# AUTOMATIC MAINTENANCE OF COVID-19 RELATED KNOWLEDGE GRAPHS BASED ON LARGE-SCALE INFORMATION EXTRACTION IN SCIENTIFIC LITERATURE

Universiteit Utrecht

Author:
Argyro (Iro) Sfoungari 6528015

Prof. Yannnis Velegrakis - Project Supervisor
Dr. Mel Chekol - Second Supervisor
Dr. Theodore Dalamagas - External Supervisor

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science
in
Computing Science

Department of Information and Computing Sciences
Utrecht University
July 2021

# Abstract

The continuous and ongoing enrichment of publications related to new scientific findings is inspiring, however, it generates new challenges for the scientific community. At any given time, researchers have to evaluate manually a plethora of publications so it is not uncommon for them to spend time reading documents that are irrelevant or inaccurate. The official form of scientific representation is document-based and therefore cannot be processed automatically. Consequently, it is necessary to find innovative ways to process and evaluate scientific text automatically. An approach which has recently been adopted by the scientific community for dealing with the above is the production of Knowledge Graphs (KGs) from scientific text. Aiming to contribute to the necessity for machine-actionable scientific representation, we create the UA-Graph [1]. The UA-Graph is a scientific KG produced in order to assist researchers in finding papers relevant to their interests. Along with the UA-Graph, we present the graph production process. We propose a methodology that processes data and extracts structured scientific information in order to be inserted in the KG. We then design a data model that is general enough to be considered domain-independent and yet precise at the same time. Finally, we implement the data model in a graph DBMS and prove that the produced graph can answer complex queries so as to facilitate scientific research.

[1] UA in an abbreviation for Utrecht University and Athena Research Center

# Acknowledgements

With the hope that this thesis will be of use to the members of the research community in their future endeavors.

This project would not have been possible without the participation, support and guidance of Prof. dr. Yannis Velegrakis, to whom I wish to extend my sincere thanks. I would like to thank ATHENA Research Center and especially Dr. Theodore Dalamagas, Research Director at ATHENA Research Center, whose support was particularly valuable. Furthermore, it should be noted that this project was undertaken during a global pandemic under difficult circumstances. My supervisors showed great understanding and provided systematic assistance and communication on a weekly basis. Moreover, I would like to thank Dr. Mel Chekol for his critical commentary regarding the project.

I am also grateful for the invaluable support provided by my family and friends during the last years of studying at Utrecht University. Throughout the master's application process, I promised to bring all my hard work and enthusiasm to Utrecht in case I was accepted. I am really happy that I did as promised.

Argyro (Iro) Sfoungari

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Every year a plethora of scientific papers becomes publicly available to present the latest progress in every scientific discipline. Papers are published in conference proceedings and journals and their official representation form is document-based. This means that researchers present their accomplishments in documents that involve unstructured text, illustrations and tables. These text documents are stored digitally, usually in PDF format, and can be found in online and offline scientific search engines.

The basic assumption before conducting any type of research is that the researchers involved are up to date with developments in their particular field. Then, they need to be aware of the recent progress on a global level, as a result of the related studies of their colleagues. The remarkable pace of advancement in science requires the involvement and vigilance of experienced researchers who will at the same time be in constant communication with one another for the benefit of all. It is often difficult to examine carefully the limitless amount of information and it is not uncommon for researchers to be disoriented reading research findings that are irrelevant or inaccurate. The document-based form of scientific representation seems to be the source of the above-mentioned concern.

Document-based scientific literature cannot be processed automatically, therefore it is difficult for researchers to quickly retrieve targeted information. The process of finding relevant literature is supported by scientific search engines (eg. Google Scholar [1]). These search engines, however, support keyword search and usually return a plethora of potentially useful results. Other existing systems (eg. Microsoft Academic Graph [2], Crossref [3] and others), use publication metadata and do not include information regarding the actual content of the text. When researchers use these search engines to find some interesting papers, they then have to evaluate and process each paper individually and subsequently create a literature overview. This procedure is very time-consuming and usually has to be repeated several times. The scientific community is, therefore, looking for automatic solutions to facilitate scientific text analysis, a difficult task indeed since the form of scientific literature contains terminology, large sentences, many abbreviations, and formulas.

## 1.1 Background

The need to identify ways to represent scientific text in a machine-readable form is a requirement that will grow over time. An innovative idea that can supply scientific literature with structure is its representation using Knowledge Graphs (KGs). KGs have been a source of great interest, in recent years, for many researchers working both in academia and industry [4]. Until 2012, the term KG was not particularly popular outside of scientific circles. The popularization of KGs is principally attributed to Google [5]. The momentum of Google's idea to represent general knowledge using KGs was so powerful that the term since then has been widely used while lacking a clear definition [4]. An indicative definition according to Paulheim [6] is that a Knowledge Graph: (i) mainly describes real-world entities and their interrelations, organized in a graph (ii) defines possible classes and relations of entities in a schema (iii) allows for potentially interrelating arbitrary entities with each other (iv) covers various topical domains. Since 2012, many companies that dominate the specific market use data represented by and processed as graphs. KGs in collaboration with other cutting-edge fields have generated numerous interesting and innovative applications. Regarding the scientific domain, KGs are efficient means to represent scientific findings in a structured way. Furthermore, KGs allow information acquisition, curation and evaluation simply and efficiently for the benefit of researchers.

## 1.2 Motivation

Our motivation is to contribute to the necessity for machine-actionable scientific representation through KGs and subsequently to assist researchers in finding papers relevant to their field of interest. According to related research, many projects are engaged to scientific KGs. The majority of them, however, produce KGs based on pre-existing data models. Our project intends to prove that a KG can be implemented based on a data model which is so general that it can be considered domain-independent and very specific at the same time.

## 1.3 Challenges & Objectives

Our objective is to produce a scientific KG which is a rather demanding process as it requires the fulfillment of many individual steps. The KG production process typically involves a) a methodology that processes the data and extracts information that will be inserted in the graph, b) the design of a data model and c) the actual KG implementation. Each of these steps demands a lot of commitment and experimentation over many different research approaches.

The first challenge related to the KG production concerns data collection and processing. The creation of a semi-automatic method that receives data and produces structured information requires a lot of research mostly over the Natural language processing (NLP) and Information Extraction (IE) techniques that will be used. Data processing is itself a difficult task, let alone in the case of scientific text which contains large sentences, terminology, abbreviations, and formulas. Even the order that the selected techniques will be applied on the available corpus is very important and requires a lot of experimentation.

The second research challenge concerns the design of a flexible data model. We envisioned the design of a general and domain-independent data model which at the same time represents efficiently all the concepts of the scientific domain. Furthermore, the model consists of classes and allows class hierarchies and inheritance. The design of such an innovative data model requires a lot of research regarding the pre-existing data models and the limitations arising from them. Additionally, through the model design process, many alternative models are expected to occur until we reach the optimal result.

The third research challenge concerns the data model implementation using a graph DBMS. This task requires extensive research regarding the available graph DBMS and their capabilities in order to select the one that will best support our model. The final KG will be produced after several attempts and compromises.

## 1.4 Contributions

Our main contribution is the production of the UA-Graph[1], which is a scientific KG that will assist researchers in finding papers relevant to their research. The production of the graph is based on three individual phases and therefore three individual contributions.

$C1$ → We propose a semi-automatic methodology that receives scientific data (publication metadata and unstructured text) and extracts structured information ready for use as input to the KG. Our methodology applies NLP, Topic and Information Extraction techniques to produce meaningful results from scientific text.

$C2$ → Additionally, we design the UA-data model, a flexible and interoperable data model which is domain-independent and consists of classes, instances and rela-

[1] UA in an abbreviation for Utrecht University and Athena Research Center

tionships between them. Moreover, the model allows class hierarchies and inheritance.

$C3$ → We implement the UA-data model in Neo4j[2], which is a graph DBMS and finally produce the UA-Graph a scientific and hierarchical KG. In addition, we present a methodology that allows querying this graph top-down.

In order to evaluate both our methodology and the produced KG, we used the COVID-19 Open Research Dataset (CORD-19) [7]. The COVID-19 pandemic[3] is today's major global health concern. Since the pandemic began to spread, a plethora of related publications have appeared. Consequently, research towards a cure will be significant globally for some time to come. Aiming to support pandemic researchers and research workers we use the biomedical domain as our paradigm although, the project design, is domain-independent and can be used in any other scientific field.

## 1.5 Thesis Outline

The rest of this thesis is structured as follows. In section 2 we present a review of the related work. In this section, we inform the readers about similar projects and also introduce the steps that are usually followed to produce a KG from scientific text. In section 3 we address the actual problem and explain the concepts that we will use for the KG production. In section 4 present all the steps we followed in order to produce the UA-Graph. In section 5 we evaluate our methodology and the produced graph. Finally, in section 6 we discuss future extensions and conclusions. In Appendix A we guide the readers for the reproduction of the semi-automatic text processing methodology. In Appendix B we present the Cypher scripts that have been used to implement the UA-Graph. The thesis outline is illustrated in Figure: 1.1.

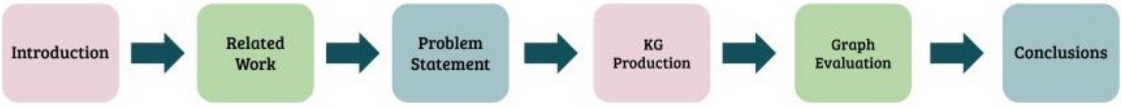


Figure 1.1: Overview of thesis outline

[2] https://neo4j.com/

[3] In December 2019, the first cases of the novel coronavirus disease (COVID-19) appeared in Wuhan City, Hubei Province, China [8]. Coronavirus, which causes severe acute respiratory syndrome, had such strength that very quickly spread beyond Hubei province, to all of China at first and eventually to the entire world. Both mortality rates and rapid transmission speed were so alarming that on March 11, 2020, the World Health Organization characterized the situation as a global pandemic [9].

# Chapter 2

# Related Work

In recent years, generating KGs from unstructured text attracts significant attention within the research community. Information Extraction (IE) techniques are intended to produce a structured version of the information presented in the text so as to make it computer-understandable. Text representation through KGs provides opportunities for simple and easy content navigation and surpasses conventional keyword search, enabling the production of direct answers to complex queries.

## 2.1 Steps Leading to KG Generation

Typically, KG generation from unstructured text is a 4-step process and consists of: Text Preprocessing, Named Entity Recognition (NER), Relationship Identification (RI) and Enrichment as illustrated in Figure 2.1. NER and RI are necessary, whereas Preprocessing and Enrichment are optional.

*Step* 1*: Text preprocessing* is a widely used NLP task and is mainly applied to simplify text for future use. Tasks such as co-reference resolution, abbreviation resolution, sentence simplification, tokenization, stop words removal, etc. may differ depending on the data quality and the nature of the project concerned. Even though text preprocessing is a crucial step, it is frequently omitted. This happens either because scientists rely on usual datasets that have been already preprocessed, or because there are advanced tools using ultra-modern technologies (e.g. neural networks) that have decreased the necessity for preprocessing [10].

*Step* 2*: Named Entity Recognition* follows preprocessing and is still another important NLP task. NER identifies named entities (i.e. natural world objects) mentioned in the text and then classifies these entities into predetermined categories.

*Step* 3*: Relationship Identification.* Once NER step has been completed, the next step is to identify relationships between these entities and describe them as triples. Relation triples represent text in the sequence of subject, predicate, and object. Subject and object represent the extracted entities of Step 2, while predicate is the relation that connects these entities. Triple extraction can be achieved through multiple information extraction techniques.

*Step 4: Enrichment.* A KG can either be enriched or rely on preexisting semantic data models, that represent knowledge about a specific domain i.e. ontologies. There exist many standard definitions about ontologies [11], [12], a more descriptive is that an Ontology:

- Is a formal representation of a domain
- It uses a standard vocabulary to describe the main concepts of a domain as well as the relations among the concepts
- It can be shared between various applications and can be enhanced
- Shared vocabularies eliminate the problem of integration, which arises when two ontologies model the same concept differently
- It supports inferencing and reasoning

The need to develop a universal vocabulary shared by a community is crucial as it promotes interoperability, which is the main idea for KG development. Although some projects do not include an already existing ontology, others use it either to enhance relation and triple extraction or to enrich the KG.

What follows is a presentation of related projects that have been categorized according to the methods Steps 2 and 3 are performed but are also according to other parameters.

## 2.2 Scientific Text Information Extraction

### 2.2.1 Named Entity Recognition and Semantic Annotations

Named Entity Recognition (NER), is considered a fundamental process in KG generation, as it identifies and disambiguates name entities from unstructured text, which then can be linked to relevant concepts or in other words, can be semantically annotated [13]. DepressionKG [14] is a disease-centric knowledge graph generated by the integration of multiple heterogeneous resources, whose content has been semantically annotated with medical terminologies using Xerox's NLP tool XMedlan. Following the same line of reasoning, COVID-KG [15] designers use CORD-NER [16] annotation system, which combines NER methods from four different sources and promotes comprehensive name entity annotation. In the CORD-19 Named Entities Knowledge

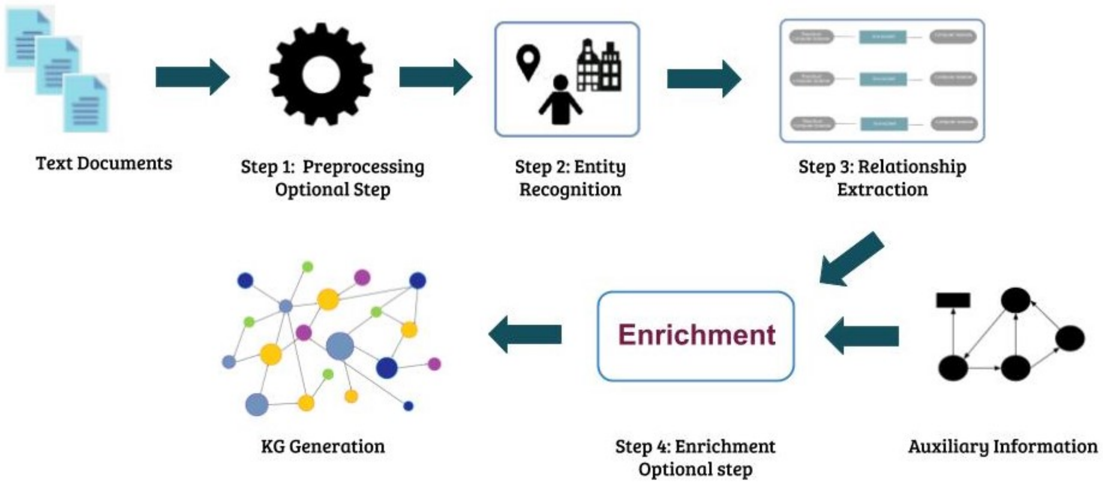

Text Documents
Step 1: Preprocessing Optional Step
Step 2: Entity Recognition
Step 3: Relationship Extraction
Enrichment
KG Generation
Step 4: Enrichment Optional step
Auxiliary Information

Figure 2.1: The four steps that typically lead to KG production

Graph (NEKG) part of the Covid-on-the-Web [17] project, NEs are identified and then used as a method to build relevant links between articles in the CORD-19 [18] corpus and knowledge bases with entity description such as DBpedia, Wikidata and Bioportal ontologies. Annotation tasks are considered very challenging due to ambiguity, therefore there are tools (e.g. DBpedia Spotlight [19]), aiming to disambiguate natural language references in particular resources.

