

GOTTFRIED WILHELM LEIBNIZ UNIVERSITÄT HANNOVER

FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK

INSTITUT FÜR VERTEILTE SYSTEME

DATA SCIENCE UND DIGITAL LIBRARIES

Data Science with Scholarly Knowledge Graphs

Masterarbeit

Kamel Fadel

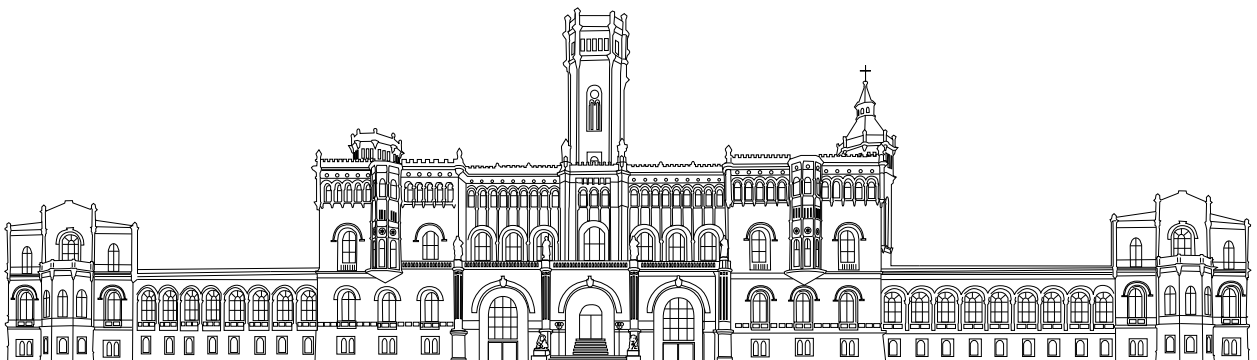
Matrikelnummer: 10016516

Erstprüfer: Prof. Dr. Sören Auer

Zweitprüfer: Dr. Markus Stocker

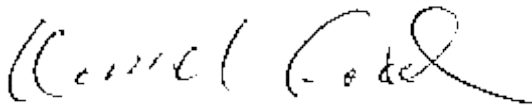
Betreuer: Dr. Markus Stocker

15.03.2021



Erklärung der Selbständigkeit

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen hat.



Kamel Fadel

Hannover, den 15.03.2021

Abstract

Nowadays, scientific articles are mostly published as PDF files containing unstructured and semi-structured text. This way of scholarly communication severely limits the possibilities to automatically process and reuse scholarly knowledge. As a consequence, applying data analysis methods to scientific literature is a non-trivial process. FAIR scholarly knowledge graphs (SKGs) is one approach to represent scholarly knowledge in a structured, machine-actionable, and semantic manner. In this thesis, we exploit SKGs, specifically the Open Research Knowledge Graph (ORKG), in data science. We introduce a generic architecture for applying data science to scholarly data. We then implement the architecture using the ORKG as the main data source and test it in two use cases in different domains. We demonstrate approaches that reuse the ORKG content to draw new insights and build applications and visualizations on top of SKGs data.

Zusammenfassung

Heutzutage werden wissenschaftliche Artikel meist als PDF-Dateien veröffentlicht, die unstrukturierten und halbstrukturierten Text enthalten. Diese Art der wissenschaftlichen Kommunikation schränkt die Möglichkeiten zur automatischen Verarbeitung und Wiederverwendung von wissenschaftlichem Wissen stark ein. Folglich ist die Anwendung von Datenanalysemethoden auf wissenschaftliche Literatur ein nicht-trivialer Prozess. FAIR scholarly knowledge graphs (SKGs) ist ein Ansatz, um wissenschaftliches Wissen in einer maschinenverarbeitbaren, strukturierten und semantischen Weise darzustellen. In dieser Arbeit nutzen wir SKGs, insbesondere den Open Research Knowledge Graph (ORKG), in Data Science. Wir stellen eine allgemeine Architektur für die Anwendung von Data Science auf wissenschaftliche Daten vor. Anschließend implementieren wir die Architektur unter Verwendung des ORKG als Hauptdatenquelle und testen sie in zwei Anwendungsfällen in unterschiedlichen Domänen. Wir demonstrieren Ansätze, die den ORKG-Inhalt wiederverwenden, um neue Erkenntnisse zu gewinnen und Anwendungen und Visualisierungen auf Basis der SKG-Daten zu erstellen.

Acknowledgements

I would like to express my deepest appreciation to my supervisor Dr. Markus Stocker for the continuous support, guidance, patience, and encouragement. Without his persistent help and guidance, this work would not have been possible. I could not have imagined having a better supervisor.

I would like to sincerely thank Prof. Dr. Sören Auer for his support and guidance from day one as I went to his office to discuss an idea for my master thesis.

A big "thank you" also to the members of the Knowledge Infrastructures junior research group.

To my family, thank you for the infinitive support, unconditional love, and trust.

To my wife, Samah, thank you for always being there for me, believing in me, and easing everything with your positivity.

Last but not the least, I would like to dedicate this accomplishment to the soul of my father and my aunt.

Contents

Abstract	ii
List of Figures	3
1 Introduction	4
1.1 Introduction	4
1.2 Motivation	5
1.3 Structure	6
2 Fundamentals	8
2.1 Scholarly data science	8
2.2 Data science pipeline	9
2.3 Scholarly knowledge graphs	11
2.3.1 Scholarly knowledge graph: A definition	11
2.3.2 Difference between relational and graph databases	12
3 Related Work	14
3.1 Scholarly databases	14
3.1.1 Domain-specific SKGs	14
3.1.2 Generic SKGs	17
3.1.3 Relational databases	19
3.2 Information extraction	21
3.3 Data analysis on article metadata	21
3.4 Data analysis on article content	23
4 Scholarly Data Science Architecture and Implementation	25
4.1 Scholarly data science architecture	25
4.1.1 Scholarly data sources	26
4.1.2 Data science environment	27
4.1.3 Hardware infrastructure	29
4.1.4 Outputs	30
4.1.5 Actors	30
4.2 Architecture implementation	31
4.2.1 Scholarly data sources	31
4.2.2 Data science environment	32
4.2.3 Hardware infrastructure	33
4.2.4 Outputs	33

	2
4.2.5 Actors	33
5 COVID-19 Use Case	35
5.1 Methods	35
5.2 SEIR model estimates	36
5.2.1 Estimated vs. observed cases and model performance .	36
5.2.2 Spatial and pattern analysis of model performance . .	38
5.2.3 Spatial and pattern analysis of prevalence	40
5.3 Multiple ORKG Comparisons as data source	43
6 Invasion Biology Use Case	47
6.1 Methods	47
6.2 Importing articles and creating comparisons	47
6.3 Conducting scholarly data analysis	49
6.3.1 Analysis with Python Jupyter Notebook	50
6.3.2 Interactive web application with R Shiny	55
6.3.3 Interactive web page using JavaScript amCharts	57
7 Discussion	59
7.1 Advantages	59
7.2 Limitations	61
7.3 Future work	62
8 Conclusion	63
Bibliography	65

List of Figures

2.1	A typical data science pipeline.	9
4.1	Scholarly data science architecture.	26
5.1	Workflow used for visualization and prediction error calculation.	36
5.2	Number of estimated and observed cases in Chinese provinces.	37
5.3	Workflow of spatial analysis of model performance.	38
5.4	A data frame snapshot with model prediction error in each Chinese province.	39
5.5	Spatial overview of model prediction error.	40
5.6	Clustering of provinces according to population.	41
5.7	Clustering algorithm summary.	41
5.8	Spatial overview of prevalence.	42
5.9	Regional distribution pattern of prevalence.	43
5.10	Workflow of updating the Guardian study.	44
5.11	The contagiousness and deadliness of SARS-CoV-2 relative to other infectious diseases.	45
5.12	The contagiousness and deadliness of SARS-CoV-2 in different locations, relative to other infectious diseases.	46
6.1	Propagule pressure hypothesis.	51
6.2	Total number released.	51
6.3	Propagule size.	51
6.4	Propagule frequency.	52
6.5	Distance from source.	52
6.6	Other proxies.	52
6.7	Hierarchy of the hypothesis.	53
6.8	Number of plant taxa investigated in each article.	53
6.9	Top-10 articles with highest numbers of investigated plant taxa.	54
6.10	Number of non-plant taxa investigated in each article.	54
6.11	Top-10 articles with highest numbers of investigated non-plant taxa.	55
6.12	Number of articles that investigated each taxon.	55
6.13	Distribution of articles over years.	56
6.14	Distribution of articles over continents.	56

Chapter 1

Introduction

1.1 Introduction

Data science gained traction in recent years due to the growing importance of data in knowledge societies. The argument is that it is possible to gain valuable information and insights from data [30].

Usually, each data science use case begins with collecting and organizing data, which takes up most of the data scientist's time¹. In general, the data required for typical data science use cases, such as sales analysis, is stored locally (e.g., in the internal databases of the company). In this case, the time spent on collecting the required data to begin the analysis is minimized. However, conducting data science tasks in other scenarios could be more challenging. For instance, applying data analysis methods and approaches to scientific literature in any research field is a non-trivial process.

Consider the following scenario. A scientist wants to keep pace with the literature analyzing the basic reproduction number (R_0) and case fatality rate (CFR) of COVID-19 in different locations of the world published in recent scientific articles. She wants to visually compare COVID-19 to other viruses in terms of contagiousness (i.e., R_0) and deadliness (i.e., CFR). To do this, our scientist needs a significant amount of time and effort to search for and select the articles that correspond to her needs, extract the relevant knowledge from the articles (i.e., R_0 and CFR), and create a data set. Only then, she can visualize and compare the R_0 and CFR of the different viruses. Depending on the number of articles our scientist reviews, considerable effort may be needed to gather the relevant data. An important part of the problem overall is that the articles are published in PDF format, i.e., knowledge is buried in machine-inactionable format [46]. This severely limits the possibilities to automatically process and reuse scholarly knowledge.

¹<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=6834623b6f63>

Nowadays, almost all scientific articles are communicated in digital forms [33], inactionable PDF documents containing semi-structured and unstructured text. Extracting data manually by humans with expert knowledge and creating structured up-to-date databases and data sets is expensive [60]. Collecting scholarly knowledge this way is a tedious and time-consuming task, and the problem compounds when we need to consider hundreds or thousands of articles.

As the numbers of articles continue to grow, these challenges are not getting any easier and the costs of analyzing and reviewing the literature also increase unless good data management systems are developed. Wilkinson et al. [74] proposed a set of four principles to guide research data management: Research data and metadata should be Findable, Accessible, Interoperable, and Reusable (FAIR), for both humans and machines. By following these principles in publishing and storing articles, any type of research data such as tools, algorithms, and results should be available to other researchers to ensure transparency, reproducibility, and reusability.

Recently, the community has proposed approaches to represent scholarly knowledge in a machine-actionable, structured, and semantic manner. One approach is scholarly knowledge graphs. Knowledge graphs are large networks of real-world entities and relationships, usually expressed in W3C standards such as Web Ontology Language (OWL) and Resource Description Framework (RDF) [75, 23]. Scholarly knowledge graphs (SKGs) are a sub-category of knowledge graphs specialized for the scholarly domain [66]. SKGs may contain metadata that describes articles entities such as authors, institutions, journals, and citation relationships. SKGs also characterize other research knowledge such as methods, technologies, and results [9]. On one hand, SKGs help researchers to share their data in a structured and reusable way. On the other hand, they enable scientists to navigate, analyze, and gain new insights from scholarly data. Examples of scholarly knowledge graphs include the Open Research Knowledge Graph (ORKG)², Microsoft Academic Graph (MAG)³, and OpenAIRE research graph⁴.

