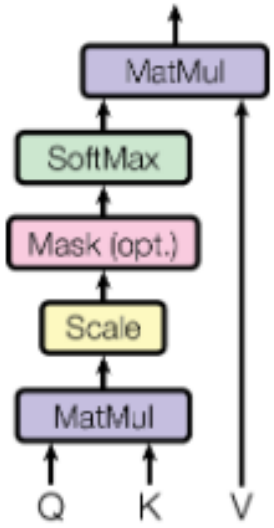


Figure 7.2: Dot-Product Attention (41)

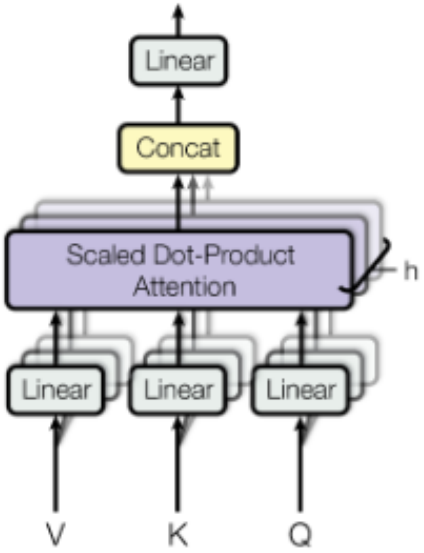


Figure 7.3: Multi-Head Attention (41)

Feed-Forward Network

Each layer concludes with a 2-multilayer perceptron model that is applied to each vector separately. The original model was introduced with the choice of activation function ϕ being RELU but in other implementations, others have been used, such as GELU for (25). Concretely we have,

$$FFN(x) = \phi(x \cdot W_1 + b_1) \cdot W_2 + b_2$$

Encoded - Decoder

All of the components previously described are combined to form a single layer. The Transformer has an encoder and a decoder with $N = 6$ stacked layers each. In other words, the first encoder layer receives the sequence of input vectors from the embedding layer, generating a sequence of vectors. Then, the second encoder processes this series of vectors, and so forth. The output from the ultimate encoder layer is subsequently utilized by the decoder. In particular to the decoder stack, the Mask Self Attention sub-layer is added in order to ban positions from attaining preceding positions of the input sequence, whereas encoder processes the complete input simultaneously, allowing each token to attend to every other token (cross-attention), thereby eliminating the necessity for masking. Also, even though they are not needed in theory, a normalization and residual connection sub-layer are included after each of the primary sub-layers on the grounds of numerical stability.

The number of layers is a parameter that can be optimised according to the needs of the application. For instance, in (25) they use a 12 layer transformer for *BART-large*.

Mathematically, we have the encoder and decoder layers:

$$EncoderLayer(X) = FFN(Attention(X))$$

with X the 1st encoder layer input. We write Z the output of the endoder stack. Then:

$$DecoderLayer(X) = FFN(Attention(MaskedAttention(X); Z))$$

7.3 REBEL Implementation - Title III

Ioannis Dikaioulias

Triplets Preprocessing

The REBEL model applied for openIE in this thesis, can be found in (21). The model was fitted on the text of each article's content, after coreference resolution, as described in 5.2, was applied, and after punctuation, `!@#$%^&*_-+=[];"\'|<>,?/~``‘`, was removed, from the text of each article.
THE REBEL pre-trained model was fitted with the following parameters:

Parameter	Value
Model Name	Babelscape/rebel-large
Max Length	256
Length Penalty	0
Number of Beams	20
Number of Return Sequences	3

The following preprocessing pipeline was applied on the extracted REBEL triplets:

- Triplets were lowercased and stopwords were removed from the Head and Tail elements.
- Punctuation was removed from each Head and Tail element to remove any redudant information.
- Lemmatization was applied to each Head and Tail to reduce each Head and Tail to each original lemma.
- Duplicate Triplets, meaning triplets with exactly equal Head-Relation-Tail elements, were removed.
- Triplets that have the exact same Head and Tail, Head=Tail, were removed as they offer no semantic information.