### 2.2.2 NLP

Notwithstanding the above, KGs can be generated using NLP or NLP along with another technique. KGen designers [20] do not assume that NER should be completed before RI, so they perform both steps at the same time. The first step in the KGen generation method is the preprocessing of the unstructured text. In so doing, the scientific text is simplified for future use. Then, the preprocessed text undergoes Semantic Role Labeling (SRL) to extract triples in the form of subject, predicate, and object. SRL is a technique that identifies the verb in a sentence as well as the semantic role of the remaining words (agent, patient). For each triple, the verb is assigned to the predicate, while subject and object are assigned to the agent and patient respectively. Concurrently, concepts and entities from the preprocessed text are extracted through tokenization and PoS tagging, which then are mapped to an ontology. The product of the above-mentioned steps is a final set of triples.

### 2.2.3 "Innovative" Methods

As research over KGs is evolving, new projects that do not fall under the aforementioned categories are constantly presented. COVID-19 knowledge Graph (CKG) [21] is an inventive KG that combines scientific metadata (papers, authors, institutions) and information derived from the scientific text (concept, topics). CKG generation is based upon Comprehend Medical (CM) Detect Entities V2, an Amazon Web Service [22] that combines NLP and Machine Learning (ML) and extracts NEs and relationships between them. CM also classifies the extracted entities into entity types or categories and entity attributes. Concerning the topic detection, Latent Dirichlet Allocation (LDA) along with multi-label classification are applied towards assigning topic labels (provided by professionals) to topic models. Argumentative Knowledge Graph (AKG), part of the Covid-on-the-Web project [17] is another exceptional KG. AKG designers employ ACTA [23], a platform that automatically analyzes clinical studies, to generate an argumentative KG. ACTA is based on argumentative mining methods, which identify the argumentative elements inside text (i.e. claims, premises), followed by their relations (i.e. support, attack), resulting in a graph containing the former as nodes and the latter as edges.

### 2.2.4 Manual Enhancement

As indicated above there are many automatic techniques that lead to a KG generation including metadata and topics extracted from scientific text. However, automation falls short when semantic representation deals with more complex structures. KG based on SemSur ontology [24] is a KG generated with less automation compared to other projects, which however, offers a much more complex structure to represent survey articles descriptively. Experts extract instances manually based on the ontology classes. They then import those instances to the ontology and finally the KG is generated including the community's contribution.

### 2.2.5 KGs from Structured Data

There are cases, however, where none of the preceding steps are needed, as data are already in a structured form like RDF, CSV etc. In such cases, there are still obstacles either during data acquisition or during schema mapping. SCM-KG [25] project contains a KG generated from scientific metadata and is based on the idea of a functional framework, which combines data of different structured formats. In such cases converting data in a common model i.e. RDF and mapping them to an ontology, is very challenging as direct mapping is not always possible. Dealing with this issue, SCM-KG pipeline uses Sparqlify-CSV, to map CSV files to an ontology.

When this is not feasible ETL component shapes the data in the required format.

## 2.3 Related Work Summary

| Project | Data | Preprocessing | NER and RI | Ontology |
|---|---|---|---|---|
| DepressionKG [14] | Medical Data from multiple heterogeneous resources: PubMed, ClinicalTrial, Medical Guidelines, DrugBank, DrugBook, DrugBook, Wikipedia Antidepressant side effects, SIDER, SNOMED CT and Patient Data | | Direct entity and concept identification when is feasible, but in case of unstructured text NER and RI are performed using, a semantic annotation tool (XMedlan). | |
| Multimedia COVID-KG [15] | CORD-19 [18] | | i) Text knowledge extraction (NER and entity linking based on the ontology defined in the CTD). ii) Highly detailed entity extraction and semantic annotation using CORD-NER [16]. iii) Additional step: Image processing to extract visual information | Entity linking step is based on the ontology defined in the Comparative Toxicogenomics Database(CTD) [26]. |
| NEKG Covid-on-the-Web [17] | CORD-19 [18] | | NER and RI through semantic annotation using DBpedia, Wikidata and Bioportal. | DCMI, Bibliographic Ontology (FaBiO), Bibliographic Ontology, FOAF, Schema.organd Web Annotation Vocabulary have been used to enrich the KG. |
| KGen [20] | Unstructured text in plain-text format (e.g., a *.txt file) | NLP to simplify sentences: Sentence splitter, Abbreviation resolver, Sentence simplification | i) SRL extracts triples in the form of subject, predicate, object. ii) Entity and concept identification (Tokenization and PoS Tagger), which then are mapped to the ontology recommended by the NCBO bioportal. iii) Combination of the above-mentioned steps (i,ii) to produce the final set of triples. | Step ii: Performs entity identification from sentenses to obtain links to the domain ontology that is recommended by the NCBO bioportal. In this case the ontology is used to enforce entity linking. |
| AKG Covid-on-the-Web [17] | CORD-19 [18] | | Argumentative element identification (i.e. claims, premises) and relationship identification (i.e. support, attack) have been established automatically using ACTA [23], a clinical text analysis tool. | Argument Model Ontology (AMO), the SIOC Argumentation Module (SIOCA) and the Argument Interchange Format have been used to enrich the KG. |
| CKG [21] | CORD-19 [18] | | In this project NER and relationship extraction are performed together using Amazon Comprehend Medical (CM) [22]. For topic detection LDA and multi-label classification have been used. | |
| KG based on SemSur ontology [24] | Specific research articles (surveys) and the respective surveyed articles | | i) SemSur ontology is created, ii) Instances from text are extracted manually based on SemSur ontology, iii) SemSur Ontology + Instances + Community : build the KG | This project is built on the top of SemSur ontology, which uses substes of other existing vocabularies i.e. DC, SWRC, FOAF, MLS, DCMI, DEO, LSC, DOAP. |
| SCM-KG [25] | Heterogeneous data in different structured formats such as CSV, RDF, web pages etc. | | Dealing with structured data from heterogeneous resources → No NER AND RI. i) Data acquisition ii) Creation of an integration ontology (see next column) iii) Data conversion in a common model and ontology mapping. This may be feasible when dealing with RDF data. Though, there are cases where direct mapping of e.g. CSV is not possible. → ETL component to shape the data etc. iv) similarity measures to prevent multiple instances referring to same things. | KG generation is based on its core ontology, which is created using subsets of: SWRC, Dublin Core, and FOAF vocabularies. It is instantiated with data from the sources, and every time a concept is missing, a new relation type is defined. |

Table 2.1: *Related work summarized information*

## 2.4 Open Science Graphs

Open Science Graphs (OSGs) [27] are scientific KGs that support free and open access to graph representations of scientific metadata, but also promote FAIRness of science, as they can be easily accessed by anyone interested in sharing or using academic information. In other words, OSGs, are metadata graphs that are continuously updated by researchers whose studies are stored in global repositories. OpenAIRE, is a European project aligned with OSGs. OpenAIRE Research Graph represents metadata and links arising from more than 13,000 data sources globally [27]. After collection, data from heterogeneous sources are subjected to some processing, deduplication, and harmonization with a standard format. The OpenAIRE Research Graph data model [28] is simplified with the intention of promoting universal participation and collaboration. Synergic contribution and collaboration is considered necessary, in order to facilitate the ultimate goal of creating a Global Research KG. At this time, there are many projects like The Open Research Knowledge Graph [29], Research Graph[1] and The OpenCitations[2] graph that share the same principles and exchange metadata and links with the OpenAIRE Research Graph.

[1] https://opencitations.net/

[2] https://opencitations.net/

# Chapter 3

# Problem Statement

Given a collection of scientific publications $P$ and publication metadata $P_m$, we intend to create a scientific KG. $P$ are documents that present scientific findings using sequences of words, illustrations, and tables. $P_m$ are supplementary bibliographic information to $P$. Possible $P_m$ are publication authors, ids, publishers, and other structured information related to $P$.

Initially, we intend to create a methodology that employs NLP and IE techniques in order to produce structured information from $P$. Subsequently, we will use $P_m$ and structured information extracted from $P$ to create a scientific KG.
In particular:
Given a large set of scientific publications $P = \{p_1, p2, p3..., pn\}$ and publication metadata for each document:
$P_m = \{p1_{m_A}, p1_{m_J}, p1_{m_{Pub}}..., pn_{m_A}, pn_{m_{Jl}}, pn_{m_{Pub}}...\}$, we create a methodology:

$$\gamma : P \rightarrow \begin{cases} \text{T} \\ \text{E} \end{cases}$$

that extracts $T$ and $E$ from text where $T$ is a set of topics identified in the scientific text and $E$ are real-words objects described by single words and phrases. Each $T = \{keyword\_1 * prob + keyword\_2 * prob + ... + keyword\_n * prob\}$ is presented as a distribution of keywords that occur frequently in the scientific text and their respective probabilities. $T$ and $E$ should be associated with the publication they have been extracted from. Additionally, $P_m$ should also be connected with the publication they belong to.

The way that $T$,$E$ and $P_m$ will be associated with $P$ is determined by a data model that consists of classes, instances and relationships.

The produced graph will be a scientific KG capable of supporting any scientific domain.

# Chapter 4

# UA-Graph Production

We describe the methodologies we developed in order to produce the UA-Graph, which is a scientific KG. The production of the UA-Graph is based on the design of a flexible data model and then on the KG implementation. Therefore we discuss three aspects related to the UA-Graph production as illustrated in Figure: 4.1.

1. **Data Processing:** The first aspect concerns a methodology which processes data in order to extract structured information. Structured information can then be inserted into the KG.

2. **Model design:** The second aspect concerns the design of a flexible data model which represents the main concepts of the scientific domain through class hierarchies. The proposed model is general enough and therefore can be considered domain-independent.

3. **KG implementation:** The third aspect concerns the data model implementation using a graph DBMS (Neo4j[1]).

Below we discuss in detail the three aspects, which are related to the UA-Graph production.

## 4.1 Data Processing

Given a collection of scientific data, the data should be processed to produce structured information that can be inserted into the KG. We, therefore, propose a methodology which applies NLP and Information Extraction techniques in order to extract structured information from text. Our methodology as illustrated in Figure: 4.2 consists of three main tasks **Text preprocessing**, **NER** and **Topic Extraction**.

[1] `https://neo4j.com/`

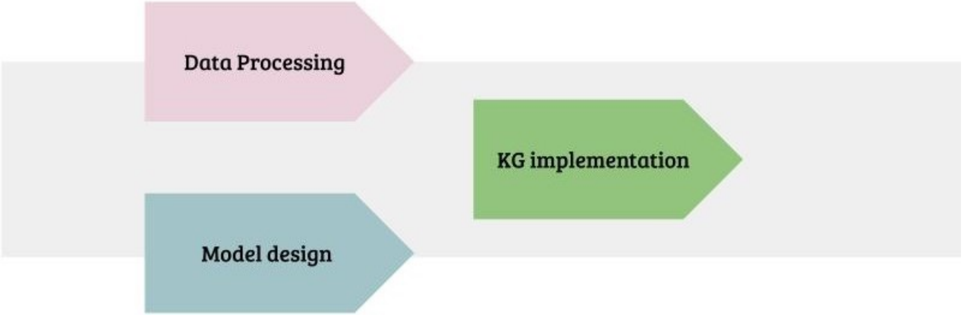


Figure 4.1: *The three main aspects which are related to the UA-Graph production*

### 4.1.1 Text Preprocessing

Text Preprocessing is the first task of our proposed methodology and involves the processing of unstructured text so that it can be used as input for the tasks that follow (NER and Topic Extraction). According to our methodology, the unstructured text should be subjected to 5 NLP techniques: **Language Detection, Tokenization, Stopword removal, Unnecessary character removal, Lemmatization** and **N-gram identification**. The result of text preprocessing is tokenized, lemmatized and clean text freed from unnecessary symbols and characters. Here we discuss the 5 NLP techniques in detail along with tools we used for their implementation and the resulting output of each one individually.

Before text preprocessing, our system separates records providing text (i.e. abstract) from those providing only publication metadata. Records providing only publication metadata can be inserted into the KG directly.

**Language Detection**

The UA-Graph supports only the representation of English publications. Therefore this step identifies the language that each scientific publication is written to and maintains only English documents. The language detection step is implemented using python's library langdetect[2].

**Tokenization**

Tokenization is an NLP task concerning the process of splitting text into smaller segments which are called tokens. A token can be a single word, a term, or even an entire sentence. Depending on the language and the desired format, tokens can be distinguished from the spaces between words or punctuation marks. Concerning the English text for example, a space between two words distinguishes the word and a full stop determines the boundaries of a sentence. Every language uses different methods

[2] `https://pypi.org/project/langdetect/`

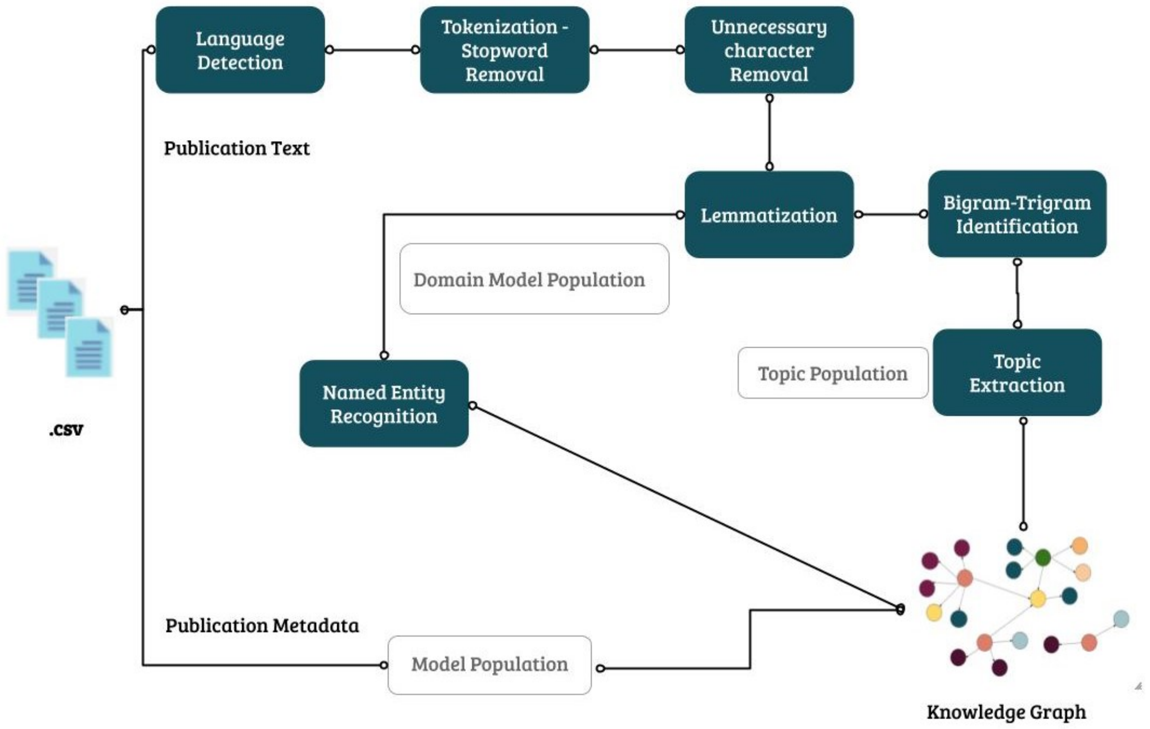


Figure 4.2: *The steps that process unstructured text and publication metadata for DB insertion*

to extract tokens. Tokenization's importance derives from the fact that smaller pieces of information presented in a document can be analyzed straightforwardly and can provide more accurate results about the meaning of the document as a whole. We implemented tokenization using spaCy's [30] tokenizer[3]. SpaCy is a free open-source python library for advanced NLP.