1.2 Motivation

Numerous stakeholders benefit from the implementation of the FAIR data principles in research infrastructures, including [74]:

- Researchers wanting to reuse each other's data.

²<https://www.orkg.org/>

³<https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/>

⁴<https://graph.openaire.eu/>

- Data publishers in service provision.
- Software developers building data analysis and processing services on top of scholarly data.
- Data scientists analyzing, mining, and discovering knowledge from scholarly data.

The work presented here relates primarily to the last two points.

One of the challenges in SKGs research area is to develop new services that reuse the data of SKGs to enable exploring the knowledge produced in research fields and drawing new insights⁵.

In this thesis, we tackle this challenge by exploiting SKGs, specifically the Open Research Knowledge Graph (ORKG), in data science. Since the ORKG implements (to a reasonable degree) the FAIR data principles, we demonstrate that the ORKG makes the reuse of scholarly knowledge in data science across literature much easier and more efficient. In other words, we develop and demonstrate approaches that reuse the ORKG content, and possibly data from other sources, to enable data science and discover new knowledge in the research areas of our case studies.

We tackle the following research questions:

- RQ1: Can SKGs, specifically the ORKG, be exploited in data science?
- RQ2: What are the technical approaches to exploit SKGs in data science?

1.3 Structure

We briefly present the structure of this thesis document and its chapters.

Chapter 2: Fundamentals. After introducing and motivating the aim of the thesis, this chapter provides an overview of the basics of scholarly data science, data science pipeline, scholarly knowledge graphs, and the difference between relational and graph databases.

Chapter 3: Related Work. This chapter presents several scholarly databases, their technological characteristics, and how we can use them in data science. Moreover, it provides an overview of existing approaches used for extracting scholarly data from the literature and conducting data analysis, using both data and metadata.

⁵<https://skg.kmi.open.ac.uk/SKG2020/>

Chapter 4: Scholarly Data Science Architecture and Implementation. This chapter introduces a generic architecture for data science on scholarly data, possible implementation choices for the architecture, as well as the specific implementation proposed in this thesis, including the tools and methods used in our work.

Chapter 5: COVID-19 Use Case. The architecture implementation presented in Chapter 4 is tested by performing numerous data science tasks, such as spatial and pattern analysis, on COVID-19 related scholarly knowledge obtained from the ORKG and other sources. Furthermore, the result of a previous study is updated by reusing the ORKG content in Jupyter Notebook.

Chapter 6: Invasion Biology Use Case. The architecture implementation presented in Chapter 4 is tested in an additional research field and other data science tasks. In contrast to the COVID-19 Use Case, here we first ingest data into the ORKG and only then use it in downstream data science. Moreover, in this use case we do not just use Jupyter Notebook but additionally develop interactive web applications.

Chapter 7: Discussion. This chapter discusses the advantages and limitations of our work and outlines future work.

Chapter 8: Conclusion. This chapter summarizes the major contributions of this thesis.

Chapter 2

Fundamentals

This chapter is an overview of the fundamental concepts used throughout this thesis. We first introduce data science in general terms and scholarly data science, specifically. Next we provide an overview of the stages of a typical data science pipeline. We then introduce the concept of knowledge graph, which we specialize to scholarly knowledge graphs. Finally, we present the main differences between relational and graph databases.

2.1 Scholarly data science

Several authors have attempted to define and describe data science. Zhu et al. [81] defined data science as a group of "theories, methods, and technologies for studying data nature". Van der Aalst [67] explained it as an integration of "statistics, data mining, distributed systems and databases" to extract value from data. However, in a more comprehensive definition, data science is a set of various methods including data extraction, preprocessing, analysis, and visualization which is used to gain valuable information and insights for a better understanding of the data. It uses techniques drawn from numerous fields such as mathematics, statistics, and computer science.

During the last years, the considerable growth in the number of published scientific articles has led to sizeable amounts of scholarly data. Scholarly data published in the literature include authors, organizations, institutions, methods, technologies, citation information, research results, and other articles content. This data is being produced and accumulated in all research fields. Reports estimate that a new article is published every 20 seconds¹. Due to the form in which scholarly data is published—primarily, as PDF documents—processing and analyzing scholarly data relies on substantial manual effort. Here is where scholarly data science comes in: applying data science methods on scholarly data extracted from the literature.

¹https://www.explainxkcd.com/wiki/index.php/The_Rise_of_Open_Access

2.2 Data science pipeline

Fayyad et al. [28] presented a process model for knowledge discovery in databases (KDD). The authors used the term KDD to refer to the overall process of extracting useful insights from data. The model consists of five stages (Data selection - Preprocessing - Transformation - Mining - Interpretation and evaluation) and leads from raw data to insights.

Since there is no formally "standardized" process or workflow that fits every data scientist [12], several other efforts introduced workflows and pipelines for data science, data analysis, and knowledge discovery such as [12, 35, 16, 28]. Based on these efforts, a typical data science pipeline, depicted in Figure 2.1, consists of the following stages:

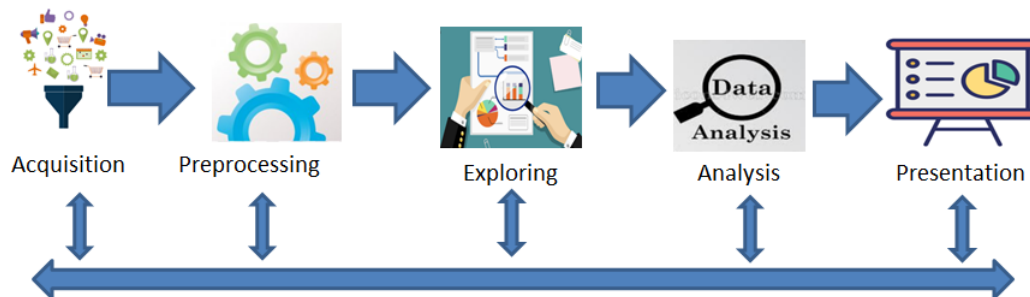


FIGURE 2.1: A typical data science pipeline.

1. **Data Acquisition:** Data is essential to any data science task. Thus, after identifying the task goal, a data scientist selects the relevant sources and starts collecting data [28]. The importance of this step comes from the fact that all the following stages in the pipeline depend on the quality of the data gathered in this stage. The following should be considered when collecting data:
 - Variety of ways to access data (e.g., files vs. APIs) [16].
 - Regulations for access and use of data (e.g., rules for personal data, API limitations).
 - Data formats (e.g., data sets vs. streams, structured vs. unstructured).
2. **Data Preprocessing:** To make data analysis ready, the collected data must typically be processed [35]. Among others, the following concerns need to be considered [76]:
 - Noise: Data may contain outliers or errors (e.g., Height=" -170 cm").
 - Incompleteness: Data may contain missing feature values or certain features of interest.

- **Inconsistency:** Data may contain conflicts in names or values (e.g., Age= "50" and date of birth = "01.01.2000").

For these reasons, data must generally be preprocessed and prepared for subsequent analyses. Once it is understood what each data feature represents, the data preprocessing stage can begin. Major techniques in this stage include [2, 76]:

- **Data cleaning:** Including dealing with missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- **Data integration:** Integrating data from different sources.
- **Data transformation:** Normalization and aggregation.

The result of the data preprocessing stage is data sets ready to be used for exploration and analysis [18].

3. **Data Exploring:** This stage is sometimes called exploratory data analysis (EDA). EDA uses statistics and graphical tools to provide an overview of the available data before applying other analysis methods [14]. For instance, data scientists may inspect the distribution of features by plotting scatterplots to check for skewed data.
4. **Data Analysis:** This stage is about in-depth analysis. Depending on the goals, users choose which methods and algorithms to use such as visualization, spatial analysis, pattern recognition, clustering, classification, regression, etc [82].

The primary goals of data analysis are prediction and description [28, 82]. If it is a predictive analysis problem, then partitioning the data into training and test sets as well as the creation and evaluation of a predictive (possibly a machine learning) model, are done in this stage. On the other hand, descriptive analysis aims to extract new and valuable information, insights, and patterns from data.

5. **Interpretation and results presentation:** The outputs and insights gained in the previous step need to be presented to/for relevant stakeholders [12]. These results may be in form of reports, presentations, applications, and different kinds of visualizations such as maps, charts, etc [12].

While working on each stage in the pipeline, it is often necessary to go back to previous stages and adjust some decisions taken earlier in the pipeline [28]. In other words, the overall process is usually repeated and tuned until the desired results are obtained, Figure 2.1.

Moreover, the time spent working on each stage should be addressed at the beginning of the use case. A 2016 survey² found that data scientists spend most of their time cleaning and organizing data (60%). Collecting data sets is another time-consuming task, as it accounts for about 19% of the work of data scientists. Data scientists spend relatively less time mining data and searching for patterns (9%).

2.3 Scholarly knowledge graphs

We now describe what is understood by knowledge graph and scholarly knowledge graph, as a specialization. We then present the main differences between typical relational databases and graph databases.

2.3.1 Scholarly knowledge graph: A definition

The semantic web aims to create a web of machine-readable knowledge [61]. It has proposed to represent knowledge using graph data models, e.g., using the Resource Description Framework (RDF) [75]. RDF is a framework for describing information about resources. Its key structure is the statement, also known as triple. An RDF statement has the following structure:

```
<subject> <predicate> <object>
```

Sets of RDF triples are called RDF graphs. The graph consists of nodes and edges where nodes are called entities (subjects and objects) and edges are called relations (predicate). Each entity is uniquely identified with an Internationalized Resource Identifier (IRI).

Paulheim [51] suggested a definition for knowledge graph by proposing a set of characteristics that distinguish knowledge graphs. These include:

- KGs describe real-world entities of numerous domains as well as the relations between entities.
- KGs define possible classes and relations between entities in a schema.

With scholarly knowledge graphs (SKGs), this broad characterization of KGs has been specialized to the scholarly domain [66]. A SKG represents scientific information and may not only contain metadata that describes the entities referred to articles such as authors, institutions, journals, etc. but also data that describes the methods, tools, data sets, technologies used in research and the obtained research results [9].

²<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=6834623b6f63>

2.3.2 Difference between relational and graph databases

Relational database management systems (RDBMSs) (such as Oracle and MySQL) are widely used. An RDBMS stores data in tables [17] based on a schema that defines the fixed features and data types of the stored records [40, 5]. The schema is the structure of the database and it should be defined at the database creation time. Relationships between the tables in the RDBMS are built using primary and foreign keys. In general, RDBMSs are efficient unless the data contains many relationships demanding joins of large number of tables [69]. At this point, joining the tables using primary and foreign keys is not a practical solution anymore. Furthermore, the current trend of big data introduces new challenges such as unstructured data, semi-structured data, and fast-changing data [79].