The following Relations were identified by REBEL:

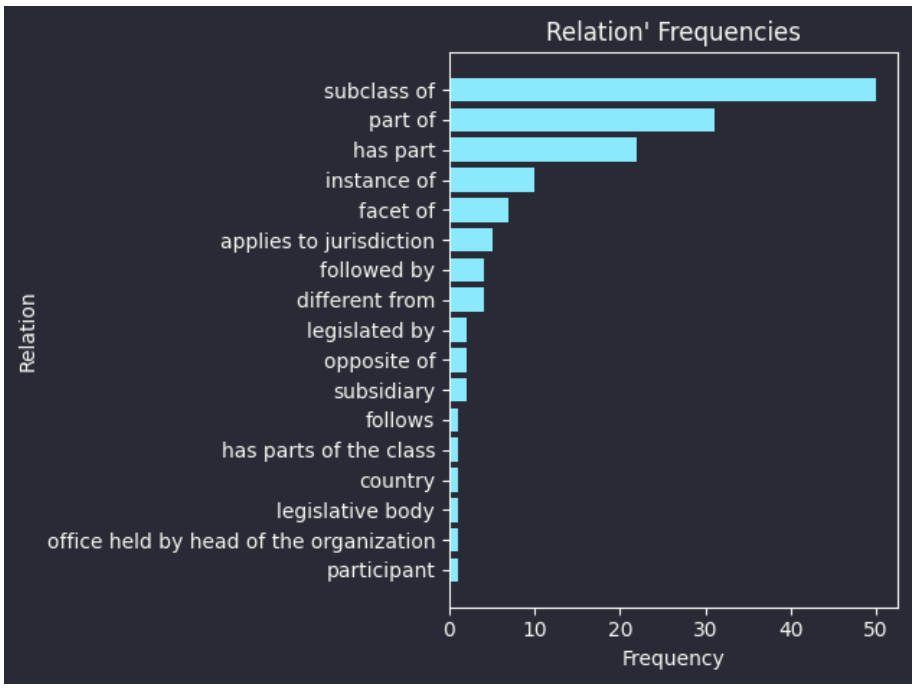


Figure 7.4: Extracted REBEL Relations

It can be observed from 7.4, that most identified Relations, appear in a frequency close to 0. Moreover, certain relations, such as "part of", "has part", "subclass of", and "has parts of the class" convey a similar meaning and would be better merged into one relation. Furthermore, the entities extracted by REBEL, were manually explored in terms of their meaning and certain group of entities were either replaced or merged. The tables below illustrate these adjuments:

Table 7.1: Manual Merge of Certain Extracted Entities

Entities to Merge	Merged Entity
council council, council the council, council, council union institution	european council
institution, body	union institution
european, parliament	european parliament
publicprivate partnership	public private partnership
union basic act	basic act
union, trade union	european union
operating charge	financial operating charge
discharge procedure, discharge, annual discharge	annual discharge procedure
solidarity fund	european union solidarity fund
parttime	part time work
humanitarian aid crisis response	crisis response
expenditure estimate, agriculture	expenditure estimate agriculture
supply agency, euratom	euratom supply agency
leverage	leverage effect
commission, commission type, commission administration	european commission
tfeu, article 314 tfeu	euratom treaty
instrument	financial instrument
table	table information
trust fund	union trust fund
country	member state
payment	payment appropriations
amend, draft	amend budget
provision title	provision
policy area	expenditure policy area
act	union act

Table 7.2: Manual Merge of Certain Extracted Relations

Relations to Merge	Merged Relation
part of, subsidiary, has part, facet of, has parts of the class, participant	subclass of
opposite of	different from

Title III Knowledge Graph - Neo4j

After the applied pre-processing pipeline of the previous section, the triplets were stored in the Neo4j Graph Database and visualized:

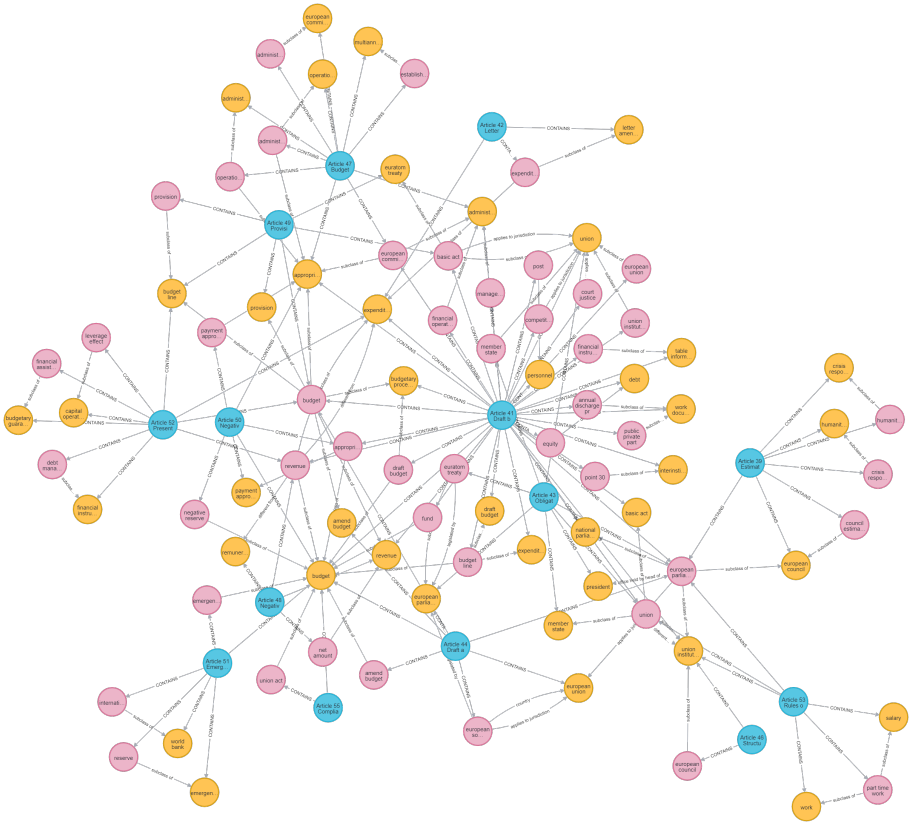
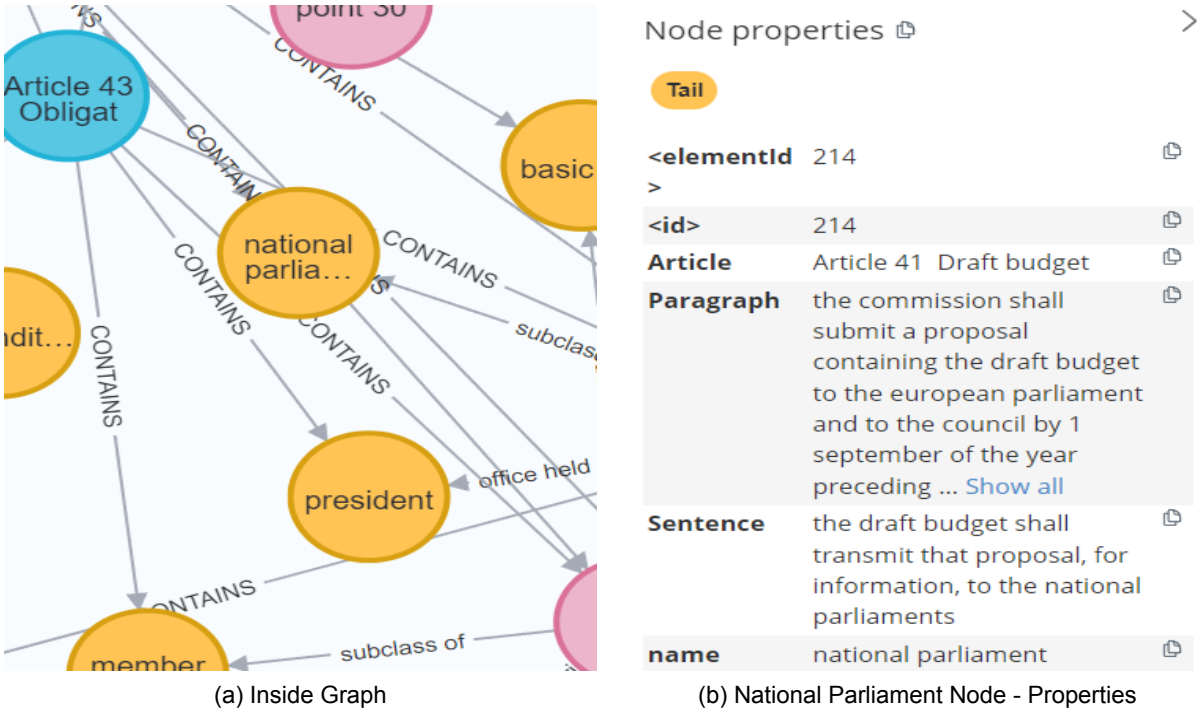