### Stopword removal

Stopword removal is the process of removing stopwords identified in the text. As stopwords, we consider all words that do not supplement text with valuable information. Stopwords usually include prepositions, articles and other uninflected parts of speech. Stopword removal is implemented adjacent to tokenization using SpaCy.

### Unnecessary character removal

Using Gensim's[4] simple_preprocess() library in this step, tokens are subjected to further cleaning so as to reduce "noise" which arises from neglected punctuation and unnecessary characters.

[3] https://spacy.io/api/tokenizer
[4] https://pypi.org/project/gensim/

**Lemmatization**

Lemmatization is the process which groups together different forms of the same token and replaces them all with the base root form of the word. The concept behind lemmatization is that the lemma, which is the root word without its specific suffix (third person or past and future tenses) can displace all the different forms of the word, without distorting the meaning of the text. All these different forms of the same word can then be analyzed as a single entity. Lemmatization is particularly important before topic extraction. If we extract topics neglecting lemmatization, different forms of the same word might be included among the representative topic keywords such as words covering the topic meaning (eg. test and tested). In doing so, forms of the same token would end up on the same subject, bypassing other important words for the topic formulation. After lemmatization, the already processed text can be used as input for NER. However, text for Topic Extraction needs to be subjected to one further step of preprocessing which is bigram and trigram identification. Lemmatization is implemented using spaCy's lemmatizer[5].

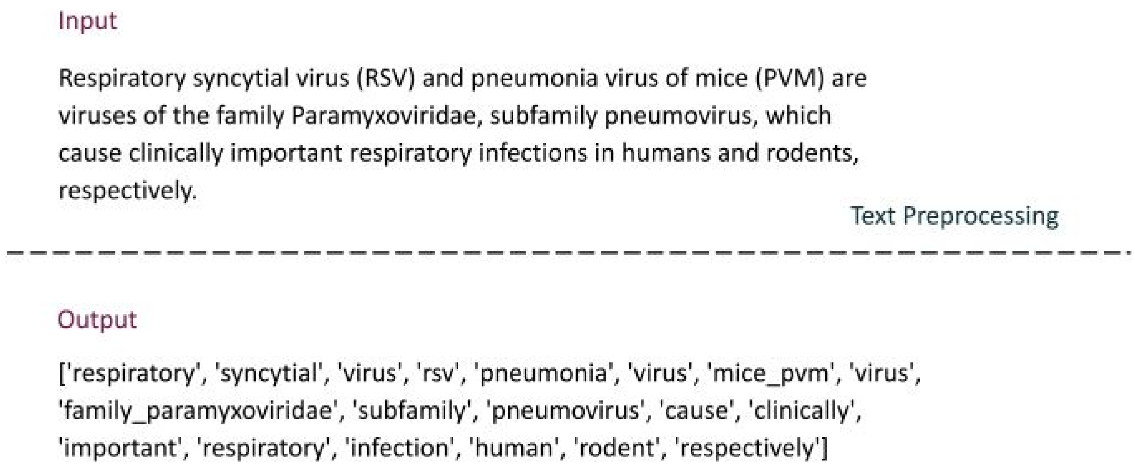
**N-gram identification**

*N*-gram identification is the process of identifying *N*-grams. As *N*-gram we consider a sequence of tokens that usually occur together in the text. In our case, we consider sequences of two words (bigrams) and three words (trigrams). After Bigram/trigram identification, bigrams and trigrams are converted into single tokens and their individual words are separated by an underscore (nitric_oxide, lactic_acid_bacterium). In topic extraction, bigrams and trigrams are considered interrelated so are represented as a single token and therefore should appear together in case they are included in the topic keywords. In the case of the entity recognition step, Bigram/Trigram identification can be overlooked. In entity recognition, words should appear together in case they have specific meanings based on the respective NER model. This is unrelated to the concept of sequences that occur together in the text. Bigram/Trigram identification is implemented using Gensim[6] library.

In Figure: 4.3 we present an example of a sentence which has been given as input in text preprocessing workflow. After text preprocessing, we see the equivalent output of this sentence. Within the output sentence, each token is enclosed by quotation marks. The lemmatizer turned the tokens: viruses, infection, humans, and rodents into their basic root form. Finally, the bigram/trigram identification model considered the words family, paramyxoviridae and mice, pvm as bigrams and therefore joined them using the underscore (family_paramyxoviridae, mice_pvm). In addition we see that articles and linking words are missing.

[5] `https://spacy.io/api/lemmatizer`

[6] `https://pypi.org/project/gensim/`



Input

Respiratory syncytial virus (RSV) and pneumonia virus of mice (PVM) are viruses of the family Paramyxoviridae, subfamily pneumovirus, which cause clinically important respiratory infections in humans and rodents, respectively.

Text Preprocessing

Output

['respiratory', 'syncytial', 'virus', 'rsv', 'pneumonia', 'virus', 'mice_pvm', 'virus', 'family_paramyxoviridae', 'subfamily', 'pneumovirus', 'cause', 'clinically', 'important', 'respiratory', 'infection', 'human', 'rodent', 'respectively']

Figure 4.3: *A sentence given as input in text preprocessing workflow and the equivalent output*

### 4.1.2 Named Entity Recognition

The second task in our methodology is NER. NER identifies and disambiguates entities from unstructured or semi-structured text. These entities are classified into predefined categories. As already mentioned in this project our case-study involves the Biomedical field, therefore we focus on biomedical named entity recognition (Bio-NER). Bio-NER is a NER task used to extract information from articles of biomedical, scientific, or clinical fields. Bio-NER is trained on scientific text and therefore it recognizes concepts of scientific content. Our methodology involves the use of three distinct packages which offer models for processing biomedical text: *Scispacy* [31], *Med 7* [32] and *Stanza* [33].

*ScispaCy* is an open-source python NLP package which aims to assist NLP implementanion in biomedical text. ScispaCy is based upon spaCy [30], which has been retrained based on data from a variety of biomedical sources. ScispaCy[7] offers different Bio NER models built on different entity categories. In our proposed methodology, NER task uses all the four ScispaCy models and identifies 28 entity types. The ScispaCy models along with the available Entity types can be seen in Table: 4.1.

[7] https://allenai.github.io/scispacy/

| Scispacy Models | | |
|---|---|---|
| Model | Description | Entity type |
| en_ner_craft_md | A spaCy NER model trained on the CRAFT corpus. | GGP, SO, TAXON, CHEBI, GO, CL |
| en_ner_jnlpba_md | A spaCy NER model trained on the JNLPBA corpus. | DNA, RNA, CELL_TYPE, CELL_LINE, PROTEIN |
| en_ner_bc5cdr_md | A spaCy NER model trained on the BC5CDR corpus. | DISEASE, CHEMICAL |
| en_ner_bionlp13cg_md | A spaCy NER model trained on the BIONLP13CG corpus | ANATOMICAL_SYSTEM, TISSUE, CANCER, CELLULAR_COMPONENT, SIMPLE_CHEMICAL, AMINO_ACID, CELL, ORGAN, ORGANISM* |

Table 4.1: *Scispacy models and Entity types*

| Med 7 Models | | |
|---|---|---|
| Model | Description | Entity type |
| en_core_med7_lg | model is trained on MIMIC-III free-text electronic health records | DOSAGE, DRUG, DURATION, FORM, FREQUENCY, ROUTE, STRENGTH |

Table 4.2: *Med 7 model and Entity types*

*DEVELOPING_ANATOMICAL_STRUCTURE, IMMATERIAL_ANATOMICAL_ENTITY, MULTITISSUE_STRUCTURE, GENE_OR_GENE_PRODUCT, PATHOLOGICAL_FORMATION, ORGANISM_SUBDIVISION, ORGANISM_SUBSTANCE

*Med* 7[8] is a transferable clinical NLP model trained on the MIMIC-III corpora and can recognize 7 drug-related entities. It can be installed and used as a part of ScispaCy. In our proposed methodology NER task identifies 7 drug-related entity types: dosage, drug, duration, form, frequency, route and strength. $Med7$ along with its available entity types can be seen in Table: 4.2.

*Stanza*[9] is a Python natural language analysis package which also contains Biomed-

[8] `https://github.com/kormilitzin/med7`

[9] `https://stanfordnlp.github.io/stanza/available_biomed_models.html`

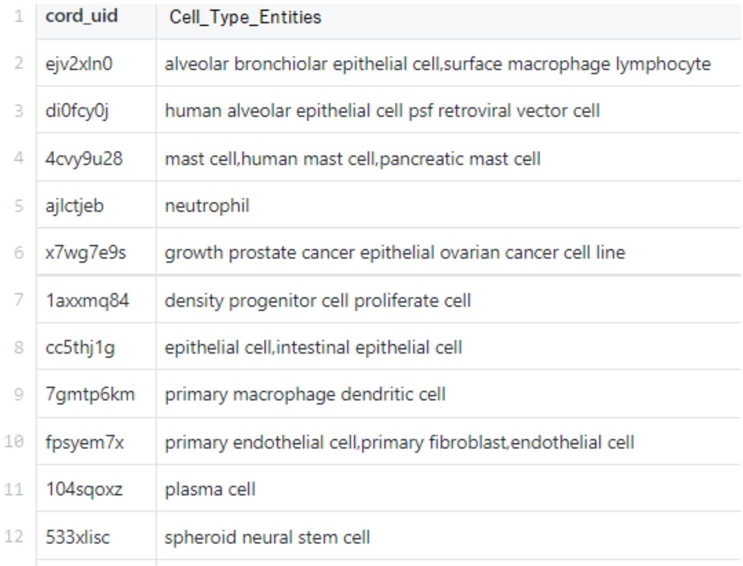
| Stanza Models | | |
|---|---|---|
| Model | Description | Entity type |
| i2b2 | All types of MIMIC-III clinical notes, general English Web Treebank. | TEST, PROBLEM, TREATMENT |

Table 4.3: *Stanza model and Entity types*

ical Models for Clinical NER and other tasks. The language models are pretrained on publicly available data-sources. BioNER models are pretrained on PubMed abstracts whereas clinical NER models are pretrained on the clinical notes from MIMIC-III database. From Stanza our proposed methodology task uses the i2b2 model and identifies three entity types: test, problem and treatment, as can be seen in Table: 4.3.

An additional and optional NER-subtask is entity categorization. As NER models usually export a plethora of entity types, our system allows users to categorize the extracted entities in case they need to. In Figure: 4.4 we see an example of categorized Cell Line entities grouped by article's id.

The output of NER is very important as it supports the domain model population.



| cord_uid | Cell_Type_Entities |
|---|---|
| ejv2xln0 | alveolar bronchiolar epithelial cell,surface macrophage lymphocyte |
| di0fcy0j | human alveolar epithelial cell psf retroviral vector cell |
| 4cvy9u28 | mast cell,human mast cell,pancreatic mast cell |
| ajlctjeb | neutrophil |
| x7wg7e9s | growth prostate cancer epithelial ovarian cancer cell line |
| 1axxmq84 | density progenitor cell proliferate cell |
| cc5thj1g | epithelial cell,intestinal epithelial cell |
| 7gmtp6km | primary macrophage dendritic cell |
| fpsyem7x | primary endothelial cell,primary fibroblast,endothelial cell |
| 104sqoxz | plasma cell |
| 533xlisc | spheroid neural stem cell |

Figure 4.4: *Cell Line entities, grouped by article's id*

### 4.1.3 Topic Extraction

Topic extraction is a Machine Learning (ML) technique which is used to identify the conceptual topics that may be discussed in a collection of documents. In our proposed methodology, topics are extracted through Latent Dirichlet Allocation (LDA) [34] which is an unsupervised machine learning model. In LDA topics are produced based on the words that occur frequently in every document. Each topic is presented as a list of representative words (keywords) along with their probability.

Before we explain how the LDA algorithm works, we briefly present three principal LDA input parameters $k$, *alpha* and *beta*. Parameter $k$ indicates the number of topics that the corpus is likely to include. The $k$ estimation can be done either empirically or by measuring topic coherence. Topic coherence indicates the relative distance between two words that are included in a topic. To find the optimal topic value, we have to train many LDA models with different $k$. The optimal $k$ is the one that gives the best coherence score. The parameter *alpha* determines the number of topics expected in a document. Respectively, *beta* determines the word distribution that is expected in a topic.

Given a collection of documents, we describe how LDA identifies topics and topic distributions. Initially, the algorithm assigns randomly all the document words to one of the $k$ topics. It then passes every word and every topic assignment to calculate $a$) how frequently a topic occurs in each document and $b$) how frequently a word occurs in a topic. Based on this information, LDA assigns again all the document words to a new topic. This process is repeated several times until the topics start making sense. Consequently, LDA extracts topics and then describes each document as a probability distribution of these topics.

After topic extraction, our system corresponds topic keywords to real-world entities that have been extracted from NER and already exist in the database. Topic keywords that match an already existing entity, are then removed along with their probability and replaced with a relationship (See paragraph 5.2.4 relationship Probability) between topic and entity.

In Table: 4.4 we present an example of topic extraction. In the example we see that a scientific publication with title: "*Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Arabia.*" can be described as a distribution of three potential topics:

$$0.06570756 * Topic\_1 + 0.82845414 * Topic\_2 + 0.029987121 * Topic\_3$$

Topic keywords are descriptive enough to understand the main topic subject. According to the first topic keywords (sample, test, positive, pcr) we understand that Topic 1 concerns the process of virus detection. Respectively, the second topic (child, respiratory, severe, symptom, infection) discusses respiratory infections and their symptoms in children. Finally, the third topic (pandemic disease, countries, outbreak, measures) concerns the pandemic outbreak. According to the topic probabilities, we assume the publication mainly addresses the problem of respiratory infections and their symp-

| Title: Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Arabia | |
|---|---|
| Topic | Probability |
| $0.022 * sample + 0.020 * positive + 0.019 * detection + 0.018 * results + 0.017 * study + 0.016 * testing + 0.014 * test + 0.013 * pcr + 0.012 * methods + 0.011 * rt\_pcr$ | 0.06570756 |
| $0.098 * patient + 0.022 * hospital + 0.019 * child + 0.016 * clinical + 0.015 * respiratory + 0.015 * severe + 0.014 * care + 0.011 * symptom + 0.011 * day + 0.010 * infection$ | 0.82845414 |
| $0.121 * covid + 0.019 * cases + 0.016 * pandemic + 0.014 * disease + 0.011 * number + 0.010 * country + 0.010 * outbreak + 0.009 * measures + 0.009 * epidemic + 0.009 * health$ | 0.029987121 |

Table 4.4: *Example of three potential topics describing a specific document*

toms (with probability 0.82845414). It also discusses the topics of virus testing and pandemic measures.