Recently, there has been much interest in non-relational database management systems (NoSQL DBMSs) [31]. NoSQL DBMSs store unstructured and semi-structured data [50, 36]. One type of these DBMSs is graph DBMSs, which store connected data, i.e., data with many interconnected relationships. Graph DBMSs are often used to store knowledge graphs by representing data as nodes and edges [50, 55]. Common graph DBMSs include Neo4j³ and OrientDB⁴.

A primary difference between relational and graph databases is that relational databases have a predefined schema to organize data while graph databases do not enforce a fixed schema [5].

Batra and Tyagi [64] performed a comparative analysis of relational and graph models, in which the performance of Neo4j and MySQL is investigated. The analysis shows that the retrieval of results in graph databases is faster than in relational ones. Furthermore, graph databases are more flexible than relational databases as new data types and relationships can be added to Neo4j without redefining the schema, which is not the case in relational databases. The schema of MySQL needs to be updated with each new data type.

One drawback of Neo4j is the absence of built-in security support [64, 70]. It has Access Control List (ACL) security mechanisms but it is managed at the application layer. On the other hand, many RDBMSs, such as MySQL, contain comprehensive support for ACL-based security.

Another significant downside of Neo4j is the lack of support [70]. Most of the support comes from its parent company's website and is limited from outside the website. On the other hand, the huge popularity of relational databases over the last decades led to increased user support. Relational databases have a unified language which is SQL. Different RDBMSs, such as Oracle and MySQL, have various SQL implementations but SQL does not

³<https://neo4j.com/>

⁴<https://www.orientdb.org/>

differ much between these implementations, i.e., support for one implementation is usually applicable to other implementations. As a result, MySQL, for instance, has comprehensive support from both its parent company (Sun) and its users.

Chapter 3

Related Work

Several databases in various domains contain structured scholarly data that can be used in data science. However, if the required data is not available in existing databases, then extracting it from articles is the first step in the data science pipeline. Thus, the work presented here relates to four aspects:

1. Scholarly databases
2. Extracting structured information from articles
3. Data analysis on article metadata
4. Data analysis on article content

3.1 Scholarly databases

This section is an overview of several scholarly databases. We present their technological differences and how we can use them in data science. Some of the databases are implemented using RDBMS, while others use non-relational database management systems to store data. For the latter we discuss scholarly knowledge graphs (SKGs) hosted on graph databases.

3.1.1 Domain-specific SKGs

Recently, several organizations and research groups worked on constructing domain-specific KGs and used them for tackling problems in the corresponding domains [39].

The CovidGraph project¹ proposed a COVID-19 KG. To build the graph, the project team integrated data from numerous public biomedical sources such as CORD-19² [72] and Lens COVID-19³ data sets. Data from open-source

¹<https://covidgraph.org/>

²<https://www.semanticscholar.org/cord19>

³<https://about.lens.org/covid-19/>

knowledge bases such as the NCBI Gene database⁴ and Gene ontology resource⁵ [6, 15] was also integrated into the graph. The graph data is stored using Neo4j [73] and can be queried using Cypher (an SQL-like query language optimized for graphs). Researchers and users can explore the data stored in the graph by using two web applications. Both applications are free and there is no need for registration.

The first application is called visual graph explorer⁶ and is used for exploring the data visually. The application supports users in finding articles, authors, patents, and genes and facilitates the understanding of the relations between these entities. For instance, users can explore which genes are mentioned in a specific article, and find the articles that mention a particular gene or a combination of genes. To use the visual graph explorer application, users start by selecting an entity type such as genes, patents, articles, authors, diseases, etc. Users can then query against the database by entering a Cypher query or a free text in the toolbar.

The second application is the Neo4j browser⁷. To use the application, users should log in. The credentials are publicly available. The following credentials are valid at the time of writing this thesis:

```
User: public  
Password: corona
```

Users can query the graph directly on the database level via Cypher. It offers a basic visualization of the resulting graph as well as data export in CSV and JSON formats.

Related to the pathophysiology of Corona-Virus, Domingo-Fernández et al. [20] also proposed a COVID-19 knowledge graph. To build the graph, the authors extracted information from both metadata and data of articles. They then transformed this data into Biological Expression Language (BEL) triples and used the triples to build the knowledge graph.

The research community can navigate and analyze the information stored in the graph by using a web application called BiKMi⁸. No registration is needed to use the application. The needed credentials are freely available to log in to the guest's dashboard and explore the statistical summaries of the data available in the graph. The following credentials are valid at the time of writing this thesis:

```
User: guest  
Password: anonymousUser
```

⁴<https://www.ncbi.nlm.nih.gov/gene>

⁵<http://geneontology.org/covid-19.html>

⁶<https://live.yworks.com/covidgraph/>

⁷<https://db.covidgraph.org/>

⁸<https://bikmi.covid19-knowledgespace.de/>

The back-end of the application⁹ is implemented using OrientDB (a NoSQL DBMS that allows both relational and graph queries). Furthermore, users can query the graph directly on the database level by logging into OrientDB and running either their own SQL queries or one of the bookmarked queries. The result of the query is shown in a table and can be exported in JSON files.

In biodiversity, Penev et al. [52] created the OpenBiodiv: an open biodiversity knowledge management system (OBKMS) that exploits text mining and semantic technologies to construct an infrastructure for managing biodiversity knowledge that fulfills the FAIR principles. To build the OpenBiodiv, the authors extracted information from more than 5000 articles and journals. They then transformed this information into RDF (Resource Description Framework) and modeled it in OpenBiodiv-O ontology. Senderov et al. [57] introduced the OpenBiodiv-O ontology with the aim to fill the gaps between the various biodiversity ontologies by focusing on biological taxonomy.

The OpenBiodiv is stored using GraphDB. To interact with OpenBiodiv, users have two options:

1. Via the front-end of the web portal¹⁰: Users with no prior knowledge of SPARQL are supported with a web application, in which they can enter a word or a phrase (e.g., a name of a species or an author) to search for.
2. Directly via the SPARQL endpoint to the database¹¹: Users can run SPARQL queries against the database.

In OpenBiodiv, users can find information about both the metadata and actual content of the articles. While several simple questions can be answered using the front-end web application, more complex questions can only be answered using SPARQL queries. The result of the query can be exported in various formats such as JSON, CSV, TSV, XML, and binary RDF.

The Cooperation Databank (CoDa) is another scholarly knowledge graph in yet another domain, namely human cooperation in social dilemmas. CoDa was developed by Spadaro et al. [59] to explore and analyze the content of the articles related to human cooperation. It is an open-access machine-readable database that contains 2641 annotated articles, theses, and unpublished raw data related to human cooperation. To build the graph, the authors performed first a comprehensive systematic search for relevant studies in English, Japanese, and Chinese in form of articles, book chapters, and doctoral theses. Next, they annotated the concepts and information contained in these studies into RDF formats and modeled them in an ontology. The ontology represents a set of concepts in the cooperation field and the relationships

⁹<http://graphstore.scai.fraunhofer.de/studio/index.html>

¹⁰<http://openbiodiv.net/>

¹¹<http://graph.openbiodiv.net/>

that exist between them. The CoDa knowledge graph supports FAIR principles and is hosted using a TriplyDB graph DBMS. CoDa is available online¹².

Two services enables interacting with CoDa:

1. R-shiny web application¹³: Users can explore the data in CoDa and conduct meta-analysis, meta-regression, and other statistical analyses. There are several tutorials and videos that guide users through the different analyses supported by the application.
2. SPARQL endpoint¹⁴: Users can run SPARQL queries against the CoDa database.

3.1.2 Generic SKGs

As we saw, domain-specific SKGs are rich in data extracted from articles content. Thus, they enable exploring and analyzing actual articles content in specific research areas. In contrast, most of the generic, domain-independent KGs typically model metadata that describes the article entities without domain restrictions. Researchers use these generic KGs for questions that do not require domain-specific knowledge [39], for instance, co-author network analysis. Examples of generic SKGs include the Microsoft Academic Graph (MAG) and the OpenAIRE Research Graph.

Microsoft Research released MAG in 2015. MAG contains structured metadata describing over 250 million article entities and related authors, institutions, and fields of study. Wang et al. [71] described the design and schema of MAG, and how it can be exploited in analytics. To build MAG, software agents with natural language processing capabilities were used to extract metadata from the scholarly literature on the web. The extracted metadata was then organized in MAG, where the nodes represent the entities (such as articles, authors, and conferences) and the edges represent the relationships between these entities.

MAG is available online¹⁵, where users can search for topics, authors, conferences, journals, study fields, etc. Furthermore, users can perform their own analyses and explore the data of the graph online. For example, users can run an analysis at the level of articles¹⁶ and explore the distribution of article types (e.g., books or patents), distribution of articles over years, as well as the top authors, journals, institutions, and conferences in specific topics. Other

¹²<https://data.cooperationdatatabank.org/>

¹³<https://app.cooperationdatatabank.org/>

¹⁴<https://data.cooperationdatatabank.org/coda/datatabank/sparql/datatabank>

¹⁵<https://academic.microsoft.com/home>

¹⁶<https://academic.microsoft.com/publications/>

analyses at the level of authors¹⁷, topics¹⁸, conferences¹⁹, journals²⁰, and institutions²¹ are also available online. The data sets used to generate the visualizations can be downloaded in CSV format. Moreover, for the users who prefer to conduct their analyses in their own environment, regular snapshots of MAG are available for download²².

Furthermore, Färber [25] created the Microsoft Academic Knowledge Graph (MAKG), an RDF version of MAG. MAKG is a large data set of RDF triples of scholarly metadata about authors, organizations, and study fields. A SPARQL endpoint along with query examples are provided online²³.

Another generic SKG is the OpenAIRE Research Graph²⁴. It is an open graph that contains a huge collection of metadata describing articles and links between scientific products such as datasets and software. Schirrwagen et al. [56] presented the curation approaches performed in the OpenAIRE infrastructure. To build the graph, the project team collected metadata from more than 70 thousand community-trusted sources such as PubMed²⁵ and Zenodo²⁶ in a data lake. After that, they transformed this metadata according to the internal metadata model and then generated the graph. In the case of open access articles, the team mined the full text of the articles and obtained links between artefacts. The graph is hosted on an Okeanos super computer. Several applications are available to interact with the graph, including:

- Explore²⁷: Users can search for scholarly works, authors, and projects. Results can be filtered by several factors such as article type or funder and can be exported in CSV files.
- Monitor²⁸: Users can search for organizations and track their research outputs and trends through several charts, maps, and graphs. The retrieved charts and data sets can be downloaded for further analysis.

Moreover, for users who prefer to conduct their analyses in their own environment, the OpenAIRE Research Graph can be exported in several dumps²⁹.

¹⁷<https://academic.microsoft.com/authors>

¹⁸<https://academic.microsoft.com/topics>

¹⁹<https://academic.microsoft.com/conferences>

²⁰<https://academic.microsoft.com/journals>

²¹<https://academic.microsoft.com/institutions>

²²<https://www.microsoft.com/en-us/research/project/open-academic-graph/>

²³<http://ma-graph.org/sparql-endpoint/>

²⁴<https://graph.openaire.eu/>

²⁵<https://pubmed.ncbi.nlm.nih.gov/>

²⁶<https://zenodo.org/>

²⁷<https://beta.explore.openaire.eu/>

²⁸<https://beta.monitor.openaire.eu/>

²⁹https://zenodo.org/record/3516918#.X_8CW0hKg2w

3.1.3 Relational databases

Relational database management systems (RDBMSs) are also widely used to store scholarly knowledge. We present three databases of this type.