Figure 7.5: Title III: REBEL Knowledge Graph

Figure 7.6: Inside the REBEL Knowledge Graph



(a) Inside Graph

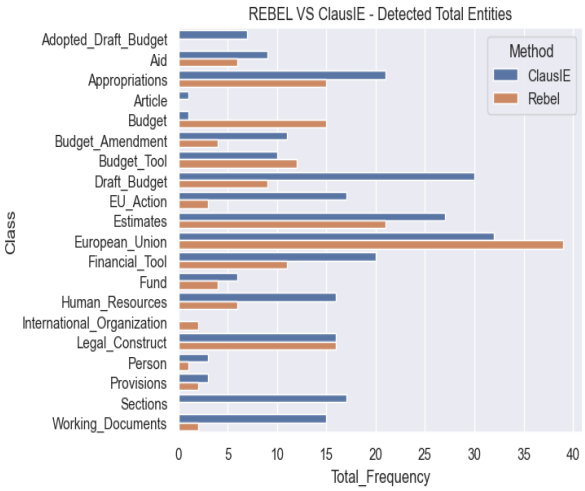
(b) National Parliament Node - Properties

The knowledge graph produced by REBEL, and with minimal preprocessing, resembles more of a preliminary graph than an actual knowledge graph. Manual manipulation of the graph is further required. Human intervention, and more importantly a domain expert's intervention would be necessary, to derive an ontology-based schema of the knowledge graph. More specifically, to derive, the use cases to be extracted, and to identify the most important classes, sub-classes, entities and relations. Furthermore, additional ontology layers would be necessary to expand the semantic layers of the graph and offer more context and meaning. The goal would be to construct a knowledge graph, similar to 5.39.

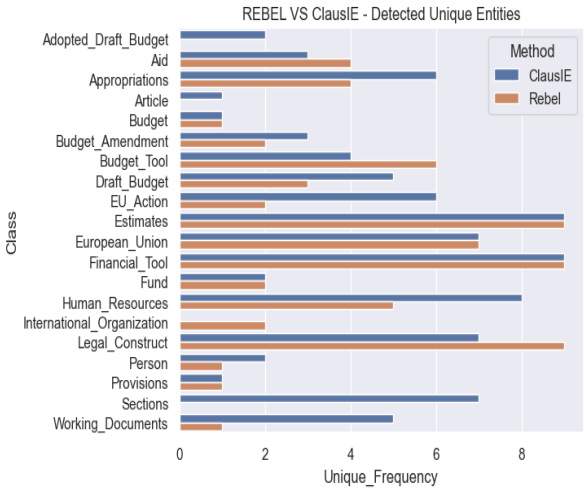
REBEL VS ClausIE

The role of the domain expert was simulated. The triplets of 7.5 were manually explored and entities were assigned to classes. The classes are the same classes used by the schema of the 5.39 final knowledge graph. The goal is compare REBEL with the knowledge graph produced in the 5 section. The clausIE based, final knowledge graph, was used as the ground truth, in order to evaluate REBEL, in terms of it's to capture use case related entities. The following graphs were produced to compare the two knowledge graphs:

Figure 7.7: REBEL VS ClausIE - Captured Entities per Class



(a) REBEL VS ClausIE Total Detected Entities



(b) REBEL VS ClausIE Unique Detected Entities

From the above plots, REBEL seems to be able capture a high amount of information in the text, even capturing entities belonging to classes which were not captured by clausIE, such as the "International_ Organization" class. REBEL appears to be able to capture a high amount of entities, especially for the "European_Union", "Estimates" and "Legal Construct" classes but captures zero to few entities belonging classes such as "Adopted_Draft_Budget", "Sections", and "Working_Documents". In conclusion, even without any training, the pre-trained REBEL model manages to capture a high amount of information in terms of captured entities. Nevertheless, as in the clausIE knowledge graph, human intervention is still critical, in order to define the use case classes, design an ontology-based schema and add relevant ontology layers as additional semantic information. The main advantage of the REBEL model is that, given an labeled domain-specific dataset, it can be fine-tuned, to achieve high performance in use case specific prediction of entities and relations.