## 4.2 Model Design

Graph data modeling is the process of describing a specific domain as a graph consisting of nodes and relationships. In order to design a data model, we need a domain, nodes to represent the domain concepts, and relationships to describe the interaction between these concepts. How each concept will be represented in graphs can vary depending on the designer's aim[10]. For example, someone may decide to model a concept as a property of a node or as a relationship to another node depending on the queries which will run.

Instead of reusing subsets of already existing vocabularies, we designed a data model that meets the needs of the academic field as a whole. Our main purpose was to design a general model in such a way as to support either general or very specific information. This way we overcome potential limitations that would have occurred in the case of using pre-existing ontologies. Limitations regarding the use of pre-existing ontologies usually concern the specific nature of these ontologies or the inconsistency between the available data and the ontology structure. We therefore propose the UA-data model which is illustrated in Figure: 4.5. The UA-data model has been designed to incorporate the fundamental concepts of the scientific domain and as can be seen in Figure: 4.6 consists of classes, instances and relationships. Additionally, the model supports class hierarchies, is flexible for future enhancement and is domain-independent.

[10] https://neo4j.com/developer/modeling-designs/

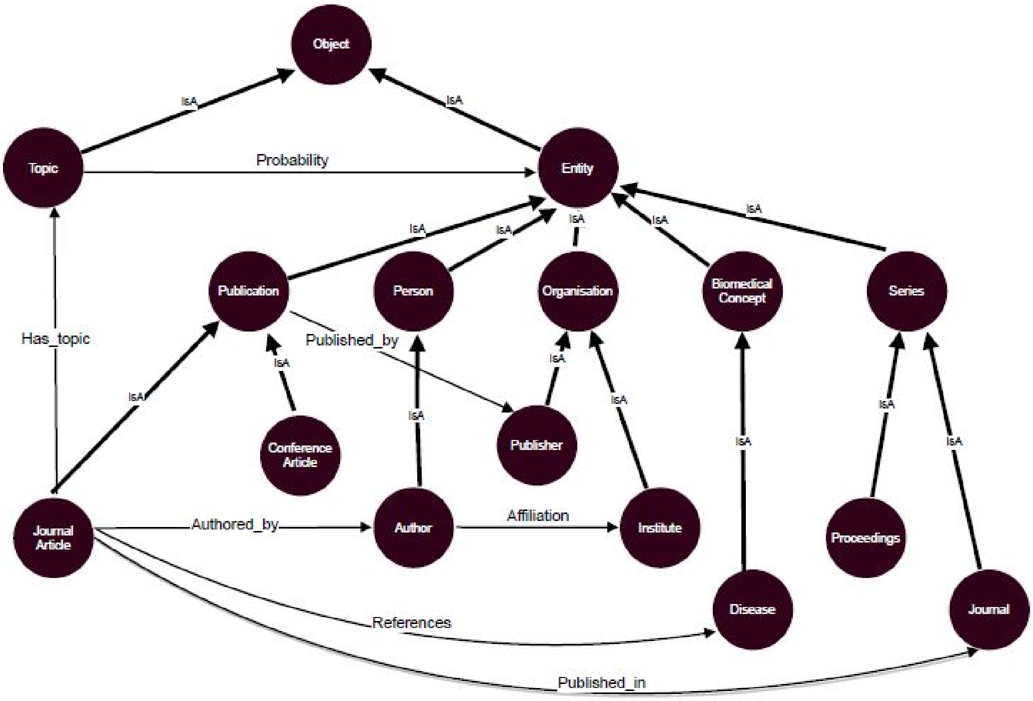


Figure 4.5: *The UA-data model, designed to cover the main concepts of the scientific domain*

### 4.2.1 Domain Concepts

The UA-data model has been designed to satisfy the requirements of the scientific domain as a whole. We briefly explain the main concepts of the scientific domain upon which we have based the design of the UA-model classes. Scientific publications are the main components of the scientific domain. Through publications, researchers can communicate the latest accomplishments in their field. Furthermore, research results are publicly available in such an authoritative way that their validity is not contestable. The scientific community can then evaluate each papers' quality and build upon the given work. Each scientific publication is written by qualified authors who are connected with an academic institution or a research center. Therefore, authors and academic institutions are two more important components of the scientific field.

In academia, there are various types of publications of which the academic journal is the most common. In this model, we include published information from both journals and conference articles. Each publication refers to a specific scientific discipline or a combination of several disciplines from humanities to natural sciences and others. Between the main scientific components, we also introduce the respective scientific discipline, though without limiting the model potentials. This is why we

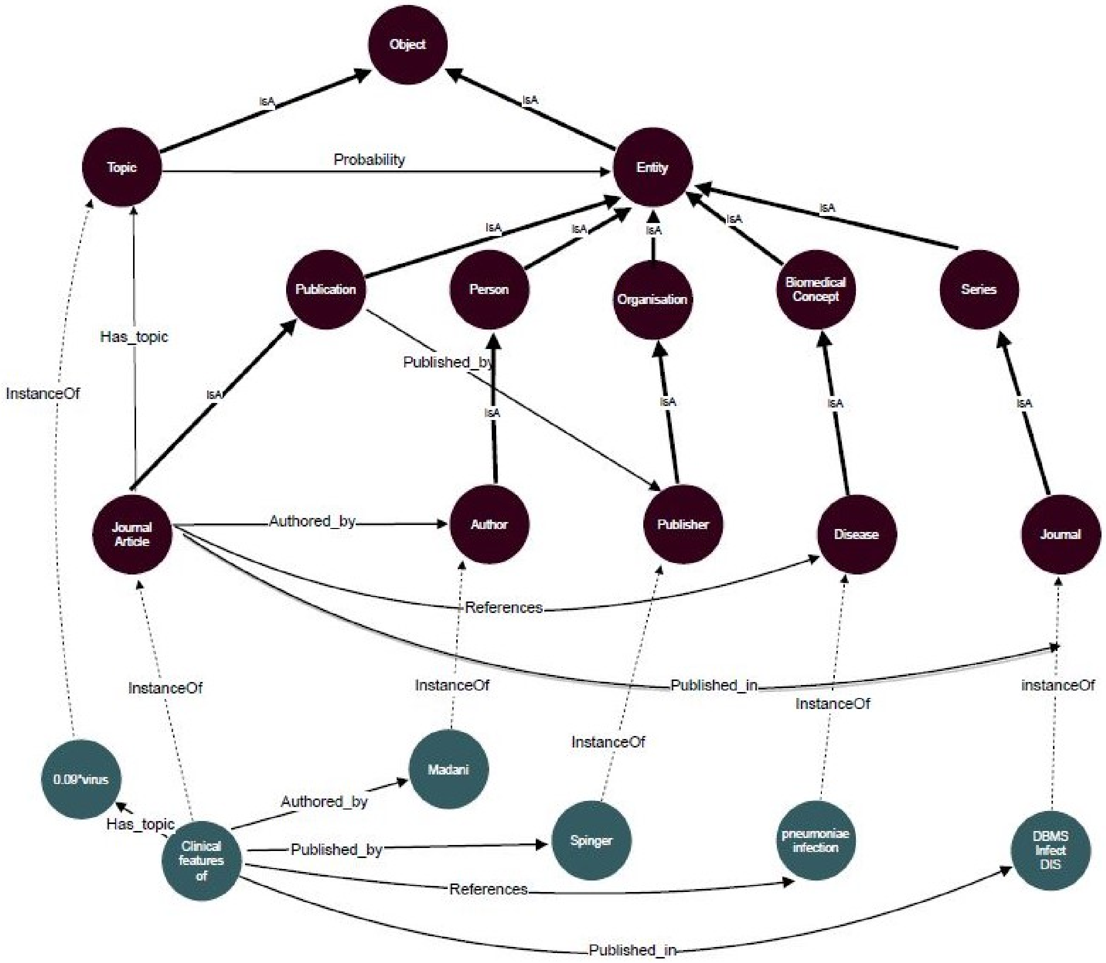


Figure 4.6: *The UA-data model and an example of class instances*

state that our model is domain-independent. For each publication, we add topics and concept entities described within its text. Based on the main concepts of the scientific domain, we present the primitive concepts of the UA-data model which are classes, instances and relationships.

### 4.2.2 Classes

Our graph data model corresponds the main components of the scientific domain (see par. 4.2.1) to classes. We use nodes and directed relationships to represent the model in a property graph. Each node represents a class, which may have attributes and can also be associated with other classes. The UA-data model is a hierarchical model that consists of four levels. The main class represents the `Object`, while its two subclasses (`Topic, Entity`) are created so as to provide supplementary information. In turn, these two subclasses have subclasses of their own (`Publication, Person, Organization, Series, Biomedical_Concept`) and so on until we reach the lowest level (`Publisher, Author, Journal_Article, Conference_Article, Proceedings, Journal, Institution`). The classes of the UA-data model inherit their attributes and properties to their subclasses. The hierarchical relationship between a class and its subclass is described by way of the `isA` relationship, which is a directed relationship. This means that the parent-child hierarchy is shown by a directional arrow that starts from the child Class and ends in the parent Class. The model contains 15 main classes overall as shown in the Table: 4.5. Below we present a detailed description of the UA-model classes and their respective attributes.

### Metamodel Classes

***Object***: Class Object is the superclass. Object models any concept that may exist in the database. In a sense it represents the "microcosm" of the database and whatever is there is subject to it.

***Entity***: Class Entity is a subclass of Object and models all the individual units that may exist in the real world. It could be a person, a location, or anything else that has real-world substance.

***Topic***: Class Topic is a subclass of Object which models all the possible topics described in each scientific publication. The attribute of this class is a string containing all of the important topic keywords, along with their probability.

### Model Classes

***Publication***: Class Publication is a subclass of Entity and includes all the scientific articles' instances. The attributes of Class Publication are id, title, publication date, DOI.

***Journal Article***: Journal_Article is a subclass of Class Publication. This Class models all the Publications published in journals.

***Conference Article***: Class Conference_Article is a subclass of Publication. This Class models all the Publications presented at conferences.

***Person***: Class Person is a subclass of the Class Entity and includes all the instances of people who may be referred to in the dataset. The attribute of this Class is the person's name.

***Author***: Class Author is a subclass of Person. This Class models all the people who have authored scientific articles in Class Publication.

***Organisation***: Class Organisation is a subclass of Entity. Organization models all the organizations involved in scientific article production. The attribute of this Class is the organization's name.

***Publisher***: Publisher is a subclass of Organisation. This Class models all the publishing companies which have distributed scientific articles that exist in Class Publication.

***Institution***: Class Institution is a subclass of Class Organisation. This Class models all the institutions where the research for a specific scientific publication was conducted.

***Series***: Class Series is a subclass of Class Entity and contains all the information regarding the publication of research findings.

***Proceedings***: Class Proceedings is a subclass of Class Series and models all the academic conferences where the research behind the publications was presented.

***Journal***: Class Journal is a subclass of Class Series. This Class contains all the instances of academic journals that present scientific accomplishments.

**Domain Classes**

Domain Classes are all those classes that model the concepts of a specific scientific discipline. As our paradigm we use the Biomedical domain, therefore in this model, the main domain class is the Biomedical_Concept Class. This model however can be adjusted according to future needs. We further add auxiliary classes, in order to categorize the available entity types. For example, the class Disease contains all the entities extracted from NER, with entity type Disease.

***Biomedical_Concept***: This Class is a subclass of Entity Class, represents a specific domain and contains all the entity categories that have been extracted from the data during NER. Our paradigm is the biomedical domain, therefore, Biomedical_Concept models all the biomedical entity types identified in NER.

***Disease***: Disease is an example class. This Class models a particular entity type and includes all the entities of this specific entity type that have been extracted from

NER.

| Model Classes | | |
|---|---|---|
| | **Object**{type: Class} | |
| **Topic**{<br>type: Class<br>isA: Concept<br>dim*: Entity<br>dim*: String} | | **Entity**{<br>type: Class<br>isA: Object } |
| **Publication** {<br>type: Class<br>isA: Entity<br>title: String<br>author*: Person<br>affiliations*:Org<br>in: Series<br>publisher: Pub<br>year: Int<br>month: Int} | **Person**{<br>type: Class<br>isA: Entity<br>name: String}<br>**Organization**{<br>type: Class<br>isA: Entity<br>name: String} | **Series**{<br>type: Class<br>isA: Entity<br>name:String}<br>**Institute**{<br>type: Class<br>isA: Organization}<br>**Bio_Concept**{<br>type: Class<br>isA: Entity<br>name:String} |
| **Publisher**{<br>type: Class<br>isA: Org}<br>**Author**{<br>type: Class<br>isA: Person} | **Journal_Article**<br>type: Class<br>isA: Publication<br>series*: Series<br>**Conference_Article**{<br>type: Class<br>isA: Publication<br>series: Series} | **Proceedings**{<br>type: Class<br>isA: Series}<br>**Journal**{<br>type: Class<br>isA: Series} |

Table 4.5: *The main classes of the model along with their attributes as presented hierarchically from highest to lowest level*

### 4.2.3 Instances

The classes of the UA-data model may have individual objects called instances. When classes have instances, they assign all their attributes to instances. The relationship between Class and Instance is described through the `instanceOf` relationship. In this directed relationship, the arrow starts from the instance of the Class and ends in the Class. Instances can also have relationships with other instances. An example of class instances is shown in Figure: 4.6.

### 4.2.4 Relationships

A significant problem was whether to model a concept as a class attribute or as a relationship to another class. Given the proposed model, we see that the class Publication has attributes: id, title, DOI and Publication_date. The authors of an individual publication instance are included in the class Author and are connected through a relationship that shows authorship. However, author names could have been included in the attributes of the class Publication. The decision of whether to model a concept as an attribute or relationship depends on the nature of the queries as well as their complexity. For example, by modeling authors as attributes of class Publication we can run a simple query like the following:

- ***Return all the authors of publication with id = "ug7v899j"***

We however, would like to answer queries like:

- **Return the authors working in a specific institution**
- **Return all the authors who have researched Ovarian cancer**

Based on this reasoning we modeled as relationships all the concepts presented below. Again these are directed relationships and the arrow direction depends on the case.

***Authored_by***: This relationship connects each scientific publication of Class Publication with the instances of Class Author and indicates authorship.

***Published_by***: This relationship connects each scientific publication of Class Publication with the publishers meaning that the respective paper was published by a specific publisher.

***Has_topic***: This relationship connects each scientific publication of Class Publication with the topics included in the Class Topic indicating all the possible topics described in each publication.

***Probability***: This relationship connects each instance of Class Topic with a specific Entity. It is an optional relationship and exists only in cases where keywords describing the topic have been already detected in the current biomedical concepts.

***Affiliation***: This relationship connects authors with institutions.

***Referred***: This relationship connects each scientific publication of Class Publication with a Biomedical concept. Referred relationship indicates that a publication refers to a specific concept within its text.

***Published_In***: This relationship connects each scientific publication of Class Publication with a specific journal, instance of Class Journal.