Bamford et al. [10] proposed the Catalogue of Somatic Mutations in Cancer database (COSMIC). COSMIC contains somatic mutation information curated manually by experts from more than 27,000 articles from PubMed. To build the database, experts read first the abstracts of the articles and identified relevant ones. They then manually extracted information about mutations, genes, and cancer types and entered it into the database. The COSMIC database is implemented in an Oracle RDBMS. The database is freely accessible³⁰ and it is updated four times a year. Users can search for specific genes, mutations, and cancer types in the database (free text search). The results of the search are shown in tables and can be downloaded in CSV, TSV, and Excel files. Furthermore, the whole database or parts of it can also be downloaded from the website for further analyses.

Additionally, there exist several scholarly databases in the field of protein interactions, such as MINT [80] and MIPS [49].

Zanzoni et al. [80] created the Molecular INTeraction database (MINT). MINT contains structured information about more than 130 thousand interactions extracted from 6024 articles in biological literature. The database is hosted in an SQL server (PostgreSQL) and is freely accessible³¹. Users can query the database via free text search. The retrieved results are shown in a table but can not be exported.

Pagel et al. [49] introduced another database specialized to protein-protein interactions in mammals called MIPS. MIPS contains information about more than 900 proteins extracted manually from more than 370 research articles and stored using a MySQL DBMS. The database is freely accessible³². The website provides different interfaces for users and experts to interact with the database. Furthermore, users wishing to perform further analyses can download the entire database from the website.

To conclude this section, in Table 3.1 we provide a tabular summary of the different databases we discussed along with their characteristics. We also create an ORKG Comparison for these databases³³.

³⁰<https://cancer.sanger.ac.uk/cosmic>

³¹<https://mint.bio.uniroma2.it/>

³²<http://mips.gsf.de/proj/ppi/>

³³<https://www.orkg.org/orkg/comparison/R70880>

TABLE 3.1: Overview of several scholarly databases and their properties.

Database	DBMS	Domain	Content	Has web application	Query language	Format
CovidGraph	Neo4j	COVID-19	Both	Yes	Cypher, Free text	CSV, JSON
COVID-19 KG	OrientDB	COVID-19	Both	Yes	SQL	JSON
OpenBiodiv	GraphDB	Biodiversity	Both	Yes	SPARQL, Free text	JSON, CSV, TSV, RDF
CoDa	TriplyDB	Human cooperation	Both	Yes	SPARQL	CSV, RTF
MAG	Unknown	Generic	Metadata	Yes	Free text	CSV
OpenAIRE	Unknown	Generic	Metadata	Yes	Free text	CSV
COSMIC	Oracle	Cancer mutations	Both	Yes	Free text	CSV, TSV, Excel
MINT	PostgreSQL	PPI	Data	Yes	Free text	No
MIPS	MySQL	PPI	Data	Yes	Free text	No

3.2 Information extraction

Some literature focuses on extracting structured data from articles using text mining and natural language processing approaches. This work can be seen as a precursor to scholarly data science, where information extraction populates a database from unstructured or semi-structured text [63]. After this, data science is applied to gain insights from data.

Several works introduced systems to abstract metadata. Tkaczyk et al. [62] presented an open-source system to extract structured metadata from articles. The system is available online³⁴. Users can upload a PDF file of the article and the system will automatically extract the metadata such as article title, authors, journal, and bibliographic information. However, it is challenging to use this system for scholarly data science purposes due to multiple reasons. First, users can only upload one article at a time, which means that extracting metadata from numerous articles to build a data set could take a considerable amount of time. Second, users cannot export the extracted metadata into data sets or files automatically.

Other disciplinary systems have been proposed that extract knowledge from literature, e.g., [60] and [63]. Swain and Cole [60] designed the ChemDataExtractor, a system for extracting structured chemical information from text and tables of research documents. The system is accessible online³⁵. Users can upload their files in PDF, XML and HTML formats. ChemDataExtractor applies natural language processing methods to deliver results. The structured machine-readable output can be saved to a database or exported in several formats such as SDF, CSV, and JSON files for further analysis.

Related to the biomedical literature, Torii et al. [63] created the RLIMS-P, a system for extracting "protein phosphorylation information on protein kinase, substrate, and phosphorylation sites" from the literature. RLIMS-P mines PubMed abstracts and open access full-text articles. It is available online³⁶. Users can enter the PMIDs of the articles, i.e., the identifier of the article on PubMed, which they want to extract information from. The returned results can be exported in CSV files.

3.3 Data analysis on article metadata

A considerable amount of work has focused on analyzing the metadata of conference series.

³⁴<http://cermine.ceon.pl>

³⁵<http://chemdataextractor.org/>

³⁶<https://research.bioinformatics.udel.edu/rlimsp/>

OpenResearch [65] is a crowdsourcing platform that supports this kind of analysis. It enables collecting, organizing, sharing, and publishing information about scientific events (such as conferences) and event series in a semantically structured way. OpenResearch is accessible online³⁷ and contains metadata about 9000 events and 1000 event series.

Examples of exploiting OpenResearch in data analysis include [26] and [27]. Fathalla et al. [26] analyzed scholarly metadata of 40 computer science conference series by performing descriptive statistics and exploratory data analysis. The authors selected the relevant series first. They then collected metadata about these events from numerous online sources. After that, the collected metadata is preprocessed, ingested into OpenResearch, and analyzed. Insights about the continuity of each conference series over the years, the time and geographical distribution of conferences, and the popularity and productivity of the computer science sub-fields were studied and visualized in different forms such as tables, charts, and maps. The authors expanded this work to include Physics, Engineering, and Mathematics as additional research fields [27].

Other examples of analyzing the metadata of conference series include [13] and [11]. Biryukov and Dong [13] analyzed 14 sub-fields in the computer science literature. They used the computer science bibliographic database (DBLP)³⁸ to build a data set from articles of 2626 conferences. They then performed statistical analysis over the data set and compared the sub-fields in terms of collaboration patterns, population stability, and productivity.

To understand the growth of the human-computer interaction (HCI) research area in Brazil, Barbosa et al. [11] explored the metadata of 340 articles published in an HCI conference series in Brazil. The authors analyzed and visualized the data to investigate the evolution of the community, co-authorship networks, institutions, and research topics over time.

Apart from conference series analysis, another effort to analyze and visualize scholarly metadata is presented by Nielsen et al. [47]. The authors developed a free web service called Scholia³⁹. Scholia fetches scholarly information from Wikidata and generates scholarly profiles of the searched entity. Users can search for researchers, topics, organizations, awards, events, etc. When searching, the system queries against the SPARQL endpoint of Wikidata and creates visualizations on-the-fly. These visualizations constitute then the scholarly profiles on Scholia. The profiles contain lists of articles of researchers or organizations, distribution of articles over time, co-author graphs, maps, and more.

³⁷https://www.openresearch.org/wiki/Main_Page

³⁸<https://dblp.org/>

³⁹<https://scholia.toolforge.org/>

Researchers exploited SKGs in analyzing metadata in different research fields. For example, Effendy and Yap [22] analyzed the research trend in computer science using MAG. The authors investigated trends, evolution, and inter-relationships in different sub-fields.

While the work presented in this section focuses on analyzing metadata, our work demonstrates some approaches to analyze the actual content of articles.

3.4 Data analysis on article content

Agrawal et al. [1] proposed a domain-independent method for extracting specific concepts from articles. The approach uses natural language processing and text mining methods to extract the aim (target of the article), methods (approaches used to achieve the aim), and results (final output) from the title, abstract, and citation contexts of articles. The extracted concepts are then used to represent the research community as a knowledge graph stored in Neo4j database. To test the domain-independence of the proposed approach, the authors used a large data set consisting of 332793 articles and performed trend analysis in two distinct communities (computational linguistics and computer vision). They studied the research growth and decline across different topics in these fields.

The constructed knowledge graph is not public and thus users can not interact with it. However, by using the graph, the authors argue that they can summarize research fields in many ways. For instance, they can find all the methods used in the articles related to a specific field. Moreover, they can summarize the work of specific authors and analyze trends in research fields. Neo4j provides an interface to help the authors query the required data to perform such analyses.

While Agrawal et al. worked on summarizing research fields in terms of aims, methods, and results of articles, our work tends to gain new knowledge in research fields by exploiting the ORKG in data science. The ORKG is not restricted to these three concepts (aim, method, and result). We can describe the articles using several other properties in the ORKG.

Furthermore, several efforts exploited scholarly databases in their analysis such as [19, 58, 77].

Dimitrova et al. [19] demonstrated how to use OpenBiodiv (Section 3.1.1) to address specific tasks, for instance, finding the institutions that store material specimens of the genus *Prosopistoma* from numerous sources in the literature. Answering this type of questions is important for several stakeholders including curators, taxonomists, and institutions.

Shepherd et al. [58] showed how to successfully mine and integrate data sets from COSMIC database (Section 3.1.3). The article provides three examples of finding certain data sets of interest.

Ye et al. [77] presented a method for exposing activating cancer mutations. To test the proposed method, the authors used data from the COSMIC database to detect clusters of activating mutations in specific protein sequences.

Chapter 4

Scholarly Data Science Architecture and Implementation

This chapter introduces a generic architecture for data science on scholarly knowledge, possible implementation choices for the architecture, as well as the specific implementation proposed in this thesis, including the tools and methods used in our work.

4.1 Scholarly data science architecture

Gupta et al. [29] presented a model for knowledge discovery in databases (KDD). The model splits the KDD process (Section 2.2) into three layers. The bottom layer is called the storage schema layer and it is responsible for storing knowledge. The middle layer is the core layer, in which the essential mining process is conducted. The top layer is the front-end layer, i.e., the user interface of the system.

Rahman [54] proposed a systematic data mining architecture to mine intellectual knowledge from social data. The architecture is divided into several units:

1. Data collection and temporary storage unit: For collecting and storing data from Facebook.
2. Data processing unit: For preprocessing and normalizing the collected data.
3. Data parsing and classifying unit: This unit is the heart of the mining process. Different activities such as parsing data to select features and classifying text attributes using the K-nearest neighbor (k-NN) algorithm are performed in this unit.
4. Knowledge representation unit: For visualizing the extracted knowledge.

Moreover, YueShun and Wei [78] and Meo et al. [44] presented additional data mining systems with different architectures.

Inspired by [29, 54, 44, 78], we introduce a generic architecture for scholarly data science. Figure 4.1 depicts the different components of the proposed architecture.