Implementation

The following link, is a path to the GitHub repository, that contains the source code, on which the implementation of the current chapter, 7.3, is achieved:

GitHub Link: `https://github.com/Yiannis7918/Final_Thesis_KG_Construction.git`

8 Knowledge Graphs vs. Large Language Models

Konstantinos Panagiotis Apostolou

Large language models have significantly impacted the technological and academic sectors in recent years, owing to their exceptional performance in different natural language processing tasks, including text generation and natural language interpretation. This comes as a result of training these models on enormous corpora of publicly available text. In spite of their merits, LLMs are black-box models that store knowledge in their parameters, and as a result, their answers lack explainability. Additionally, these models are susceptible to hallucinating (45), which is to provide answers that are factually inaccurate and, as a consequence, significantly detriment the credibility of the model. Last but not least, they lack real-time knowledge updates and require retraining frequently.

On the other hand, knowledge graphs store verifiable real-world knowledge in the form of entity-relation triplets. This accumulated knowledge is explicit; the derived answers are explainable through the use of SQL-like queries on graph databases and can be tuned by domain experts to provide accurate results for a given field. Unfortunately, KGs are difficult to construct or expensive because of the need for a domain expert to manually tune them and currently do not generalize easily to different NLP tasks.

Large Language Models (LLMs) and Knowledge Graphs (KGs) are complementary constructions, and recent advancements in the field investigate how one model might address the deficiencies of the other. There are two schools of thought that define a framework to combine LLMs and KGs, (a)*KG-enhanced LLMs* 8.1 and (b)*LLM-augmented KGs* 8.2. The first is to ground the LLM's responses using factual knowledge by introducing a KG either in the pre-training stage or the inference stage, and the latter entails using LLMs as text encoders for tasks related to the KG (35).

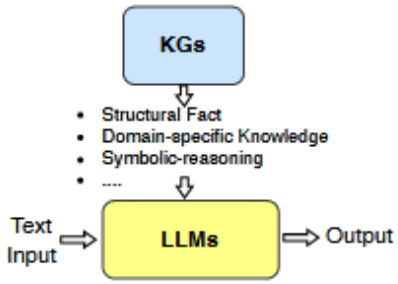


Figure 8.1: KG enhanced LLM

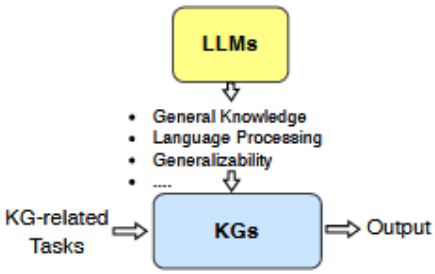


Figure 8.2: LLM augmented KG

Every paradigm possesses its own advantages, and the suitability of any given method is largely contingent upon the specific application at hand. For example, while incorporating the knowledge graph during the training phase proves beneficial for delivering factual or domain-specific information (27), it significantly increases both the time and cost associated with model training and necessitates re-training to comprehend previously unencountered knowledge. The aforementioned facts render training stage methods particularly advantageous for domain-specific applications that do not necessitate generalization. Conversely, inference methods are particularly well-suited for addressing open domains characterized by the continual evolution of knowledge.

One of the most common applications of augmenting KGs is *End-to-End KG construction* where a large language model, like BERT, is utilized to extract entities and relations from the corpus, thereby facilitating the construction of the knowledge graph (24).

In this work our objective is to construct a Knowledge Graph; however in the light of LLMs we should question ourselves how appropriate is this effort. From what was discussed above we see that the use of KG is logical as we need accuracy, explainability and flexibility in our answers. Nevertheless, a possible extension of

our work would be a chat bot that will enable the user to learn about the EU's financial regulations through from a Q/A point of view and then the use of a KG-enhanced LLM might prove beneficial.

9 Conclusion

Konstantinos Panagiotis Apostolou & Ioannis Dikaioulias

Natural language processing (NLP) is a sub discipline of data science that is concerned with data represented through natural language. The primary tasks are information retrieval and knowledge manifestation and usually data are collected from vast corpora of text and either rule-based, statistical or deep learning methods are applied.

A knowledge graph is an organized representation of real-world entities and their relationships. It is generally kept in a graph database, which inherently preserves the relationships among data elements. Entities inside a knowledge graph may denote objects, events, circumstances, or concepts. The relationships among these things encapsulate the context and significance of their connections.