## 4.3 Knowledge Graph Implementation

Neo4j[11] is an open-source, NoSQL graph database that is based on the labeled property graph model [35]. This means that data are organized as nodes, relationships and properties. Moreover, the property graph representation approach is flexible for future enhancement and modification by allowing users to add new nodes and relationships over time. We designed the UA-data model in such a way that is flexible for future enhancement. Therefore, we believe that Neo4j is the ideal graph DBMS to implement our model.

### Neo4j Main Concepts

In order to structure and store data Neo4j uses four basic structures[12] [35]: nodes, relationships and relationship types, properties, and labels as explained below.

**Nodes**

Nodes are used as a means to store entity information. Given a data model which represents a field of the scientific domain, possible examples of nodes could be publications, authors, and institutions. The simplest form of a graph is a single node graph. Conversely, a complex graph does not necessarily consist of many nodes, however these nodes present high complexity between their relationships.

**Relationships**

Relationships connect nodes between them and determine the complexity of the graph. Additionally they represent the main way to attribute structure to entities. A relationship can be self-referencing/looping, but should always start and finish at a node.

**Relationship Types**

Each relationship should have only one relationship type. Based on the scientific domain example, two possible relationship types are written_by, work_in :

$$\text{Publication}-\text{written\_by} \rightarrow \text{Author}$$
$$\text{Author } -\text{worked\_in} \rightarrow \text{Institution}$$

Relationships should always have a direction. The node where the relationship starts is called the source node while the one which received the action is called the target node. The reversal of a relationship is unnecessary unless it is required for a specific reason.

[11] https://neo4j.com/developer/graph-database/

[12] https://neo4j.com/docs/getting-started/current/graphdb-concepts/

**Properties**
Nodes and relationships may have properties. Properties are pairs of values $< attr, value >$ used to add characteristics to nodes and their relationships respectively. In our example publication may have:

$$< title, value >, < doi, value >$$

and the relationship:

$$-\text{written\_by}\rightarrow \text{ may have } < date, value >$$

A property value can either be a number, a string or a boolean.

**Labels**
Labels are used as a means for categorizing nodes. All nodes sharing the same label belong to the same category. This means that when Neo4j needs to perform transactions over a specific group of nodes, it can perform them directly by requesting the particular category that the nodes belong to.
Possible labels for the scientific domain example could be:

$$: \text{Authors}, : \text{Publications}, : \text{Institutions}$$

In such a case we can easily request all the nodes with the label, for example : Authors that share a specific characteristic. After the introduction of labels, new potentialities were revealed for Neo4j, as operations difficult to perform in the past can be achieved in a much more straightforward way. By using labels, subgraphs can be quickly created and the type property has been replaced. Furthermore, there is no need to connect nodes to definition nodes which were regularly used to provide supplementary information. Nodes can have zero to multiple labels depending on the way one may want to structure the graph. In a simpler structure nodes can have one label, but when dealing with many different dimensions, attributing multiple labels to the nodes is admittedly an effective suggestion. Lastly, as labels can be easily added and removed at anytime, they can be used temporarily to show the current state of the nodes. Relations have only one label.

### 4.3.1 Challenges

The process of implementing the UA-data model using Neo4j was quite challenging and went through various stages of testing. During this process, we had to correspond the data model primitive components (class instances and relationships) to Neo4j structures and capabilities. Below we present all the challenges we had to handle during the UA-data model implementation in Neo4j and the approaches that we finally followed in order to produce the UA-Graph.

**Challenge 1: Hierarchical class representation and querying in Neo4j**
The first challenge concerns the hierarchical class representation and querying. Neo4j does not propose a particular way to structure hierarchies nor does it understand the meaning behind classes and instances. Classes and instances can be represented by nodes and relationship types, however, querying such a structure from the top (superclass) to the bottom (instance) is impossible.

An example of an idea that was implemented but finally rejected was to create a label for every class. Then we rendered multiple labels at each node based on the classes that the node belongs to (based on Figure: 4.5, for example, a node with the label :Author, should also have the label :Person, :Entity, and :Object). However, the UA-Graph needs to support many hierarchy levels. This implies the need to attribute multiple labels at each node. Such a modeling approach made querying very difficult. By using multiple labels we can easily search for all the nodes that indeed belong to a specific category, but the reverse is impossible. A query like this: ***Return all the nodes with label :Person that are NOT authors*** returns nothing. This query cannot be answered because the database recognizes only the nodes that belong to a specific category. We, therefore, concluded that a multi-labeling suggestion would be better suited to a model that supports fewer hierarchy levels or answers simpler queries and so we focused on a different approach.

An indicative way to represent hierarchies, proposed by Neo4j manual is to base our model on Neosemantics[13] (n10s). Neosemantics is a Neo4j plugin that supports the use of Resource Description Framework (RDF) and among many other tasks also promotes basic inferencing and reasoning. RDF is a general approach used to represent both data and data models to a computer understandable form. Neosemantics allows users to insert and export RDF data without losing information in the process. Furthermore, it allows the import and export of ontologies and taxonomies in different vocabularies.

Concerning the inferencing-reasoning part, this plug-in allows Neo4j to understand information that is not intentionally stored. More precisely, in our case, we need to create nodes that represent classes, subclasses and instances and connect them with relationships showing hierarchies. However, the meaning behind this hierarchical structure is unknown to Neo4j. The main idea is to let the tool distinguish the nodes classes from the nodes instances. We, therefore, expect that Neo4j will be able to recognize the hierarchical model and return both classes, subclasses, and instances when it is asked, even though the instances are not directly connected with their superclasses. Hence Neosemantics recommends a method to represent a model in a

[13] `https://neo4j.com/labs/neosemantics/4.0/inference/`

way that the graph DB can understand even though some concepts are not explicitly described.

Following the Neosemantics manual, we modeled class hierarchies as follows. To model a hierarchy of classes (or categories as referred to in Neosemantics manual) nodes are used to represent the classes, which are then connected through a relationship that shows the hierarchy. We considered the word `isA` a representative word in order to show the class and subclass hierarchy. After declaring the class hierarchy, we had to add the instances of each class. For instances Neosemantics proposes two options. The first can be implemented by adding an additional label to the instance node showing the class it belongs to. The second can be implemented by linking node instances with the class they belong to using a relationship. Finally, Neosemantics proposes specific commands to query the graph built upon the above-mentioned convention (see section 4.3.4).

**Challenge 2: Superclass - Subclass Inheritance in Neo4j**

In the UA-data model, all the attributes and properties of the superclass are inherited to its subclasses. Neo4j, however, does not recognize the superclass-subclass concept and therefore cannot inherit class attributes to other classes.

As the superclass-subclass concept is not recognized by Neo4j, we had to proceed to the following agreement. Every class of the UA-data model is represented by a unique Neo4j node. Therefore, each node's purpose is to "imitate" a model class. The nodes representing classes in Neo4j, however, do not have actual class attributes. We agreed that the actual class attributes are assigned to the nodes that represent the instances of the respective class.

**Challenge 3: Superclass - Subclass Relationships in Neo4j**

In the UA-data model, all the superclass relationships are inherited to its subclasses. For example, in Figure: 4.5 we see that a publication is published by a publisher. This information passes to the subclass journal article. This means that the instances of the class journal article are published by a publisher even though their class is not directly connected with the class publisher.

As the superclass-subclass concept is not recognized by Neo4j, we had to proceed to the following compromise. We, therefore, decided that every class of the UA-data model is represented by a unique node in Neo4j. The class nodes in Neo4j do not include the actual class attributes neither are connected with other classes (except their hierarchical classes). The actual class attributes and relationships are assigned to the nodes that represent the instances of the respective class.

### 4.3.2 Classes and Instances in Neo4j

In this section, we show how we can implement a hierarchical structure in Neo4j based on the proposed model and Neosemantics' principles. Nodes that represent classes and subclasses are connected through the relationship type `isA`. After declaring the class hierarchy, we add the instances of each class. According to Neosemantics we link the nodes-instances with their classes through the relationship type `instanceOf`. Below we implement a small hierarchical model in Neo4j which consists of two classes A, B and an instance C as presented in Figure: 4.7. In this model, class B is a subclass of A and C is an instance of class B. The example is written in Neo4j's graph query language, cypher[14].

**Preliminaries:**
We create two types of nodes (or labels: Classes and Instances). We need to create a unique node property constraint.

```
CREATE CONSTRAINT ON (c:Class) ASSERT c.id IS UNIQUE;
CREATE CONSTRAINT ON (i:Instance) ASSERT i.id IS UNIQUE;
```

**Creating Class A:**
To create a class A we create a node that represents the class (using the label :Class) with the name A and a unique id that we provide. This can be done with the following command:

```
CREATE (a:Class {name: "A", id:"1"})
```

**Class A Properties:**
A property of a class is modeled in Neo4j as a node property. This means that the command that is defining the class A could also contain the specification of the property $< attr, value >$. The command would be:

```
CREATE (a:Class { name: "A", id: "1" , attr: "value"})
```

**Creating Class B, subclass of A:**
An isA hierarchy is created by considering a special relationship type (isA) between

[14] https://neo4j.com/developer/cypher/

the nodes that represent the classes. As we have already created the class A, this time we create only the class B and the isA relationship type.

```
MATCH  (a:Class{name: "A", id: "1"})
CREATE (b:Class{name: "B", id:"2"})
CREATE (b)-[:isA]->(a)
```

**Creating C an instance of B:**
To create an instance of a class B we create a node with label :Instance and then we connect this node with the class B using the special (instanceOf) relationship. In particular, the command will be:

```
MATCH (b:Class{name:B})
CREATE (i:Instance { name: "C", id: "3" })
CREATE (i)-[:instanceOf]->(b)
```

**Instance properties:**
If an instance needs to have some properties $< attr, value >$ this is declared during its creation time. For example, the creation of the instance C will be:

```
CREATE (i:Instance { name: "C", id: "3", attr: "value" })
```

**A relationship type between classes A and B:**
To create the relationship type P between classes A and B, we do so by adding a directed edge [P] between the node of A and the node of B. The edge direction depends on the case.

```
MATCH (a:Class{name: "A"})
MATCH (d:Class{name: "D"})
CREATE (b)-[:P]->(a)
```

**A relationship type between instances:**
A relationship type between instances is defined in the same way it is defined in the case of classes. For example, to define the relationship type [P] from the instance node i to the instance node i2, the command would be:

```
MATCH (i:Instance{name: "C"})
```

```
CREATE (i2:Instance{name: "i2"})
CREATE (i1)-[:P]->(i2)
```

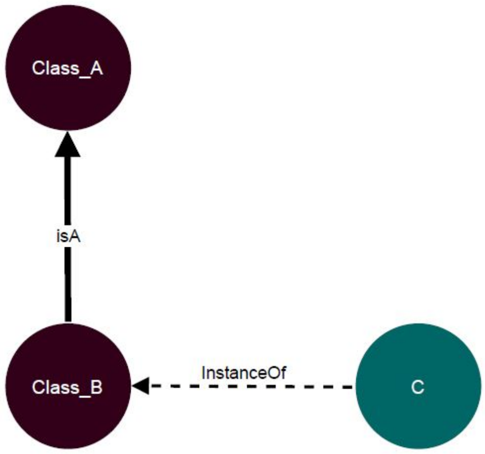


Figure 4.7: *An example of a small hierarchical model that consists of class A, B, and instance C. Class B is a subclass of A and C is an instance of class B*

### 4.3.3 UA-Graph Implementation in Neo4j

The UA-data model implementation in Neo4j required a lot of experimentation and went through various stages of testing. Despite Neo4j's potentialities, we had to proceed to many compromises in order to produce a graph true to the model. These compromises concern structural components of the graph and should always be followed in the case of the UA-Graph reproduction. We implement the UA-Graph using Cypher, Neo4j's graph query language according to the following conventions:

- Every class corresponds to a unique node. We use the nodes-classes in order to create a structure that follows the UA-data model. Every unique node that represents a class, has a specific label

  :Sci_Domain (Scientific Domain abbreviation)

  All the nodes with label :Sci_Domain have two attributes: {name, id}. The hierarchy of classes is established according to Neosemantics. Nodes that represent the model classes are connected through a relationship that shows their hierarchy. We selected `isA` as a representative relationship type in order to show the class and subclass hierarchy. In Table 4.6 we present all the nodes that correspond to the model classes along with their attributes. Further in Figure: 4.8 we present a Neo4j example that includes all the nodes-classes and their hierarchies.

- What follows is the need to create the instances of each class. Class instances are nodes that contain the actual data. We attribute to the instances of each class a label related to the name of the class (For eg. to label the instances we used the name of the class in plural: Class author → has instances with label Authors, Class publisher → has instances with label Publishers, etc.). Instance - class relationship is shown by the `instanceOf` relationship type.
- Currently, Neo4j does not support inheritance between Classes. This means that a node defined as a superclass does not inherit attributes and properties to its subclasses. Therefore, the actual data are represented by nodes defined as instances at the lower hierarchical level. The relationship types exist only in the case of instances, as relationship types between classes do not add valuable information.

Below we present all the labels used to categorize the nodes instances (i.e. the actual data):

### Labels

***:Journal_Articles***: We attribute this label to all the nodes that represent the actual Journal articles included in our dataset.
All nodes with label **:Journal_Articles** are connected with with the node **:Sci_Domain {name: Journal_Article}** thought the instanceOf relationship type.

***:Conference_Articles***: We attribute this label to all the nodes that represent the actual Conference articles included in our dataset. All nodes with label **:Conference_Articles** are connected with with the node **:Sci_Domain {name: Conference_Article}** thought the instanceOf relationship type.

***:Authors***: We attribute this label to all the nodes that represent the actual Authors included in our dataset. All nodes with label **:Authors** are connected with with the node **:Sci_Domain {name: Author}** thought the instanceOf relationship type.

***:Journals***: We attribute this label to all the nodes that represent the actual Journals included in our dataset. All nodes with label **:Journals** are connected with with the node **:Sci_Domain {name: Journal}** thought the instanceOf relationship type.

***:Publishers***: We attribute this label to all the nodes that represent the actual Publishers included in our dataset. All nodes with label **:Publishers** are connected with with the node **:Sci_Domain {name: Publisher}** thought the instanceOf relationship type.

***:Conferences***: We attribute this label to all the nodes that represent the actual Conferences included in our dataset. All nodes with label **:Conferences** are connected with with the node **:Sci_Domain {name: Proceedings}** thought the instanceOf relationship type.

***:Institutions***: We attribute this label to all the nodes that represent the actual Institutions included in our dataset. All nodes with label **:Institutions** are connected with with the node **:Sci_Domain { name: Institution}** thought the instanceOf relationship type.

**Relationship Types**

Relationship Types correspond to the UA-model relationships as described in paragraph 5.4.2.

***Authored_by***: This relationship type connects each scientific publication node with the nodes that represent its authors.

***Published_by***: This relationship type connects each scientific publication node with the node that represents the respective company that has published this paper.

***Has_topic***: This relationship type connects each scientific publication node with the nodes that represent all the possible topics described in each publication.

***Probability***: This relationship type connects each topic node with a specific biomedical concept node. It is an optional relationship and exists only in the case that keywords describing the topic have been already detected in the current biomedical entities.