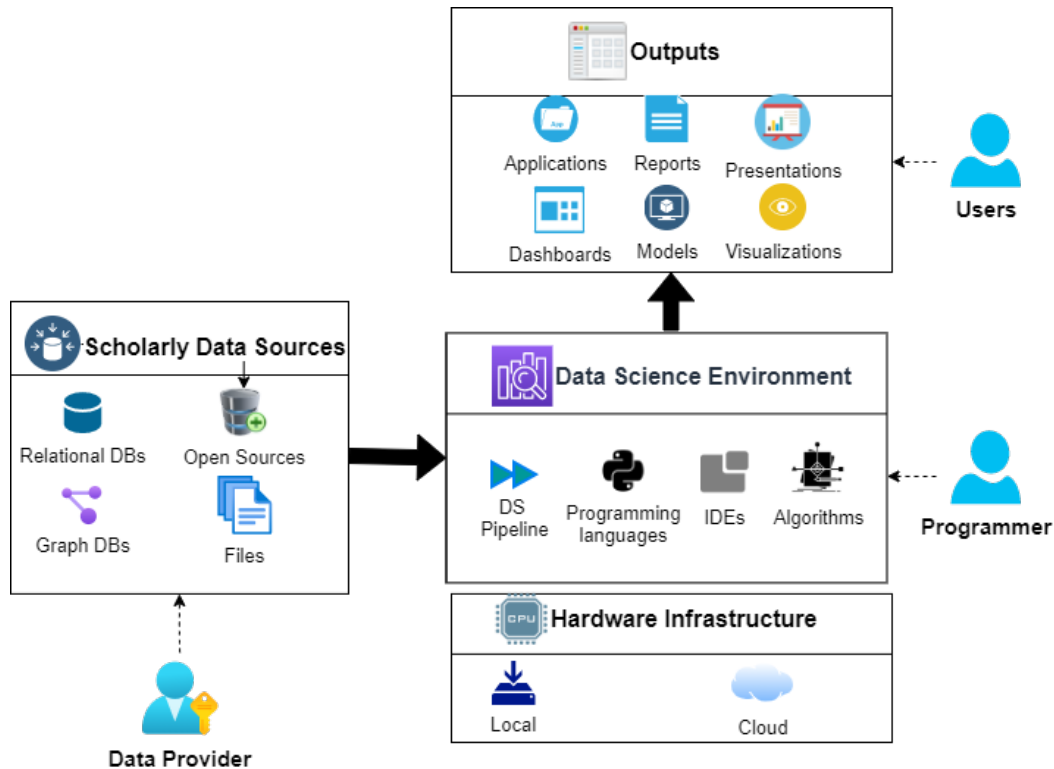


FIGURE 4.1: Scholarly data science architecture.

4.1.1 Scholarly data sources

We begin with scholarly data sources. As we saw in Section 2.2, specifying the relevant data sources is one of the first steps of any data science use case. After defining the problem, the scientist determines the required data sources and collects data.

Common types of scholarly data sources include:

- **Relational databases:** A very popular storage solution, where data can be stored in tables [17]. Common relational database management systems (RDBMSs) include SQL Server¹, MySQL², and Oracle³. Examples

¹<https://www.microsoft.com/de-de/sql-server/sql-server-2019>

²<https://www.mysql.com/de/>

³<https://www.oracle.com/de/>

of scholarly relational databases include MINT and MIPS (mentioned in Section 3.1.3).

- **Non-relational databases:** These databases cover different types such as key-value stores and graph databases [45]. Graph databases represent data as a network of nodes and edges [50] and are widely used for storing knowledge graph data [55]. Common graph database management systems include Neo4j and OrientDB. Examples of scholarly knowledge graphs include the Open Research Knowledge Graph (ORKG) and OpenAIRE research graph.
- **Flat files [7]:** Store data in plain text format with one record per line. Each record is separated by a delimiter (e.g., comma as in the CSV format or tab as in the TSV format). Data in flat files map to a single table, unlike relational databases that contain multiple tables.
- **Spreadsheet files:** A special type of flat files that also organize the data in tabular format, i.e., rows and columns. However, a spreadsheet can contain multiple sheets and each sheet can map to a different table, i.e., multiple data sets may be merged together into one spreadsheet [8]. Common spreadsheet formats are XLS and XLSX.
- **Open data sources:** Such as Application Program Interfaces (APIs) and web services. Data can be obtained from open sources in various formats such as CSV, JSON, and HTML.

Several databases implemented using relational and graph database management systems are discussed in 3.1.

4.1.2 Data science environment

The second component in the architecture is the data science environment, which is where the data science pipeline explained in Section 2.2 is executed. Depending on the problem at hand, such pipelines are often complex and involve numerous steps involving different kinds of processing ways, algorithms, and models. Moreover, a wide range of tools, frameworks, and programming languages may be used in the environment. Some of the most important data science programming languages and packages include:

- **Python⁴:** A general-purpose programming language, used widely for addressing data science issues [68]. Several libraries are built-in Python

⁴<https://www.python.org/>

to support data science and machine learning such as NumPy⁵, Pandas⁶, Bokeh⁷, Matplotlib⁸, Tensorflow⁹, Geoviews¹⁰, etc.

- R¹¹: A programming language for data science and statistics. It allows data to be prepared, processed, and modeled [43]. R contains several useful libraries such as Tidyverse¹² and Plotly¹³. The majority of data science community now use Python and R [53].
- Shiny¹⁴: An R package for creating interactive web applications. Shiny has pre-installed input and output controls that have an automatic “reactive” binding between them. This allows the application to be customized interactively and live by the user without any background knowledge of the programming language.
- JavaScript¹⁵: An object-oriented programming language. Although it is mostly used for web development, libraries such as D3¹⁶ and AmCharts¹⁷ are powerful for data visualization and dashboard creation.
- Scala¹⁸: A general-purpose programming language used for different purposes such as web development and machine learning. It is also efficient for handling big data.
- Structured Query Language (SQL): A popular programming language for storing, processing, and retrieving data from relational databases. Even though it is not central to data science operations, it is important in the stage of collecting data since data are often managed by a RDBMS.
- SPARQL¹⁹: While SQL is widely used to retrieve data from relational databases, SPARQL is the standard language to query graph data encoded following the RDF data model. In order to make a SPARQL query, a SPARQL endpoint is needed, i.e., software that can send a query formulated in SPARQL syntax using the SPARQL protocol to the

⁵<https://numpy.org/>

⁶<https://pandas.pydata.org/>

⁷<https://docs.bokeh.org/en/latest/>

⁸<https://matplotlib.org/>

⁹<https://www.tensorflow.org/>

¹⁰<https://geoviews.org/>

¹¹<https://www.r-project.org/>

¹²<https://www.tidyverse.org/>

¹³<https://plotly.com/r/>

¹⁴<https://shiny.rstudio.com/>

¹⁵<https://www.javascript.com/>

¹⁶<https://d3js.org/>

¹⁷<https://www.amcharts.com/>

¹⁸<https://www.scala-lang.org/>

¹⁹<https://www.w3.org/TR/sparql11-query/>

server that contains the queried data, and presents the results received back in a suitable form.

Software and script development in any programming language should, and perhaps typically does, occur using an integrated development environment (IDE). Common IDEs include:

- Jupyter Notebook²⁰: An open-source web application that follows an ordered list of input/output cells. These cells provide space for code, markdown text, mathematical formulas, and media content. Users can interact with their code and show the outputs of the code cells in the same document. The final work can be exported as a PDF, HTML, and Python file. Jupyter Notebook supports multiple languages like Python and R.
- PyCharm²¹: An IDE designed for programming in Python. Free open-source community and professional versions are available. With the community version, pure Python projects can be created, while the professional version offers further possibilities of web development and HTML, JS, and SQL support.
- RStudio²²: A very popular IDE for R programming language.
- Visual Studio²³: An IDE from Microsoft for JavaScript development. Visual studio code is a very popular JavaScript text editor.

4.1.3 Hardware infrastructure

The data science environment needs to be implemented and executed on hardware. Possible implementations for the hardware infrastructure include:

- Local installation: Considering the available computing power, the necessary tools are installed on a local machine. When working with big data sets, the technical characteristics of the machine play an important role. Furthermore, to use data science programming languages such as Python and R, programmers can choose between installing Jupyter Notebook or the corresponding IDEs.
- Cloud implementation: Sometimes, it is not desirable (or possible) to perform data science tasks in a local environment. Handling large amounts of data poses multiple challenges for data access and processing. To overcome these challenges, cloud-based computing technologies are infrastructure for handling the intensive use of computing power and

²⁰<https://jupyter.org/>

²¹<https://www.jetbrains.com/de-de/pycharm/>

²²<https://rstudio.com/>

²³<https://visualstudio.microsoft.com/>

storage [38]. Available cloud services include Amazon Web Services (AWS)²⁴, Google Cloud Platform²⁵, and Microsoft Azure²⁶. Moreover, if a collection of Jupyter notebooks is located in a Git repository, Binder²⁷ is another solution to serve easily executable notebooks, making the code reproducible by users everywhere without requirements on the local machine except for internet connectivity and a browser.

4.1.4 Outputs

As for the results, some of the common forms of data science experiment outputs include [12]:

- Reports and presentations: Telling the story of the data along the data science pipeline to delivering results. Ideally, reports are clearly written containing concise conclusions and omitting unnecessary data. Furthermore, reports should be reproducible.
- Applications, web pages, dashboards, and visualizations: Intuitive, clear, and easy to use.
- Statistical and machine learning models.

Users (mentioned in next section) employing data science outputs for their purposes may face some challenges such as [35]:

- Poor documentation of the sequence of steps made during the analysis.
- Static and inflexible reports that do not allow for interactive inspecting of contents.

4.1.5 Actors

The last part of our architecture represents the actors interacting with the different components, including:

- Data providers: Actors with credentials who ingest their data into data sources. Data providers should be authorized to avoid any inconvenience related to the quality of the added data.
- Programmers: Actors with programming skills who import the data into the data science environment, explore, model, and analyze the data. On one hand, this kind of actor consumes the data collected from

²⁴<https://aws.amazon.com/de/>

²⁵<https://cloud.google.com/>

²⁶<https://azure.microsoft.com/>

²⁷<https://mybinder.org/>

the source, on the other hand, they generate new content and services for another kind of actor.

- Users: Actors who use the services such as applications and dashboards resulting from the data science environment for various purposes, e.g., management and decision making.

4.2 Architecture implementation

As discussed in the previous section, several diverse choices are available for implementing each architecture component. We now present our choices for the architecture implementation, including the tools and methods.

4.2.1 Scholarly data sources

As for the scholarly data sources, our main source is the Open Research Knowledge Graph (ORKG). We start by presenting some of the relevant main features of the ORKG.

The ORKG is an infrastructure that enables the acquisition, curation, publishing, and processing of scholarly knowledge published in the literature [33]. It is available as an online service that stores research contributions (e.g., the contributions published in research articles) in a graph-based database, enabling easier reuse of scholarly knowledge. One kind of reuse is in downstream data science.

Since ORKG represents research contributions in a structured and machine-readable form, it becomes possible to automatically generate comparisons of contributions addressing the same research problem. Oelen et al. [48] presented a workflow to compare research contributions in a scholarly knowledge graph and implemented the workflow in ORKG. Each article in the ORKG consists of at least one research contribution and is represented with data including for instance methods, materials, results, etc. The output of the flow is a comparison table containing the compared contributions as columns (or rows), the commonly shared properties as rows (or columns), and the values of these properties in the table cells. Users can modify the comparison table according to their needs, share it using a link, and export it in numerous formats such as PDF, CSV, and \LaTeX . As these comparisons tabulate relevant data extracted from the literature, they can be used for scholarly data science.