The objective of this work was to construct a knowledge graph of European Union financial regulations, which involved the extraction of information as entities (a process known as Named Entity Recognition) and relationships (termed Relationship Extraction) from raw data. Information extraction can be approached through either open-domain (OpenIE) or closed-domain (ClosedIE) methods. Open-domain methods are more flexible since they allow for a schema-less approach, although through fine tuning one could link derived entities and relationships to a knowledge base. However, this flexibility comes at the cost of difficulties in evaluation, where either a ground-truth test set is required or the input of a domain expert. On the other hand, closedIE has the distinct advantage that testing can be done through traditional metrics such as Recall, Precision or F1 score, but the supply of domain-specific knowledge bases is usually scarce. In light of the intrinsic challenges linked to domain-specific methodologies, we developed a methodology that is inspired by OpenIE and rule-based techniques for information extraction.

In order to construct the knowledge graph, a pipeline was derived. Initially, the raw text needed to be denoised and processed in a way that expressions which referenced the same entity were disambiguated. Having acquired a "clean" dataset the ClausIE model was applied. ClausIE is based on grammatical and syntactical rules and extracts subject-verb-object triplets. These triplets are in need of futher process and in particular the object part, were large sentences are included. Large sentences carry a lot of semantic meaning but detriment the usability of the knowledge graph, therefore, a technique was used to filter out the noun chuncks from each object. Finally, we obtained a preliminary graph that was manually explored from us to in order to derive a schema that would fit the data. Once we derived the use cases we were able to assign a class to each node and construct the knowledge graph. Also, in the context of making it more informative, we mapped the external FIBO ontology to our KG. This augmented our knowledge graph with disambiguation properties where entities could be looked up in FIBO to find their official meaning. This final knowledge graph is illustrated in figure 5.39.

Regarding deep learning methods, the REBEL model was used to tackle automated knowledge extraction and the generated triplets were compared to those produced by the rule-based method, which served as the ground truth set. The results demonstrated that REBEL could extract the majority of the information without requiring additional tuning. That level of performance, along with the ability to easily tune the model to a specific domain with the use of predefined ontology, illustrates the power deep learning methods have.

Furthermore, we investigated the utilization of graph analytics to reveal hidden patterns inside the early graphs, enabling us to refine an external knowledge base and enhance its compatibility with our graph. A pipeline utilizing centrality analysis with the HITS algorithm and community detection with Louvain enabled us to derive a compact graph and identify a potential use-case for filtering FIBO and constructing a knowledge graph. This remains a prospective avenue for further investigation.

Last but not least, we explored large language models (LLMs) and their potential relevance to our issue. Large Language Models represent the future of natural language processing; nevertheless, they currently possess intrinsic limitations that render them inadequate for providing a comprehensive solution to our issue. Conversely, constructing knowledge graphs is challenging, and their efficacy diminishes without the presence of a domain expert. This two constructs have to earn from each other and we propose as a future research

path the development of an in-house knowledge graph-enhanced LLM that would enable the retrieval and utilization of information stored inside regulatory documents.

To conclude, our goal was to construct a knowledge graph while exploring possible benefits such a construct offers. There are several ways to achieve this, ranging from traditional rule-based methods to state-of-the-art techniques harnessing large language models and transformers. Another dichotomization is between open-domain and closed-domain methods. At any place in the information extraction chain, one thing is apparent though; the critical position the human input holds. Regardless of the choices previously mentioned, the necessity of a domain expert to pose the write questions and make use cases, to define the schema of the knowledge base or to label a testing set to evaluate the final knowledge graph is evident. In other words, knowledge graphs are information storage and representation tools built with cutting-edge techniques, but still, their usability and efficacy lie in the hands of the user.

Bibliography

[die] Representation and Extraction of Diesel Engine Maintenance Knowledge Graph with Bidirectional Relations Based on BERT and the Bi-LSTM-CRF Model | IEEE Conference Publication | IEEE Xplore.

[2] (2024). Königsberg bridge problem | Mathematics, Graph Theory & Network Theory | Britannica.

[3] (2025). huggingface/neuralcoref. original-date: 2017-07-03T13:04:16Z.

[4] Agrawal, G., Deng, Y., Park, J., Liu, H., and Chen, Y.-C. (2022). Building Knowledge Graphs from Unstructured Texts: Applications and Impact Analyses in Cybersecurity Education. *Information (Switzerland)*, 13(11).

[5] Arcila, Damian Trilling & Carlos, W. v. A. (2022). Computational Analysis of Communication.

[Artifex] Artifex. Pymupdf 1.25.1 documentation. `https://pymupdf.readthedocs.io/en/latest/`. Accessed on 2025-01-13.

[Barrasa and Webber] Barrasa, J. and Webber, J. Building knowledge graphs. O'Reilly Online Learning. Accessed on 2025-01-13.