***Affiliation***: This relationship type connects nodes representing authors with the institutions they collaborate.

***Referred***: This relationship type connects each publication node to the biomedical entity node indicating that this concept is referred in this specific document.

***Published_In***: This relationship connects each publication node with the node representing the journal that published the respective publication.

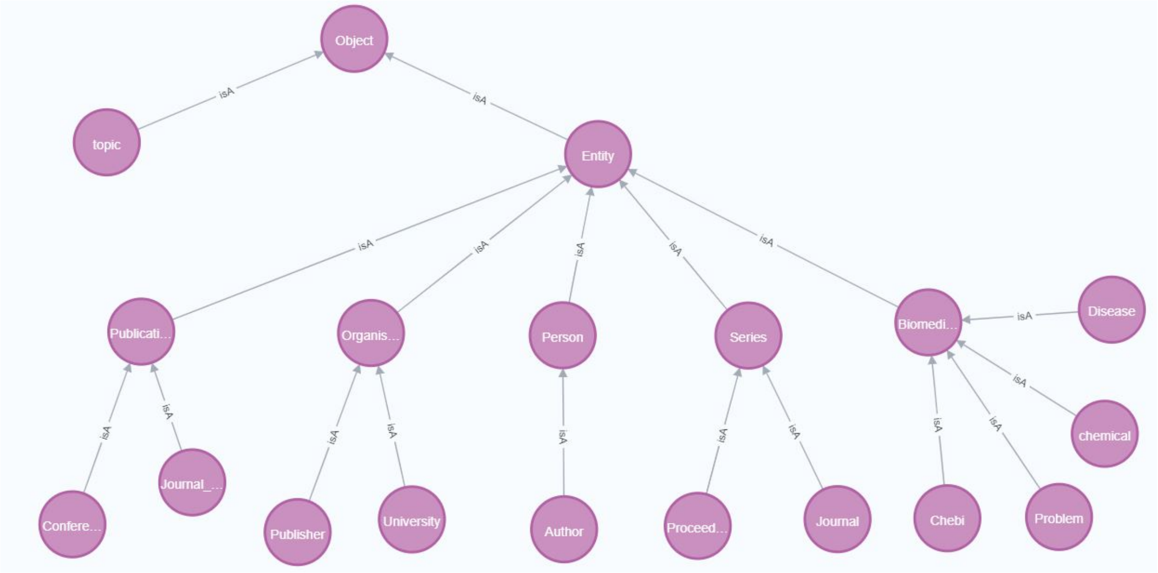


Figure 4.8: *A Neo4j example that shows all the nodes with label Sci_Domain that represent the model classes*

| Nodes with label Sci_Domain | | |
|---|---|---|
| Node | Description | Properties |
| Object | Represents the Supercalss Object | <name: Object, id: ObjectId> |
| Topic | Represents the Class Topic | <name: Topic, id: TopicId> |
| Entity | Represents the Class Entity | <name: Entity, id: EntityId> |
| Publication | Represents the Class Publication | <name: Publication, id: PublicationId> |
| Person | Represents the Class Person | <name: Person, id: PersonId> |
| Organisation | Represents the Class Organisation | <name: Organisation, id: OrganisationId> |
| Conference_Article | Represents the Class Conference_Article | <name: Conf_Article, id: Conf_ArticleId> |
| Journal_Article | Represents the Class Journal_Article | <name:Journal_Article, id:Journal_ArticleId> |
| Biomedical_Concept | Represents the Class Biomedical_Concept | <name:Bio_Concept, id:Bio_ConceptId> |
| Author | Represents the Class Author | <name:Author, id: AuthorId> |
| Series | Represents the Class Series | <name:Series, id: SeriesId> |
| Journal | Represents the Class Journal | <name:Journal, id: JournalId> |
| Proceedings | Represents the Class Proceedings | <name:Proceedings, id: ProceedingsId> |
| Institution | Represents the Class Institution | <name:Institution, id: InstitutionId> |
| Publisher | Represents the Class Publisher | <name:Publisher, id: PublisherId> |

Table 4.6: *Classes to nodes correspondence*

### 4.3.4 Querying Hierarchies in Neo4j

This section analyzes the UA-Graph hierarchy querying process. Neo4j at this point encounters the graph as a set of nodes that have labels and relationships. In the example described in paragraph 5.3.4 Class B isA Class A and C instance Of class B. The main challenge is to access the instance C, not only from B but also from its superclass A. Querying a graph that consists of three nodes may seem straightforward, however, considering the UA-model top-down querying is impossible. The superclass cannot communicate with its subclasses so a simple query that is asking for instance categories like:

- ***Which are the categories of publications?***

cannot be answered.

To alleviate this problem we query the graph using Neosemantics as explained in par 4.3.1. To query the graph hierarchically we use two main Neosemantics procedures n10s.inference.nodesInCategory and n10s.inference.inCategory. These procedures take as input two parameters:

- subCatRel: Which requests information about the class hierarchy (in our case:

subCatRel=isA).

- inCatRel: Which requests information about the way instances are connected to the class (in our case: inCatRel=instanceOf).

Below we present two indicative examples to better explain querying with Neosemantics.

**Preliminaries:**
Neosemantics[15] (n10s) plug-in needs to be installed in Neo4j.

**Query 1: Get the nodes in a particular category:**
Use n10s.inference.nodesInCategory to get the nodes in a particular category.

```
MATCH (cat:Sci_Domain { name: "Entity"})
CALL n10s.inference.nodesInCategory(cat,
{ inCatRel: "instanceOf", subCatRel: "isA"})
yield node
return node.title as title, labels(node) as categories;
```

The Cypher query returns a list of articles as shown in Table: 4.7. The query returns articles even though they are not directly connected to the node Entity.

| Title | Categories |
|---|---|
| “Factors affecting translation at the programmed 1 ribosomal frameshifting site of Cocksfoot mottle virus RNA in vivo” | [“Articles”] |
| “ Protein secretion in Lactococcus lactis : an efficient way to increase the overall heterologous protein production” | [“ Articles”] |

Table 4.7: *Articles returned by the procedure n10s.inference.nodesInCategory*

**Query 2: Verify whether a node belongs to a particular category:**

Create inference parameters

```
:param inferenceParams =>
({ inCatRel: "instanceOf", subCatRel: "isA"});
```

Querying using n10s.inference.inCategory()

[15] https://neo4j.com/labs/neosemantics/installation/

```
MATCH (cat:Sci_Domain {name: "Object"})
MATCH (:Articles { id : "t35n7bk9"})-[:In]->(b:Journals)
WHERE n10s.inference.inCategory(b,cat,$inferenceParams)
RETURN b.name as name;
```

In Query 2, we requested the journal that published the Article with id: *t35n7bk9*, as a subcategory of Object. In Table 5.2 we see that journal “Retrovirology” is the returned result, even though :Journals are not directly connected to Object.

| Name |
|---|
| “Retrovirology” |

Table 4.8: Journal returned by the procedure n10s.inference.inCategory

# Chapter 5

# UA-Graph evaluation

In this section, we evaluate the quality of the produced KG. The evaluation method consists of two parts. The first part concerns the KG production using real-world data. In the second part, we prove that the produced graph can indeed answer queries which in a different data model would be more difficult to answer. Moreover, through a concise exploratory data analysis, we show that querying the UA-graph can inspire users towards further exploration.

## 5.1 KG Production using Real-World Data

### 5.1.1 The CORD-19 Dataset

To evaluate the produced KG, we used abstracts and metadata from Cord-19 [7], the Covid open research dataset. Cord-19 is created by the Allen Institute for AI and offers a plethora of scientific publications related to corona viruses in general. Since December 2019, the literature has mostly focused on Covid-19. Data are collected from four different sources (Pubmed, World Health Organisation, bioRxiv, medRxiv) and usually include abstracts, full-text content, authors, journals, unique identifiers, and other fields.

### 5.1.2 KG Production using the CORD-19

To produce the UA-graph we used abstracts and metadata from 381817 Cord-19 scientific publications. Of these, only 269232 have available English abstracts. Concerning the NER part, we extracted all the available Scispacy and Stanza entity types presented in section 4. From med 7 we extracted six drug-related entities (drug, dosage, duration, form, frequency, route). The entity type strength was not detected in the Cord-19 abstracts. Regarding Topic extraction, we had to find the optimal topic number $k$. Following the directions presented in par.4.1.3 we trained 30 LDA models and examined the coherence score for every model individually. We then used PyLDAvis library [36]. PyLDAvis is a python library that allows the LDA model examination in a two-dimensional plane. Each topic is presented as a circle in the plane. A good topic model gives us similar, non-overlapping circles which are spread over the four quadrants of the plane. By combining the above-mentioned approaches we concluded that $k = 17$ is the optimal topic number for the Cord-19 corpus. Information and code regarding the UA-graph production are stored in the UA-Graph project repository[1]. The initial dataset along with the extracted information (topics, entities) are stored in the UA-Graph data. Quantitative information about the nodes and the relationship types of the UA-Graph are presented in the Table: 5.1 and Table: 5.2.

| Nodes | Count |
|---|---|
| Journal Articles | 360861 |
| Authors | 1035838 |
| Journals | 29325 |
| Biomedical Concepts | 590322 |
| Topics | 494 |
| Classes | 52 |

Table 5.1: *All nodes of the UA-Graph*

| Relationship Type | Count |
|---|---|
| Published In | 339974 |
| Authored by | 2035608 |
| Refers | 2522752 |
| Has Topic | 1600808 |
| instanceOf | 2122134 |
| isA | 51 |

Table 5.2: *All the UA-Graph relationship types*

In Figure 5.1 we present a small subgraph part of the UA-Graph. As it can be seen from the illustration, the scientific publication (Blue node) with title "*Increased expression of CD8 marker on T-cells in COVID-19 patients*" covers several Biomedical Concepts (Beige nodes). To show the correspondence between scientific text and the extracted entities, we cite below a part of the publication's abstract. We highlight the entities that have been extracted from the text and are presented in the graph.

*"BACKGROUND: Cell-mediated immunity including T-cells (T helper and cytotoxic) plays an essential role in efficient antiviral responses against coronavirus disease-2019 (COVID-19). Therefore, in this study, we evaluated the ratio and expression of CD4 and CD8 markers in COVID-19 patients to clarify the immune characterizations of*

[1] https://github.com/Irosfouggari/RM-THESIS---The-UA-Graph

*CD4 and CD8 T-cells in COVID-19 patients. METHODS: Peripheral blood samples of 25 COVID-19 patients and 25 normal individuals with similar age and sex as the control group were collected. White blood cells, platelets, and lymphocytes were counted and CD4 and CD8 T lymphocytes were evaluated by flow cytometry.*"

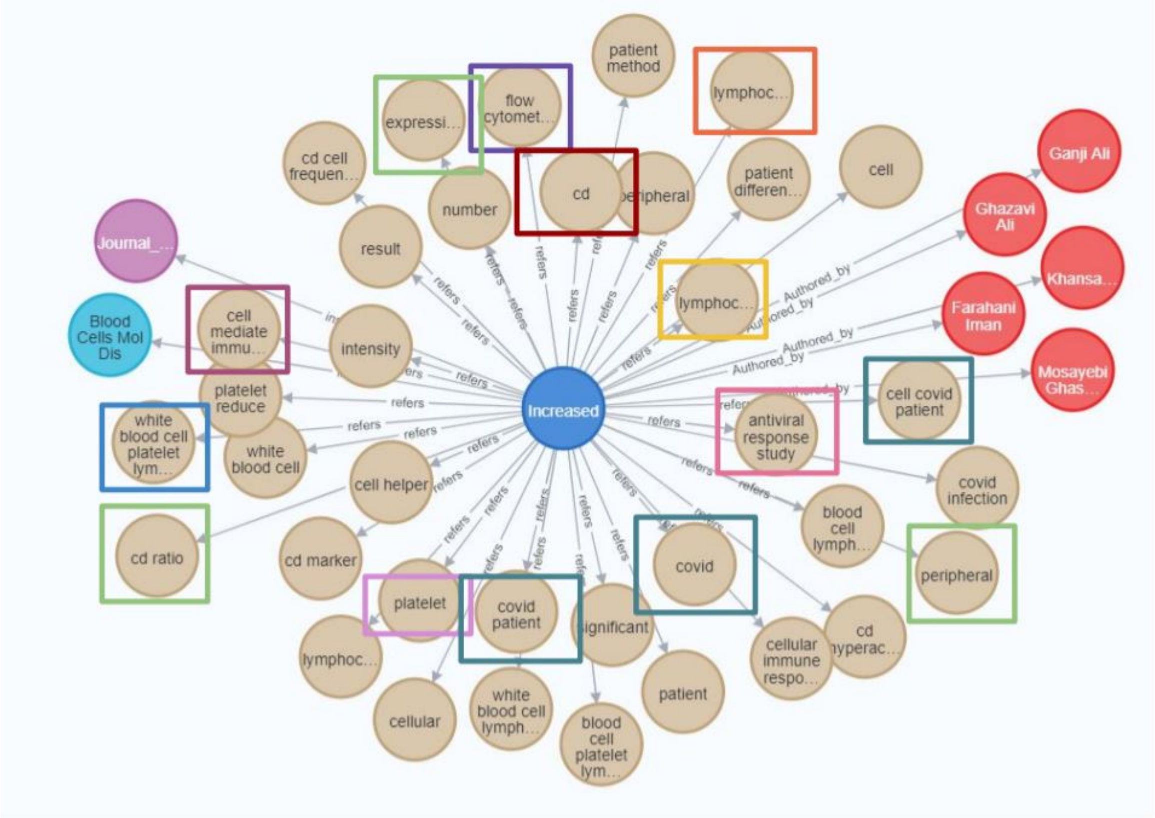


Figure 5.1: *A small part of the UA-Graph. The Journal Article (Blue node), is connected to Biomedical Concepts (Beige nodes). The article has Authors (Red nodes), is published in a Journal (Light blue node) and is an instance of Journal Article Class (Purple node).*

Respectively, in Figure 5.2 we present another small subgraph part of the UA-Graph. As it can be seen from the illustration, the scientific publication (Blue node) with title "*Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Arabia*" discusses five possible topics (Pink nodes). As explained in section 4.1.3, our system corresponds topic keywords to domain (biomedical) concepts that already exist in the database. In the example below, we see that the node Patient (Beige node - label :Biomedical_Concepts) is connected with two topic nodes. This means that the concept patient used to be a topic keyword but finally removed as it was already in the database. The keyword's probability became a property of the relationship type Probability.