The ORKG Python library²⁸ is another key ORKG feature that supports using comparisons for data science purposes. The library streamlines reading ORKG content, and contribution data in particular, directly into native

²⁸<https://pypi.org/project/orkg/>

Python data structures (specifically, Pandas data frames). This means that ORKG content can be easily accessed and reused in a data science environment, e.g., Jupyter Notebook.

Wikidata²⁹ is another knowledge graph used as a data source in this work. Wikidata is a knowledge graph database operated by the Wikimedia foundation since 2012. Wikidata has a SPARQL endpoint, which allows the data to be queried via SPARQL queries. In particular, we query the population data of Chinese provinces from Wikidata to help answering an analytical question in Chapter 5.

Finally, as a third data source we also integrate data from flat files such as CSV files, from GeoJSON files and Shapefiles. These files are obtained from open sources.

4.2.2 Data science environment

After importing scholarly data from the sources, we perform data science tasks in our environment using several tools such as programming languages, libraries, and IDEs. We use Jupyter Notebook to create and execute Python notebooks. In the notebooks, we use numerous Python libraries, including Pandas, Numpy, Bokeh, Geoviews, and Matplotlib to process the data. We use also the ORKG Python library to access ORKG content in Jupyter Notebook. In addition to Python and Jupyter, we also use R with its IDE RStudio and several libraries such as Shiny, Plotly, Tidyverse and its ggplot2 to create an interactive shiny web application. Moreover, we use visual studio code with the JavaScript library amCharts to build a web page containing an interactive network visualization. SPARQL is also used to retrieve data from Wikidata.

As for the methods, common methods are used in both use cases—described in following chapters—to implement the pipeline explained in Section 2.2, in particular:

- Importing and preprocessing data: After importing scholarly data into our environment, we preprocess it. The aim is to obtain data frames ready for analysis. Furthermore, we apply regular expression operations (RegExr) for filtering text strings. In RegExr a text pattern is described by a special syntax and then applied to the data frame.
- Visualizing data: We use multiple visualization libraries in Python, R, and JavaScript to produce interactive visualizations and applications on top of scholarly data.

²⁹https://www.wikidata.org/wiki/Wikidata:Main_Page

In addition to these methods common to both use cases, we use additional methods that are specific to one use case. These methods are detailed in the respective chapters.

4.2.3 Hardware infrastructure

The software and service environment is implemented locally. We have installed the IDEs and the required libraries on a local machine and performed all the tasks using it. We used a dell latitude 5480 laptop that has the following hardware characteristics:

- Processor: Intel(R) Core(TM) i3-7100U CPU @ 2.40 GHz
- Random Access Memory (RAM) : 8 GB
- Used disk space: 12 GB (without the space used for the operation system).

4.2.4 Outputs

Outputs from our data science environment include:

- Different types of interactive visualizations, including charts, maps and graphs
- Interactive web application
- Interactive web page
- Notebooks and reports

We consider the challenges [35] mentioned in Section 4.1.4. We provide reproducible and well-documented analyses using Jupyter Notebook. For each analysis, we embed the script, its documentation, and the interactive visualizations and results in one notebook.

All scripts, applications, visualizations, and other results are available on GitLab repositories^{30 31}.

4.2.5 Actors

Overall, we have three kinds of actors interacting with different components in the architecture.

³⁰https://gitlab.com/TIBHannover/orkg/orkg-notebooks/-/tree/master/covid_19

³¹https://gitlab.com/TIBHannover/orkg/orkg-notebooks/-/tree/master/invasion_biology

- Data providers: Actors and researchers can ingest structured descriptions of their articles and research contributions in the ORKG database.
- Programmers: Actors who import the data into data science environments to analyze, model, and visualize data. As notebooks are available online and are served in executable form, users with little or no programming skills can interact with the scripts, run them, and inspect the outputs.
- Users: Actors who use services, specifically applications and dashboards, for their own purposes, e.g., exploring, analyzing, management, and decision making.

In the following chapters, we test the presented architecture implementation in two use cases in different domains and report our results.

Chapter 5

COVID-19 Use Case

In this chapter, we apply the presented scholarly data science architecture implementation to COVID-19 related scholarly knowledge obtained from the ORKG and other sources. Specifically, we describe a data science pipeline to visualize model estimates against the observations and quantify the overall model performance. Next, we present two pipelines to perform a spatial and pattern analysis of both model performance and COVID-19 prevalence across China. Finally, we describe another pipeline for updating a study result by combining multiple ORKG Comparisons. We provide the links to the notebooks in the corresponding sections to support their online execution and thus reproducibility and active exploration of the results and visualizations (e.g., hovering over the visualizations in the notebooks shows more details than the static visualizations presented in this document).

5.1 Methods

In addition to the common methods listed in Section 4.2.2, specific methods used in this use case include the following.

- **Data combination:** After retrieving ORKG (Comparison) data into a data science pipeline, it is integrated with data from other sources to answer specific analytical questions.
- **(Geo-)Spatial analysis and pattern discovery:** A method for studying the geographical distribution of data to state patterns, trends, differences, and associations within regional distributions [37]. Moreover, it can be used to analyze and interpret model results.
- **Clustering analysis:** A method for partitioning a set of data points according to some measure of similarity (e.g., distance) into subsets [3]. Each subset is a cluster, such that data points that are in the same cluster are similar to each other and different from data points in other clusters. The goal of clustering is to reveal subgroups within heterogeneous

data. It is an unsupervised learning procedure, i.e., predefined groups of data points are not required.

5.2 SEIR model estimates

Aleta et al. [4] investigated the effect of travel restrictions on the spread of COVID-19 in China. To simulate the spread of the epidemic, a stochastic SEIR-metapopulation model was implemented. The model estimates the number of infected cases depending on the mobility data in 2019 and 2020. This allows comparing two radically different scenarios, one with no travel restrictions (2019) and another in which mobility is reduced by a travel ban (2020).

The information relevant to us is the numbers that the model estimated and the number of observed cases in each Chinese province. This data is extracted and organized in an ORKG Comparison¹. Building on this, we performed different data science tasks implemented as two Jupyter notebooks. The notebooks are described in the next three sub-sections.

5.2.1 Estimated vs. observed cases and model performance

This sub-section describes our work in an online accessible Jupyter notebook².

We first visualize the data, i.e., visualize the number of estimated against the observed cases in each Chinese province using a bar plot. Then, we evaluate the performance of the model by determining the overall prediction error. Figure 5.1 shows the workflow used to implement these two tasks.

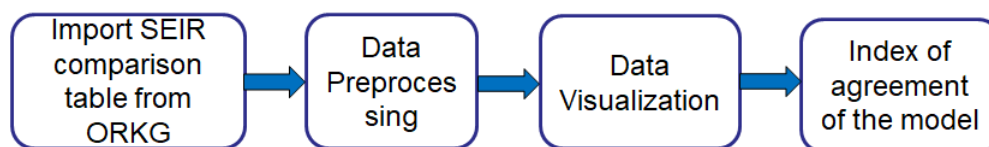


FIGURE 5.1: Workflow used for visualization and prediction error calculation.

Using the ORKG Python library, we import the comparison of SEIR model estimates into a Jupyter notebook. Then, we preprocess the data frame, i.e.,

¹<https://www.orkg.org/orkg/comparison/R39082>

²https://mybinder.org/v2/g1/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=covid_19%2FSEIR/visualization

we clean the data and apply regular expression operations to prepare the data for visualization. Figure 5.2 shows the final plot.

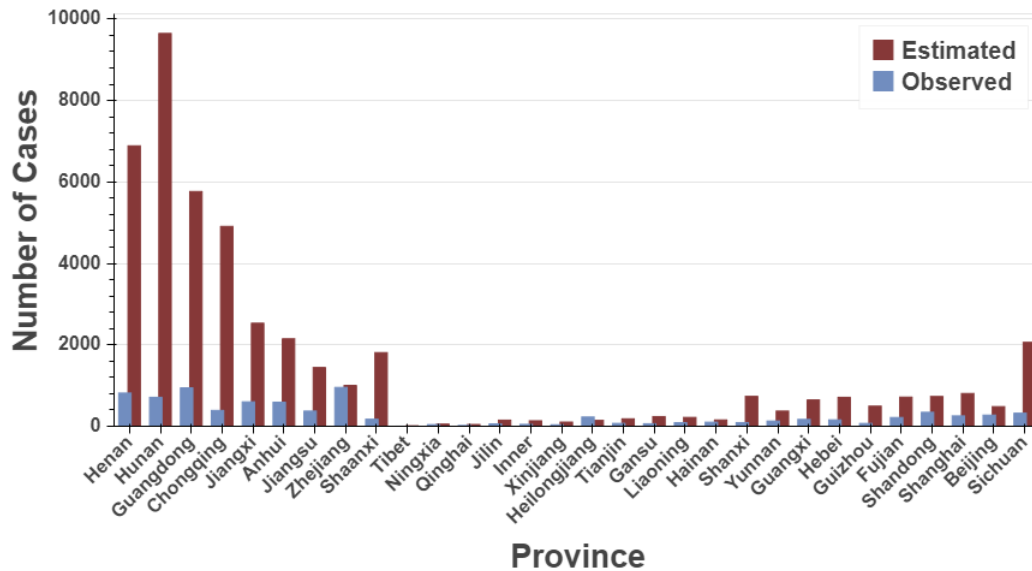


FIGURE 5.2: Number of estimated and observed cases in Chinese provinces.

The figure clearly shows that the model considerably overestimates the number of infected cases in almost all Chinese provinces. We notice that the estimation of the model was close to the observations in only a few provinces (e.g., Zhejiang and Hainan). In other provinces, there is a huge gap, indicating that the model has a poor performance in these provinces.

Next we quantify the model performance by calculating the index of agreement of the model. As a performance indicator, the index of agreement³ is a standardized measure of the degree of model prediction error that varies between 0 and 1. The agreement value of 1 indicates a perfect match between the observed and estimated values, while 0 indicates no agreement at all.

We write a function to calculate the index of agreement. The function takes two parameters: an array of observed values and an array of estimated values. As a result, the function returns the index. The index of agreement of the SEIR model is $d = 0.194$, i.e., poor overall performance.

The next two sub-sections describe our work in another online accessible Jupyter notebook⁴.

³<https://agrimetsoft.com/calculators/Index%20of%20Agreement>

⁴https://mybinder.org/v2/g1/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=covid_19%2FSEIR/spatial_analysis

5.2.2 Spatial and pattern analysis of model performance

Having evaluated the overall model performance, we now evaluate the model performance in light of spatial patterns and differences by plotting a map of China and investigating each province separately. To achieve this, data from the ORKG is not enough since we lack geospatial data. Thus, we integrate data from two additional sources into our data frame. The steps followed in this section are presented in Figure 5.3.

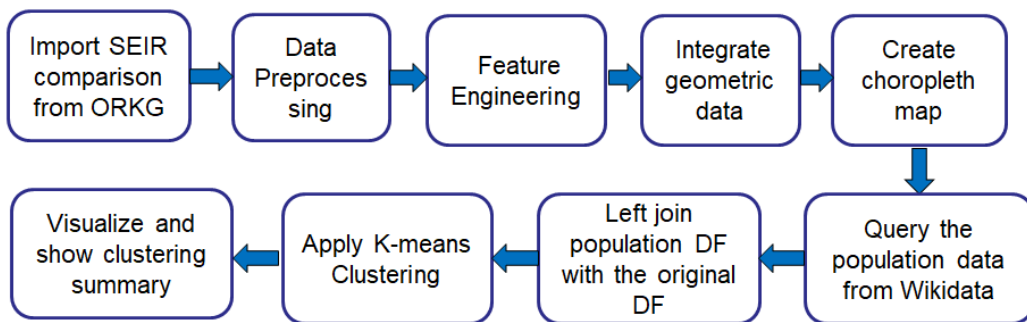


FIGURE 5.3: Workflow of spatial analysis of model performance.