[8] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008. arXiv:0803.0476 [physics].

[9] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117.

[10] Chourdakis, E. and Reiss, J. (2018). Grammar informed sound effect retrieval for soundscape generation. In *DMRN+ 13: Digital Music Research Network One-day Workshop*, page 9, London, UK.

[11] Clark, K. and Manning, C. D. (2016). Deep Reinforcement Learning for Mention-Ranking Coreference Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Austin, Texas. Association for Computational Linguistics.

[12] Del Corro, L. and Gemulla, R. (2013). ClausIE: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, Rio de Janeiro Brazil. ACM.

[13] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs].

[FIBO] FIBO. Financial industry business ontology (fibo). `https://spec.edmcouncil.org/fibo/`. Accessed on 2025-01-13.

[15] for Budget (European Commission), D.-G. (2018). *Financial regulation applicable to the general budget of the Union: July 2018*. Publications Office of the European Union, LU.

[16] Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239.

[17] Gómez, S. (2019). Centrality in Networks: Finding the Most Important Nodes. In Moscato, P. and de Vries, N. J., editors, *Business and Consumer Analytics: New Ideas*, pages 401–433. Springer International Publishing, Cham.

[18] Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. (2021). Knowledge Graphs. *ACM Computing Surveys*, 54(4):71:1–71:37.

[19] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spacy: Industrial-strength natural language processing in python. https://doi.org/10.5281/zenodo.1212303. Accessed on 2025-01-13.

[20] Huguet Cabot, P.-L. and Navigli, R. (2021a). REBEL: Relation Extraction By End-to-end Language generation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[21] Huguet Cabot, P.-L. and Navigli, R. (2021b). REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[22] Jeon, S., Khosiawan, Y., and Hong, B. (2013). Making a Graph Database from Unstructured Text. In *2013 IEEE 16th International Conference on Computational Science and Engineering*, pages 981–988.

[23] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[24] Kumar, A., Pandey, A., Gadia, R., and Mishra, M. (2020). Building Knowledge Graph using Pre-trained Language Model for Learning Entity-aware Relationships. In *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 310–315.

[25] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461 [cs] version: 1.

[26] Li, C., Yang, X., Luo, S., Song, M., and Li, W. (2022). Towards Domain-Specific Knowledge Graph Construction for Flight Control Aided Maintenance. *APPLIED SCIENCES-BASEL*, 12(24):12736. Num Pages: 20 Place: Basel Publisher: MDPI Web of Science ID: WOS:000902307700001.

[27] Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., and Wang, P. (2019). K-BERT: Enabling Language Representation with Knowledge Graph. arXiv:1909.07606 [cs].

[28] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs].

[29] Neo4j (2024a). Fully managed graph database service | neo4j auradb. Accessed on 2025-01-13.

[30] Neo4j (2024b). Native graph database | neo4j graph database platform. https://neo4j.com/product/neo4j-graph-database/. Accessed on 2025-01-13.

[Neo4j Graph Data Platform] Neo4j Graph Data Platform. neosemantics (n10s): Neo4j rdf & semantics toolkit - neo4j labs. https://neo4j.com/labs/neosemantics/. Accessed on 2025-01-13.

[32] Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582.

[Noy and McGuinness] Noy, N. F. and McGuinness, D. L. Ontology Development 101: A Guide to Creating Your First Ontology.

[34] Orlando, R., Cabot, P.-L. H., Barba, E., and Navigli, R. (2024). ReLiK: Retrieve and LinK, Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget. arXiv:2408.00103 [cs].

[35] Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., and Wu, X. (2024). Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599. arXiv:2306.08302 [cs].

[PDFMiner.six] PDFMiner.six. Pdfminer.six documentation. `https://pdfminersix.readthedocs.io/en/latest/`. Accessed on 2025-01-13.

[37] Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., and García-Castro, R. (2022). LOT: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755.

[Schema.org] Schema.org. Schema.org. `https://schema.org/`. Accessed on 2025-01-13.

[39] Sukumar, S. T., Lung, C.-H., and Zaman, M. (2023). Knowledge Graph Generation for Unstructured Data Using Data Processing Pipeline. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 466–471. ISSN: 0730-3157.

[40] Sutton, C. and McCallum, A. (2010). An Introduction to Conditional Random Fields. arXiv:1011.4088 [stat].

[41] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention Is All You Need. arXiv:1706.03762 [cs].