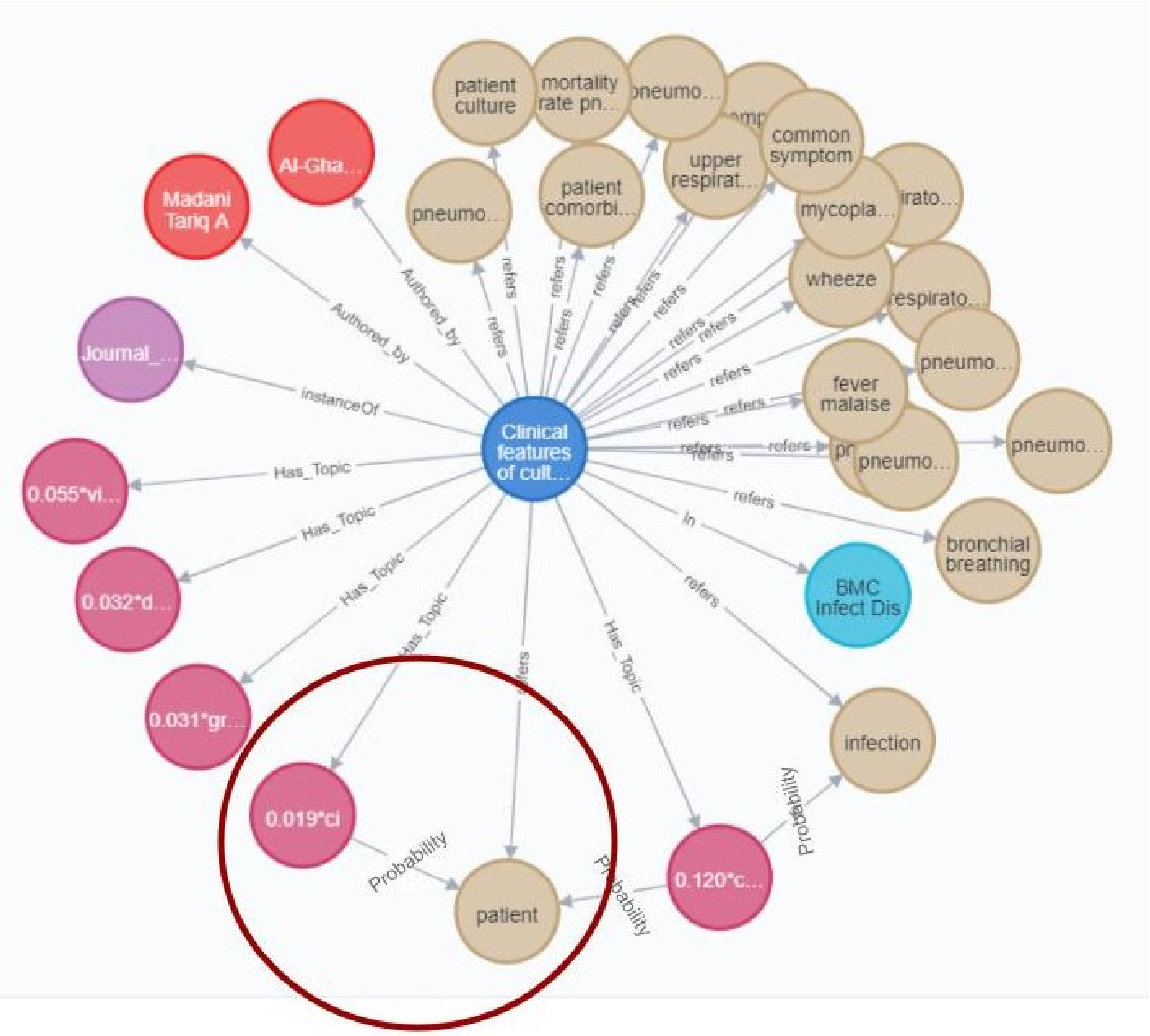


Figure 5.2: *A small part of the UA-Graph. The Journal Article (Blue node), is connected to Biomedical Concepts (Beige nodes). The article has 5 possible topics (Pink nodes) and is an instance of Journal Article Class (Purple node). The relationship type: Probability connects a specific concept with a topic. This relationship shows that the concept is included among the topic keywords.*

## 5.2 Querying the UA-Graph

### 5.2.1 Indicative Queries

The flexible data model of the UA-Graph facilitates querying in cases where a different model would not. To prove this, we present a sequence of queries that can be answered efficiently by the UA-Graph. An interesting idea would be to search for the most popular Journals in publishing scientific articles. Such a question can be easily answered by the UA-Graph using simple Cypher commands as can be seen in Query: 1. This query returned three journals. The "bioRxiv" comes first with 3621 articles.

*Query 1: Return the Journals that have published the majority of the articles in the DB.*

```
MATCH (a:Articles)-[r:In]->(j:Journals)-
[:instanceOf]->(j2:Sci_Domain {name:"Journal"})
RETURN j,j2, count(DISTINCT r) AS num
ORDER BY num  DESC limit 3
```

| Journals | Counts |
|---|---|
| bioRxiv | 3621 |
| BMJ | 3254 |
| PLoS One | 3133 |

Let us assume that we want to identify the most popular concept of a specific biomedical category (anatomical system). This can be answered easily by the following query. The biomedical concept “cardiovascular” seems to be the most popular part of anatomical system in the knowledge base.

*Query 2: Return the most popular instance of Anatomical System.*

```
MATCH (a:Articles)-[r:refers]->(b1:Biomedical_Concepts)-
[:instanceOf]->(b2:Sci_Domain {name:"Anatomical_System"})
RETURN b1,b2, count(DISTINCT r) AS num
ORDER BY num  DESC limit 1
```

| BC: Anatomical System | Counts |
|---|---|
| cardiovascular | 3161 |

In the same concept, we can identify the two most discussed diseases. According to the result, covid surpasses infection which comes second in a row. This is an expected result as the graph mostly concerns covid-related literature.

*Query 3: Return the two most popular diseases.*

```
MATCH (a:Articles)-[r:refers]->(b1:Biomedical_Concepts)-
[:instanceOf]->(b2:Sci_Domain {name:"Disease"})
RETURN b1,b2, count(DISTINCT r) AS num
ORDER BY num  DESC limit 2
```

| Disease | Counts |
|---|---|
| covid | 102050 |
| infection | 40701 |

Another interesting idea would be to search for the subject that a specific researcher usually writes about. For example, Weissbrich Benedikt is mostly interested in viruses and patients.

*Query 4: Return a Biomedical Concept that a specific researcher usually writes about.*

```
MATCH (a1:Authors{name:" Weissbrich Benedikt"})
<-[r1:Authored_by]-(a2:Articles)-[r2:refers]->
(b1:Biomedical_Concepts)-[:instanceOf]->(b2:Sci_Domain)
RETURN a1,b1, count(DISTINCT r2) AS num
ORDER BY num  DESC limit 2
```

| Biomedical Concept | Counts |
|---|---|
| virus | 6 |
| patient | 4 |

Respectively, the graph allows users to find researchers who have written about a specific topic. In the following example authors: Htun N. N. and Turski Lechoslaw in their articles (see table below), mention the concept of “thromboembolism”.

*Query 5: The name of the researchers who are interested in thromboembolism along with the respective publication.*

```
MATCH (a1:Authors)<-[r1:Authored_by]-(a2:Articles)
-[r2:refers]- (b1:Biomedical_Concepts{name:"thromboemboism"})
RETURN a1.name as Author,a2.title as Title,b1.name as Concept
```

| Author | Title |
|---|---|
| Htun N. N. | “Diabetic Ketoacidosis in Coronavirus Disease Patients With Type 2 Diabetes Mellitus” |
| Turski Lechoslaw | “AhR and IDO1 in pathogenesis of Covid-19 and the ”Systemic AhR Activation Syndrome:” a translational review and therapeutic perspectives” |

Another interesting idea, would be to find a concept that is usually included in the articles published in a specific journal. For example journal “PLoS Pathog” uses to publish articles dealing with viruses and cells.

*Query 6: Which is the most popular Biomedical concept published in PLoS Pathog.*

```
MATCH (a1:Journals{name:"PLoS Pathog"})<-[r1:In]-(a2:Articles)
```

```
-[r2:refers]->(b1:Biomedical_Concepts)
-[:instanceOf]->(b2:Sci_Domain)
RETURN a1,b1, count(DISTINCT r2) AS num
ORDER BY num  DESC limit 2
```

| Biomedical Concept | Counts |
|---|---|
| virus | 323 |
| cell | 219 |

Finally, it is straightforward to find articles that include a specific term in their title and dealing with specific concepts. In the following example, we are looking for articles the include the term “myocarditis” in their title but also mention the concept of death inside their text content.

*Query 7: Which articles include the word “myocarditis” in their title and are related to the concept “death”.*

```
MATCH (n:Articles)-[:refers]->(b1:Biomedical_Concepts{name:"death"})
MATCH (n)-[Authored_by]->(a:Authors)
WHERE n.title =~ '.*myocarditis.*'
RETURN n,b1 AS num
```

| Article |
|---|
| “Management perspectives from the 2019 Wuhan international workshop on fulminant myocarditis” |
| “Case report: high-grade atrioventricular block in suspected COVID-19 myocarditis” |
| “First report on fatal myocarditis associated with adenovirus infection in Cuba” |

### 5.2.2 Exploratory Data Analysis

The exploratory data analysis in the case of the UA-Graph aims to highlight the capacities of the generated graph instead of providing statistics about the KG entities. The proposed analysis is based on the user’s needs and the queries used to produce an interesting result. Therefore, we asked a professional clinician to explore the UA-Graph. Along with the clinician we created a query scenario in which each subsequent query is guided by the result of the previous one. We show that the UA-Graph allows easy content navigation and gives direct answers to complex queries.

Below we present all the Cypher queries that have been used for this concise exploratory analysis.

### Query scenario: Corona virus related myocarditis

An interesting topic for research could be Corona virus-related myocarditis. With the following cypher query we receive all the journal articles that include the word “myocarditis” in their title. Within the UA-Graph there may be publications that do not include abstracts. In the following query we request all those articles that are connected to biomedical concepts. If an article is connected to biomedical concepts, this usually means that this article has a publicly available abstract.

*Query 1: Return all the articles that are connected with biomedical entities and contain the word myocarditis in their title*

```
MATCH (a:Articles)-[:refers]-(b:Biomedical_Concepts)
WHERE a.title =˜ ’.*myocarditis.*’
RETURN a.title as Title,b.name as Entity
```

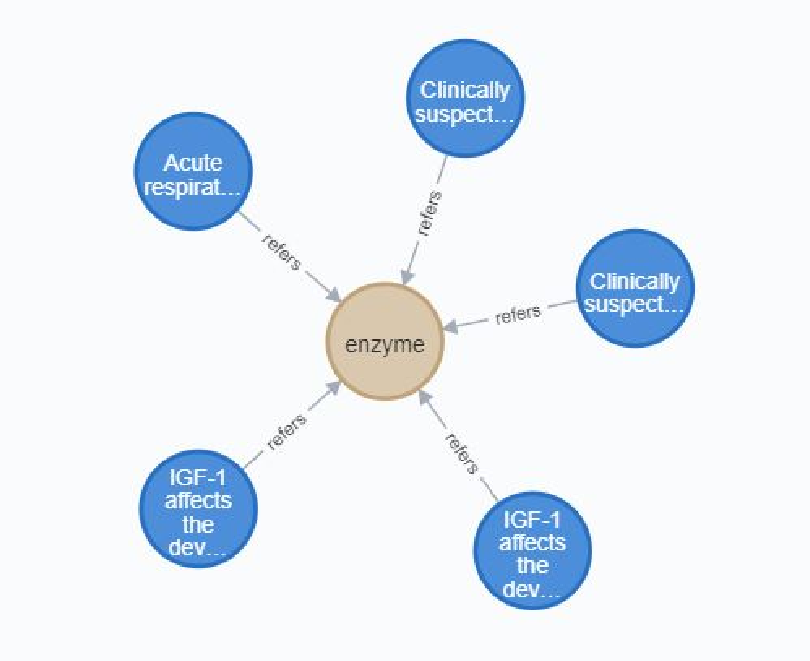


Figure 5.3: *Articles (Blue nodes) that contain the term “myocarditis” in their title and are all connected to the concept “Enzyme” (Beige node).*

The query returned a plethora of scientific publications that include the word “myocarditis” in their title. We selected only 5 of them and explored the biomedical concepts that each publication is related to. As can be seen in Figure: 5.3, among the biomedical concepts, the “Enzyme” had a dominant presence. The following step was to check the entity type of the concept Enzyme. Therefore, we ran the following query.

*Query 2: Which is the entity type of the biomedical concept enzyme*

```
MATCH (b1:Biomedical_Concepts{name:"enzyme"})-[:instanceOf]->
(b2:Sci_Domain)
RETURN b1,b2
```

According to the KG enzymes are proteins. Therefore, the next step is to find the most "popular" biomedical concepts that belong to Proteins (instances of Protein). In the UA-Graph the entities are categorized by the entity type they belong to. Therefore it is very easy to find the biomedical concepts of a specific category.

*Query 3: Return the two most popular Proteins*

```
MATCH (a:Articles)-[r:refers]->(b1:Biomedical_Concepts)-
[:instanceOf]->(b2:Sci_Domain {name:"Protein"})
RETURN b1, count(DISTINCT r) AS num
ORDER BY num  DESC limit 2
```

According to the KG, enzyme is the second most popular protein. The proteins troponin and cardiac troponin were also included into the 40 most discussed Proteins that are included in the graph. Troponin concept is referred by 325 publications.

*Query 4: How many connections does the biomedical concept Troponin has.*

```
MATCH (b1:Biomedical_Concepts{name:"troponin"})
WITH b1, SIZE(()-[:refers]->(b1)) as authorCnt
ORDER BY authorCnt DESC LIMIT 10
MATCH (a)-[:refers]->(b1)
RETURN count(a)
```

According to the professional, the role of troponin seems to be significant in corona virus. Patients with high troponin are tend to be more severe Covid-19 cases. Finally, we decided to look for articles that include the word "myocarditis" in their title and also include the concepts "troponin" and "death" in their content. The KG returned only one publication.

*Query 5: Return all the publications that refer troponin and death.*

```
MATCH (n:Articles)-[:refers]->
                  (b1:Biomedical_Concepts{name:"troponin"})
MATCH (n:Articles)-[:refers]->
                  (b2:Biomedical_Concepts{name:"death"})
MATCH (n)-[Authored_by]->(a:Authors)
WHERE n.title =~ '.*myocarditis.*'
```

```
RETURN n,b1,b2,a
```

| Title | Authors |
|---|---|
| “Case report: high-grade atrioventricular block in suspected COVID-19 myocarditis” | Loke Wei Ian, Ashok Vishnu |

# Chapter 6

# Conclusions and Future work

In this project, we presented all the steps leading to the UA-Graph production. The UA-Graph is a scientific KG, which has been created to represent publication metadata and information extracted from scientific text. The UA-Graph production consists of three individual steps: i) the production of a methodology that processes data and extracts information that can be used as input to the graph ii) the design of a data model that represents the concept of scientific domain and iii) the implementation of the data model using a graph DBMS. We, therefore, propose a semi-automatic methodology that processes data using several NLP techniques. It then detects entities using NER models and finally uses LDA to identify possible topics discussed in the corpus. Furthermore, we propose a flexible data model that uses classes and instances to model the concepts of the scientific domain. We then insert each publication in the graph, along with its metadata and other relevant information extracted from the proposed methodology. Our model and methodology are domain-independent. This means that both are designed to cover the needs of any scientific field.

To evaluate our methodology and graph we used abstracts and metadata from the CORD-19 dataset. We show that taking advantage of its flexible model, the UA-Graph allows easy content navigation and can answer complex scientific questions.

## 6.1 Future Work

In conclusion, we believe that UA-Graph can motivate future research.
Regarding the NER part:

- Professionals can annotate scientific text in order to extract entity types that are not included in the existing models.