The first two steps in this workflow are similar to the previous one. After importing and preprocessing data, feature engineering is applied to determine the model prediction error in each province. Previously, we used the index of agreement to measure the overall prediction error. However, this method requires the average observation and prediction values. Thus, it can not be used to compute the prediction error for individual records, i.e., provinces. To achieve this, we use the Mean Absolute Error (MAE). Therefore, a new feature for MAE is created in our data frame based on the ‘estimated’ and ‘observed’ features.

However, we noticed that the number of cases is skewed, i.e., the numbers in Hubei are much higher than the numbers in other provinces, which results in a skewed MAE. To adjust for the skewness, we use logarithmic transformation. Figure 5.4 shows a partial snapshot of the data frame, in which the provinces are sorted by the model prediction error in descending order.

We still miss geographic information of the administrative provinces. We integrate this data from a GeoJSON file into our original data frame using left join.

Finally, regarding the map type, we choose choropleth maps. A choropleth map is used to spatially visualise data by coloring or shading geographic regions depending on the values of a particular variable⁵ [21]. With this type

⁵<https://www.axismaps.com/guide/choropleth>

	location	MAE	MAE_log
1	Hubei	10608.871	9.269446
2	Hunan	288.258	5.663856
0	Henan	195.871	5.277456
3	Guangdong	155.516	5.046749
4	Chongqing	145.806	4.982277
5	Jiangxi	62.290	4.131801
26	Sichuan	56.129	4.027653
9	Shaanxi	52.774	3.966019
6	Anhui	50.226	3.916533
7	Jiangsu	34.516	3.541423
17	Shanxi	20.774	3.033702
20	Hebei	17.871	2.883179
24	Shanghai	17.774	2.877737
22	Fujian	16.161	2.782601
19	Guangxi	15.452	2.737738
21	Guizhou	13.710	2.618125
23	Shandong	12.516	2.527008
18	Yunnan	7.903	2.067242
25	Beijing	6.710	1.903599
14	Gansu	5.677	1.736423
15	Liaoning	4.129	1.418035
13	Tianjin	3.645	1.293356
11	Jilin	2.935	1.076707
12	Heilongjiang	2.677	0.984697
16	Hainan	1.774	0.573237
8	Zhejiang	1.581	0.458058
10	Qinghai	0.806	-0.215672

FIGURE 5.4: A data frame snapshot with model prediction error in each Chinese province.

of map, we can inspect how the model prediction error changes across the provinces as shown in Figure 5.5.

A color bar varying from dark blue to yellow is used to represent the magnitude of prediction error. Dark blue indicates a very low error, i.e., the model estimations are accurate, and yellow references a high error. Based on this result, the model performs best in Qinghai and worst in Hubei.

Next, we investigate the relationship between the model prediction error and province population size. We hypothesize that the model performs worse (i.e., has a higher prediction error) in highly populated provinces.

To test the hypothesis, we need the population data of each province. Therefore, as described in Figure 5.3, we first query the population data from Wiki-data and merge the results with the original data frame using left join. With

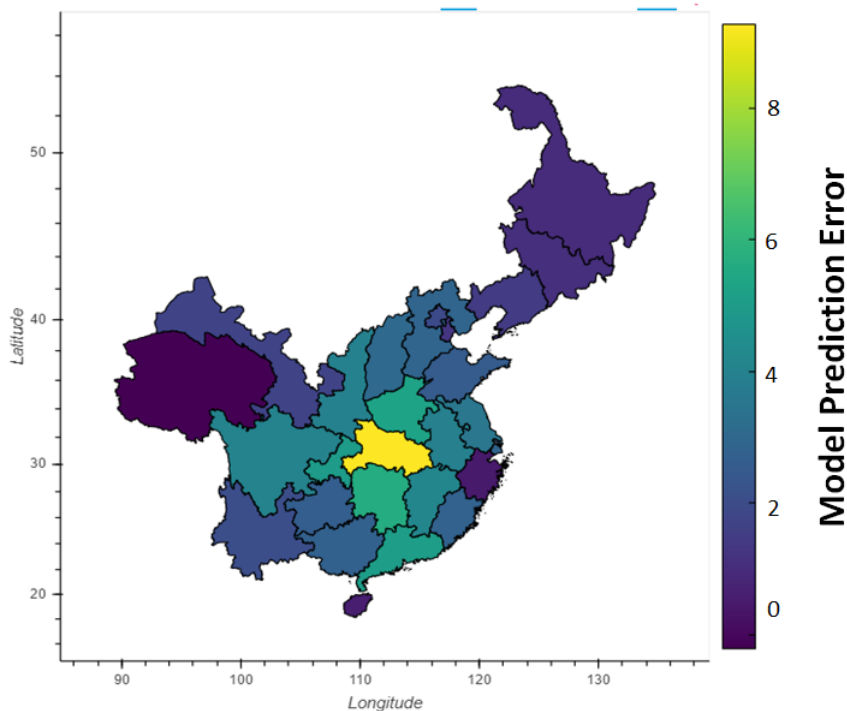


FIGURE 5.5: Spatial overview of model prediction error.

the population data, we cluster the provinces based on their population into two groups “high population” and “low population” using the K-means algorithm [41, 42]. Figure 5.6 shows the provinces clustered into two groups. There are 17 low-populated and 10 high-populated provinces.

Figure 5.7 shows the clustering summary of the algorithm. As we can see, the model has a much lower prediction error in the low populated provinces (22.411), compared to the error in high populated provinces (1142.135). This result confirms our hypothesis that the model performs worse in the higher populated provinces.

5.2.3 Spatial and pattern analysis of prevalence

So far we have gained a spatial overview of the model performance. The next task is to obtain a spatial overview of the spread of COVID-19. In other words, we analyze the distribution of the confirmed cases in the provinces spatially. Therefore, we create a choropleth map showing the prevalence across China.

As shown in Figure 5.8, until 05.02.2020 (date of the SEIR model estimates article [4]), the number of confirmed cases in Hubei was highest. Note that we divided the number in Hubei by 20 in order to ensure informative color grades on the map. At the time the article was written, there were 19665

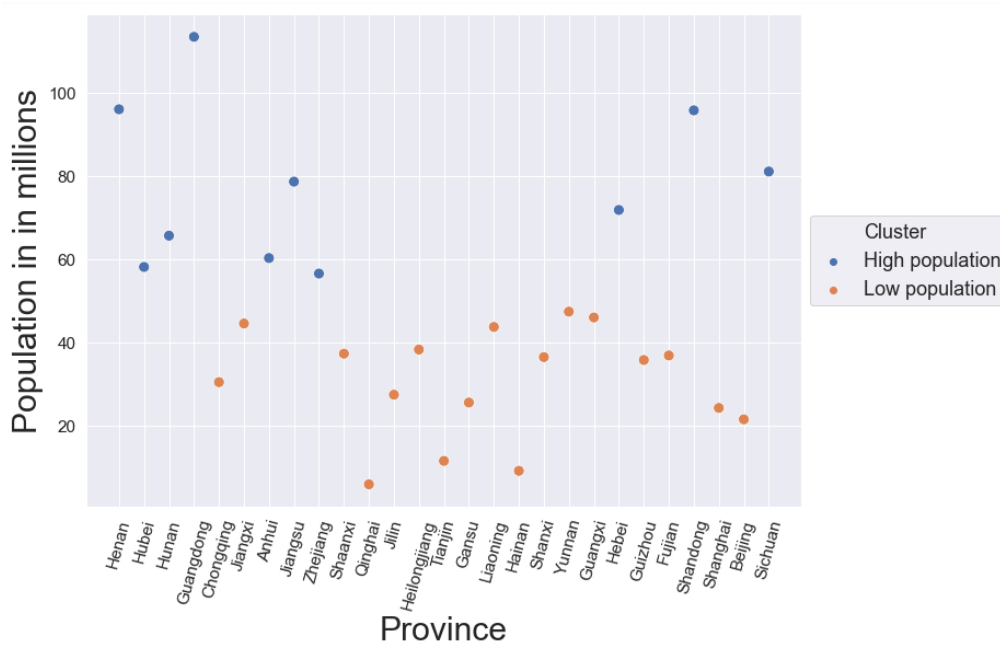


FIGURE 5.6: Clustering of provinces according to population.

Cluster	index	estimated	observed	MAE	MAE_log	pop	Clusters
High population	10.800000	37893.600000	2487.400000	1142.135500	4.261136	77.763089	1.0
Low population	18.058824	860.588235	175.588235	22.411588	2.233390	30.740653	0.0

FIGURE 5.7: Clustering algorithm summary.

confirmed cases in Hubei and 8317 in the rest of China. This means that in Hubei alone there were 70% of the total confirmed cases in China. Moreover, we note that the provinces neighboring Hubei in the north, south, and east also have a high number of cases. By further inspecting the map, we could divide the provinces into four categories depending on the number of cases:

1. Very high prevalence: Provinces with more than 1000 cases.
2. High prevalence: Provinces with cases between 500 and 1000.
3. Moderate prevalence: Provinces with cases between 150 and 499.
4. Low prevalence: Provinces with less than 150 cases.

Figure 5.9 shows this regional distribution pattern.

Here, we clustered the provinces by studying the spatial distribution pattern of the prevalence. However, we have also applied K-means clustering algorithm on the data to group the provinces based on the number of confirmed cases. This led to similar clustering results.

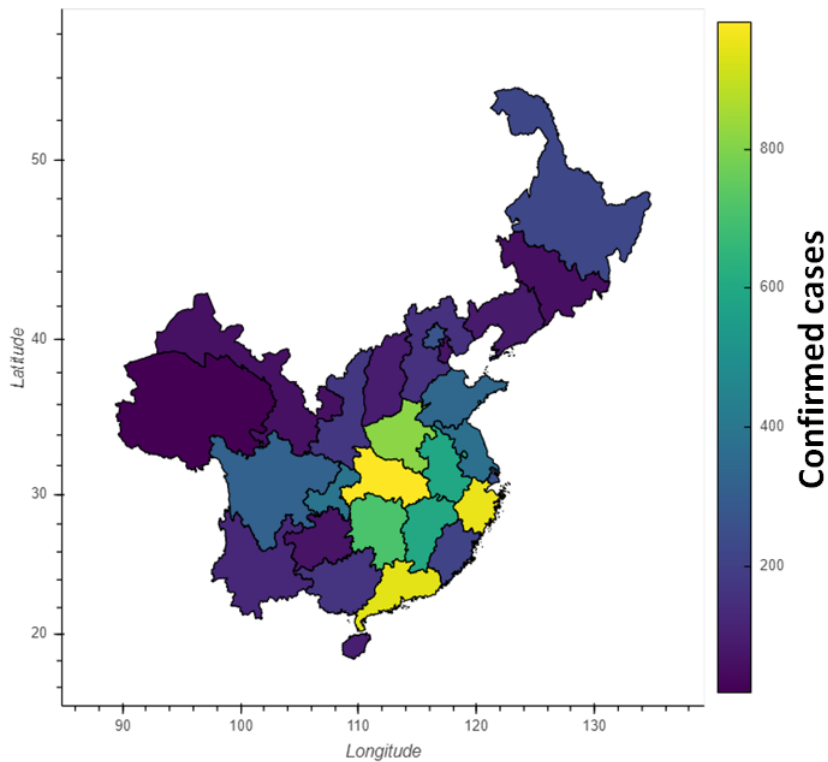


FIGURE 5.8: Spatial overview of prevalence.