[W3C] W3C. Rdf - semantic web standards. `https://www.w3.org/RDF/`. Accessed on 2025-01-13.

[43] Wang, X., Sun, Y., Chen, C., and Cui, J. (2022). A Relation Extraction Model Based on BERT Model in the Financial Regulation Field. In *2022 2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, pages 496–501.

[44] Wolf, T. (2020). State-of-the-art neural coreference resolution for chatbots. Medium. Accessed on 2025-01-13.

[45] Yao, J.-Y., Ning, K.-P., Liu, Z.-H., Ning, M.-N., Liu, Y.-Y., and Yuan, L. (2024). LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples. arXiv:2310.01469 [cs].

[46] Yiming, L. and Li, D. (2022). Research on the Construction of Maritime Legal Knowledge Graph. pages 903–908.

[47] Zhang, J. and Luo, Y. (2017). Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network. pages 300–303. Atlantis Press. ISSN: 1951-6851.

[48] Zhao, Q., Huang, H., and Ding, H. (2021). Study on Military Regulations Knowledge Construction based on Knowledge Graph. pages 180–184.

Appendices

Appendix A

Prediction Metrics

This is the content for the TF-IDF Algorithm appendix. Add more text here as needed.

For any given stastical experiment, let's say with two possible states and two possible outcomes we have the contigency matrix:

	Prediction A	Prediction B
True State 1	TP	FN
True State 2	FP	TN

Table 1: 2x2 Contingency Matrix

where **TP** stands for True Positive, **FN** for False Negative, **FP** for False Positive and **TN** for True Negative. Using these quantites we can derie the formulas for the Precision, Recall, F1 and Accuracy metrics.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Appendix B

TF-IDF Algorithm

The TF-IDF algorithm comprises of two main parts. The Term-Frequency part (TF) and the Inverse Document Frequency (IDF) part.

Given a collection of documents

$$D = \{d_1, d_2, d_3, \ldots, d_n\}$$

For each document, d, the TF score for each token is calculated as such:

$$\mathsf{TF}(t, d) = \frac{\mathsf{f}(t, d)}{\sum_{w \in d} \mathsf{f}(w, d)}$$

t : The token on which the TF is calculated
d : The given document
w : The tokens contained in the document

In simple words, the TF score represents how frequent a token is in a specific document compared to all the tokens of that document. Although this metric can serve as an importance metric, it does not take into account the rest of the corpus (collection of documents). If a specific token has high frequency in a document, but also appears frequenty in every document, then this token is more likely not of high importance. For example, a stopword such as "the", would have a really high frequency in a specific document, but also in every document in the corpus. Thus, to take into account the entire corpus, a penalty is required.

The IDF score penalizes high values of the TF score. The IDF for each token in a given document d, is calculated as such:

$$\mathsf{IDF}(t) = \mathsf{log}\left(\frac{N}{1 + \mathsf{DF}(t)}\right)$$

t : The token on which the TF is calculated
N : Total documents in the corpus
$\mathsf{DF}(t)$: Number of documents that t appears

*The addition of 1 in the denominator serves as a way to ensure that no zero divisions are made. This is especially a problem where the size of the corpus, N, is too large, while the number of documents that t appear, is extremely low.

Low values of the IDF, for a specific token, translates to a high appearance of this token across the collection of documents, while high values of the IDF, indicate a low appearance in the corpus.

The combination of both the TF and IDF score, lead to the combined TF-IDF score. This score is defined as the multiplication of the above scores:

$$\mathsf{TF\text{-}IDF}(t, d) = \mathsf{TF}(t, d) \times \mathsf{IDF}(t)$$

$$\mathsf{TF\text{-}IDF}(t, d) = \frac{\mathsf{f}(t, d)}{\sum_{w \in d} \mathsf{f}(w, d)} \times \mathsf{log}\left(\frac{N}{1 + \mathsf{DF}(t)}\right)$$

For a token t, in a given document d, the TF-IDF measures the importance of that token in the document, in relevance to the entire collection of documents. A high TF-IDF score translates to high importance while a low TF-IDF score translates to low importance.

The TF-IDF are represented in a Document-Term Matrix (DTM), also known as a bag-of-words (....reference). The DTM matrix is of Documents X Vocabulary dimensions, where Documents refers to the total number of documents, where each row represents a document, and Vocabulary refers to the total number of unique tokens in the collection of documents.

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be