- After NER, Professionals can check the extracted entities for possible errors. Given the errors, the NER models can be retrained according to the corpus needs.

Regarding the data model implementation:

- Implement the UA-data model according to the actual data - model.

# Appendix A

# Scripts for text processing

Information and code regarding the UA-graph production are stored in the UA-Graph project repository[1].

The following Figure:A.1 shows all the steps that produce the semi-automatics methodology which processes text and extracts structured information. Each individual step corresponds to a python script.

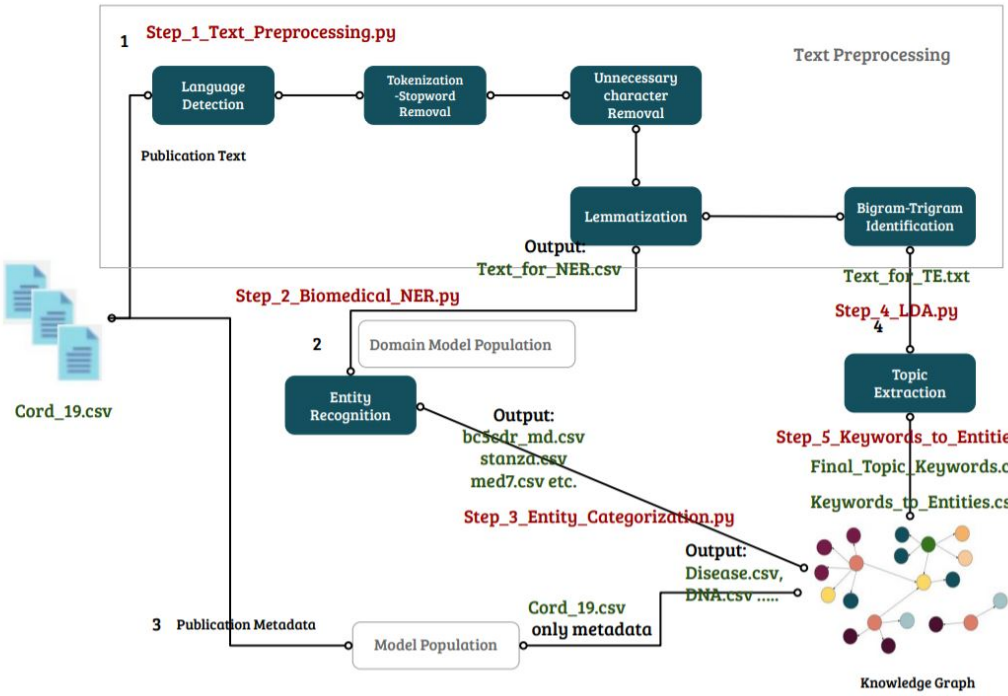


Figure A.1: *A Neo4j example that shows all the nodes with label Sci_Domain that represent the model classes*

[1] https://github.com/Irosfouggari/RM-THESIS---The-UA-Graph

# Appendix B

# Cypher scripts

Below we present all the cypher scripts used to produce the UA-Graph.

Unique property constraints ensure that property values are unique for all nodes with a specific label.

```
CREATE CONSTRAINT ON (sd:Sci_Domain) ASSERT sd.id IS UNIQUE;
CREATE CONSTRAINT ON (jas:Articles) ASSERT jas.id IS UNIQUE;
CREATE CONSTRAINT ON (journals:Journals) ASSERT journals.id IS UNIQUE;
CREATE CONSTRAINT ON (authors:Authors) ASSERT authors.name IS UNIQUE;
CREATE CONSTRAINT ON (topics:Topics) ASSERT topics.tid IS UNIQUE;
```

Then we have to create all the nodes that represent the classes based on the instructions presented in section 4.3.3.

```
CREATE (o:Sci_Domain { name: "Object", id: "ObjectId" })
CREATE (t:Sci_Domain { name: "topic",  id: "TopicId" })
CREATE (e:Sci_Domain { name: "Entity",  id: "EntityId" })
CREATE (p:Sci_Domain { name: "Publication", id:"PublicationId" })
CREATE (per:Sci_Domain { name: "Person", id: "PersonId" })
CREATE (org:Sci_Domain { name: "Organisation", id: "OrganisationId" })
CREATE (bc:Sci_Domain {name: "Biomedical_Concept",id: "Biomedical_ConceptId"
CREATE (ca:Sci_Domain {name: "Conference_Article",id: "Conference_ArticleId"
CREATE (ja:Sci_Domain {name: "Journal_Article",id: "Journal_ArticleId" })
CREATE (a:Sci_Domain { name: "Author", id: "AuthorId" })
CREATE (pub:Sci_Domain { name: "Publisher", id: "PublisherId" })
CREATE (uni:Sci_Domain { name: "University", id: "UniversityId" })
CREATE (ser:Sci_Domain { name: "Series", id: "SeriesId" })
CREATE (j:Sci_Domain { name: "Journal", id: "JournalId" })
CREATE (pro:Sci_Domain { name: "Proceedings", id: "ProceedingsId" })
```

Create the class hierarchies.

```
CREATE (t)-[:isA]->(o)
CREATE (e)-[:isA]->(o)
CREATE (p)-[:isA]->(e)
CREATE (per)-[:isA]->(e)
CREATE (org)-[:isA]->(e)
CREATE (bc)-[:isA]->(e)
CREATE (ca)-[:isA]->(p)
CREATE (ja)-[:isA]->(p)
CREATE (a)-[:isA]->(per)
CREATE (pub)-[:isA]->(org)
CREATE (uni)-[:isA]->(org)
CREATE (ser)-[:isA]->(e)
CREATE (j)-[:isA]->(ser)
CREATE (pro)-[:isA]->(ser)
```

Create all the journal article nodes. Data are stored in a csv file. Then connect all the articles with the Journal they have been published in and finally with their parent class “:Sci_Domain Journal_Article”.

```
:auto using periodic commit 100000
load csv with headers from   \\file....   as line
with line where line.journal is not null
match (p:Sci_Domain {name: "Publication", id:"PublicationId" })
match (j:Sci_Domain { name: "Journal", id: "JournalId" })
merge (js:Journals {name: line.journal})
merge (a:Articles {id: line.cord_uid})
on create set
    a.title =line.title ,
    a.doi = line.doi,
    a.publish_time=line.publish_time
merge (a)-[:In]->(js)
merge (a)-[:instanceOf]->(p)
merge (js)-[:instanceOf]->(j)
```

Create all authors. Connect them with their articles and their parent class.

```
:auto using periodic commit
load csv with headers from   // file ...   as line
merge (articles:Articles {id: line.cord_uid})
on create set
    articles.title =line.title ,
    articles.doi = line.doi,
    articles.publish_time=line.publish_time
with articles,line,split(line.authors, ',') AS authors
match (a:Sci_Domain { name: "Author", id: "AuthorId" })
foreach (f in authors |merge (author:Authors {name: f})
merge (articles)-[:Authored_by]->(author))
foreach (f in authors |merge (author:Authors {name: f})
merge (author)-[:instanceOf]->(a))
```

Create all the biomedical concepts. Connect them with the articles and with their entity type. This script is specifically for biomedical concept: chemical. This should be done individually for every category.

```
:auto using periodic commit
load csv with headers from    // file ...     as line
match (bc:Sci_Domain { name: "Biomedical_Concept",
id: "Biomedical_ConceptId" })
merge (articles:Articles {id: line.cord_uid})
merge (disease:Sci_Domain {name: "Chemical", id: "ChemicalId" })
merge (disease)-[:isA]->(bc)
with articles,line,split(line.Entity, ',') AS entities
match (disease:Sci_Domain {name: "Chemical", id: "ChemicalId" })
foreach (f in entities |merge (entity:Entity {name: f})
merge (entity)-[:IdentifiedIn]->(articles))
foreach (f in entities |merge (entity:Entity {name: f})
merge (entity)-[:instanceOf]->(disease))
return count(entities)
```

Create all the topics. Connect them with the articles and with their parent class.

```
:auto using periodic commit
load csv with headers from    // file ...     as line
match (topic:Sci_Domain { name: "topic", id: "TopicId" }),
(articles:Articles {id: line.cord_uid})
merge (t:Topics {tid: line.rec_id})
merge (articles)-[r:Has_Topic{Probability:line.Probability}]->(t)
merge (t)-[:instanceOf]->(topic)
with t, SPLIT(line.Final_Topic_Keywords,',') AS word
  set t.k1 = word[0], t.k2= word[1],t.k3 = word[2], t.k4= word[3],
  t.k5 = word[4], t.k6= word[5],t.k7 = word[6],
  t.k8= word[7],t.k9 = word[8], t.k10= word[9]
return count(t)
```

This scrip creates relationships between topics and biomedical concepts.

```
:auto using periodic commit
load csv with headers from    // file ...    AS line
merge (t:Topics {tid:line.rec_id})
merge (e:Biomedical_Concepts {name: line.Word})
merge (t)-[:K{Probability:line.Probability}]->(e)
```

# Bibliography

[1] E. M. L. Bergman, “Finding citations to social work literature: The relative benefits of using web of science, scopus, or google scholar,” *The journal of academic librarianship*, vol. 38, no. 6, pp. 370–379, 2012.

[2] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, “Microsoft academic graph: When experts are not enough,” *Quantitative Science Studies*, vol. 1, no. 1, pp. 396–413, 2020.

[3] G. Hendricks, D. Tkaczyk, J. Lin, and P. Feeney, “Crossref: The sustainable source of community-owned scholarly metadata,” *Quantitative Science Studies*, vol. 1, no. 1, pp. 414–427, 2020.

[4] L. Ehrlinger and W. Wöß, “Towards a definition of knowledge graphs.,” *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, pp. 1–4, 2016.

[5] A. Singhal, “Introducing the knowledge graph: things, not strings,” *Official google blog*, vol. 5, 2012.

[6] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods,” *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[7] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. M. Kinney, Z. Liu, W. Merrill, P. Mooney, D. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. B. S. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. A. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier, “Cord-19: The covid-19 open research dataset,” *ArXiv*, 2020.

[8] H. Harapan, N. Itoh, A. Yufika, W. Winardi, S. Keam, H. Te, D. Megawati, Z. Hayati, A. L. Wagner, and M. Mudatsir, “Coronavirus disease 2019 (covid-19): A literature review,” *Journal of Infection and Public Health*, 2020.

[9] M. L. Ranney, V. Griffeth, and A. K. Jha, “Critical supply shortages—the need for ventilators and personal protective equipment during the covid-19 pandemic,” *New England Journal of Medicine*, vol. 382, no. 18, p. e41, 2020.

[10] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, "Named entity extraction for knowledge graphs: A literature overview," *IEEE Access*, vol. 8, pp. 32862–32881, 2020.

[11] C. Welty, "Ontology research," *AI magazine*, vol. 24, no. 3, pp. 11–11, 2003.

[12] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.

[13] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic annotation, indexing, and retrieval," *Journal of Web Semantics*, vol. 2, no. 1, pp. 49–79, 2004.

[14] Z. Huang, J. Yang, F. van Harmelen, and Q. Hu, "Constructing knowledge graphs of depression," in *International Conference on Health Information Science*, pp. 149–161, Springer, 2017.

[15] Q. Wang, M. Li, X. Wang, N. Parulian, G. Han, J. Ma, J. Tu, Y. Lin, H. Zhang, W. Liu, *et al.*, "Covid-19 literature knowledge graph construction and drug repurposing report generation," *arXiv preprint arXiv:2007.00576*, 2020.

[16] X. Wang, X. Song, Y. Guan, B. Li, and J. Han, "Comprehensive named entity recognition on cord-19 with distant or weak supervision," *arXiv preprint arXiv:2003.12218*, 2020.

[17] F. Michel, F. Gandon, V. Ah-Kane, A. Bobasheva, E. Cabrio, O. Corby, R. Gazzotti, A. Giboin, S. Marro, T. Mayer, *et al.*, "Covid-on-the-web: Knowledge graph and services to advance covid-19 research," in *International Semantic Web Conference*, 2020.

[18] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. Kinney, Z. Liu, W. Merrill, *et al.*, "Cord-19: The covid-19 open research dataset," *ArXiv*, 2020.

[19] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th international conference on semantic systems*, pp. 1–8, 2011.

[20] A. Rossanez and J. C. dos Reis, "Generating knowledge graphs from scientific literature of degenerative diseases.," in *SEPDA@ ISWC*, pp. 12–23, 2019.

[21] C. Wise, V. N. Ioannidis, M. R. Calvo, X. Song, G. Price, N. Kulkarni, R. Brand, P. Bhatia, and G. Karypis, "Covid-19 knowledge graph: Accelerating information retrieval and discovery for scientific literature," *arXiv preprint arXiv:2007.12731*, 2020.

[22] "Amazon Comprehend Medical." `https://aws.amazon.com/comprehend/medical/`.

[23] T. Mayer, E. Cabrio, and S. Villata, "Acta: a tool for argumentative clinical trial analysis," 2019.

[24] S. Fathalla, S. Vahdati, S. Auer, and C. Lange, "Towards a knowledge graph representing research findings by semantifying survey articles," in *International Conference on Theory and Practice of Digital Libraries*, pp. 315–327, Springer, 2017.

[25] A. Sadeghi, C. Lange, M.-E. Vidal, and S. Auer, "Integration of scholarly communication metadata using knowledge graphs," in *International Conference on Theory and Practice of Digital Libraries*, pp. 328–341, Springer, 2017.

[26] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. C. Wiegers, and C. J. Mattingly, "The comparative toxicogenomics database: update 2017," *Nucleic acids research*, vol. 45, no. D1, pp. D972–D978, 2017.

[27] A. Aryani, M. Fenner, P. Manghi, A. Mannocci, and M. Stocker, "Open science graphs must interoperate!," in *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pp. 195–206, Springer, 2020.

[28] P. Manghi, A. Bardi, C. Atzori, M. Baglioni, N. Manola, J. Schirrwagen, and P. Principe, "The openaire research graph data model," Apr. 2019.

[29] M. Y. Jaradeh, A. Oelen, K. E. Farfar, M. Prinz, J. D'Souza, G. Kismihók, M. Stocker, and S. Auer, "Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge," in *Proceedings of the 10th International Conference on Knowledge Capture*, pp. 243–246, 2019.

[30] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, vol. 7, no. 1, pp. 411–420, 2017.

[31] M. Neumann, D. King, I. Beltagy, and W. Ammar, "Scispacy: Fast and robust models for biomedical natural language processing," *arXiv preprint arXiv:1902.07669*, 2019.

[32] A. Kormilitzin, N. Vaci, Q. Liu, and A. Nevado-Holgado, "Med7: a transferable clinical natural language processing model for electronic health records," *arXiv preprint arXiv:2003.01271*, 2020.

[33] Y. Zhang, Y. Zhang, P. Qi, C. D. Manning, and C. P. Langlotz, "Biomedical and clinical english model packages in the stanza python nlp library," *arXiv preprint arXiv:2007.14640*, 2020.

[34] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[35] M. Needham and A. E. Hodler, *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, 2019.

[36] C. Sievert and K. Shirley, "Ldavis: A method for visualizing and interpreting topics," in *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pp. 63–70, 2014.