Finally, we return to model performance. We investigate the model prediction error in each of the four prevalence clusters by calculating the index of agreement. As mentioned before, computing the index of agreement for only one observation is not possible. Since the 'Very high prevalence' cluster has only one province (Hubei), we drop it from further analysis. Of course, we still can consider the prediction error in Hubei using MAE as shown in Figure 5.4. However, the index of agreement and the MAE are not directly comparable.

The evaluation results show that the model has a prediction error of:

- 0.03 for category 2, which corresponds to provinces with high prevalence (500-1000).
- 0.11 for category 1, which corresponds to provinces with moderate prevalence (150-499).
- 0.22 for the category 0, which corresponds to provinces with low prevalence (< 150).

This result suggests that the performance of the model varies with the change of prevalence. The model works best in the provinces where the prevalence is low. The performance gets worse as the prevalence increases.

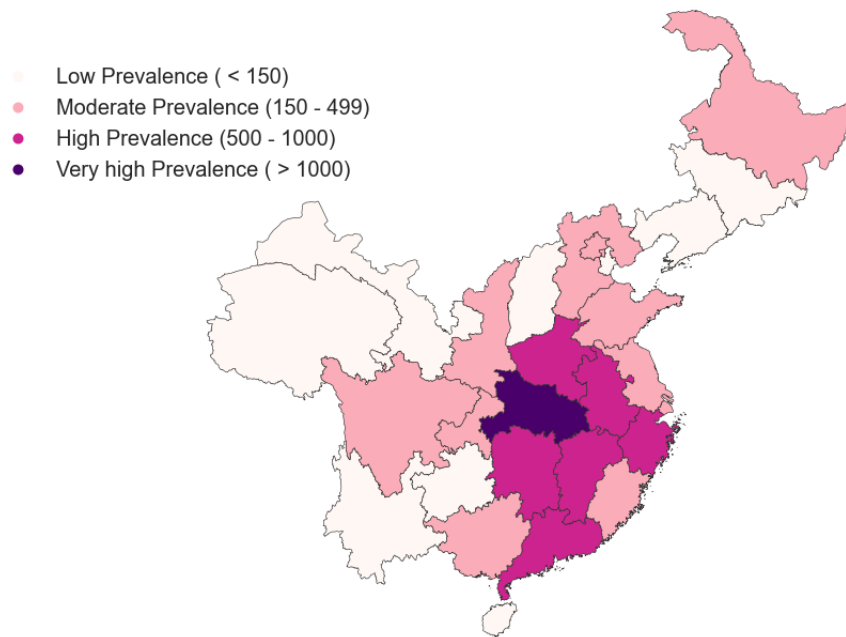


FIGURE 5.9: Regional distribution pattern of prevalence.

5.3 Multiple ORKG Comparisons as data source

This section describes our work in two online accessible Jupyter notebooks⁶.

We integrate data from multiple ORKG Comparisons and from another source (Google Docs), to update the results of a previous study.

In 2014, a study that compares Ebola to other infectious diseases was published in a Guardian article⁷. In this study, the basic reproduction number (R0) and case fatality rate (CFR) of numerous infectious diseases were gathered from the literature and organized in a data set. Then, a comparison of the diseases in terms of contagiousness and deadliness was visualized. The gathered data is available on Google Docs⁸. As the article was published in 2014 it does not include COVID-19.

This is an opportunity for us to demonstrate how to update the Guardian Microbe-Scope with COVID-19 by leveraging ORKG, the existing ORKG Comparisons for R0 and CFR of COVID-19 and the data science approaches we already introduced.

⁶https://mybinder.org/v2/g1/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=covid_19/R0_CFR

⁷<https://www.theguardian.com/news/datablog/ng-interactive/2014/oct/15/visualised-how-ebola-compares-to-other-infectious-diseases>

⁸<https://docs.google.com/spreadsheets/d/1kHCEWY-d9HX1WrfT9jjRQ2xf6WHQlmwyrXel6wjxkW8/edit#gid=0>

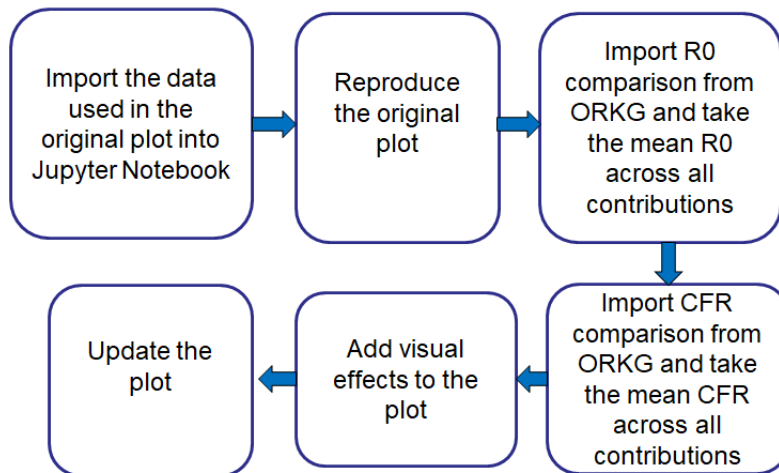


FIGURE 5.10: Workflow of updating the Guardian study.

Data for R0 and CFR of COVID-19 were already extracted from the literature and organized in two ORKG Comparisons⁹¹⁰. We use this scholarly data along with the Guardian data to create a new visualization that includes the original infectious diseases and COVID-19. Figure 5.10 shows the workflow of the process. First, we import the third-party data, preprocess it, and reproduce the original plot. Then, we import R0 and CFR comparisons and compute the mean value for each. After that, we plot the new visualization, shown in Figure 5.11.

Building on the original Guardian article, we adopt the following five classes of contagiousness:

- Not very: If R0 is less than 1.
- Quite contagious: If R0 is between 1 and 5.
- Very contagious: If R0 is between 5 and 7.
- Highly contagious: If R0 is between 7 and 12.
- Vaccinate now!: If R0 is higher than 12.

As for the deadliness, we adopt the following four classes:

- Not too deadly: If CFR is less than 1%.
- Quite deadly: If CFR is between 1 and 20%.
- Deadly: If CFR is between 20 and 50%.
- Extremely deadly: If CFR is higher than 50%.

⁹<https://www.orkg.org/orkg/comparison/R44930>

¹⁰<https://www.orkg.org/orkg/comparison/R41466>

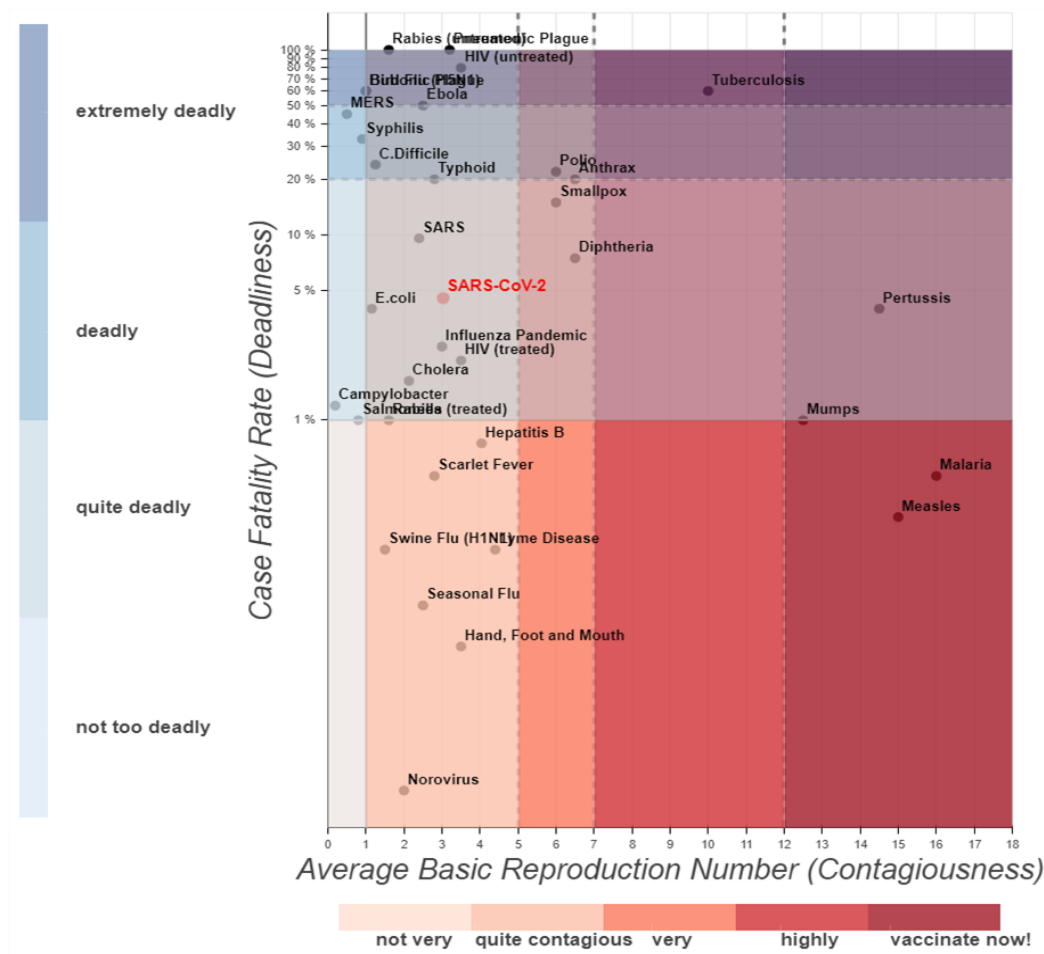


FIGURE 5.11: The contagiousness and deadliness of SARS-CoV-2 relative to other infectious diseases.

COVID-19 is represented with the red dot in the plot. According to the ORKG data, COVID-19 is classified as quite contagious and quite deadly relative to other infectious diseases.

Finally, we visualize the variance of COVID-19 contagiousness and deadliness relative to other infectious diseases by plotting COVID-19 in different locations. For this, we searched for common locations in the two comparisons, i.e., locations that have both R0 and CFR values in the corresponding ORKG Comparisons. This led to four common locations: Hubei, Wuhan, China, and Japan.

For the uncommon locations, we took the mean of each of R0 and CFR to represent COVID-19 in one extra dot called 'international'. As a result, we have five dots in our plot indicating how contagious and deadly COVID-19 is in different locations.

This final plot for this use case is shown in Figure 5.12. The figure suggests

that COVID-19 is classified differently in two locations: In Japan, COVID-19 is quite contagious and not too deadly, while in Hubei, the virus is very contagious and quite deadly.

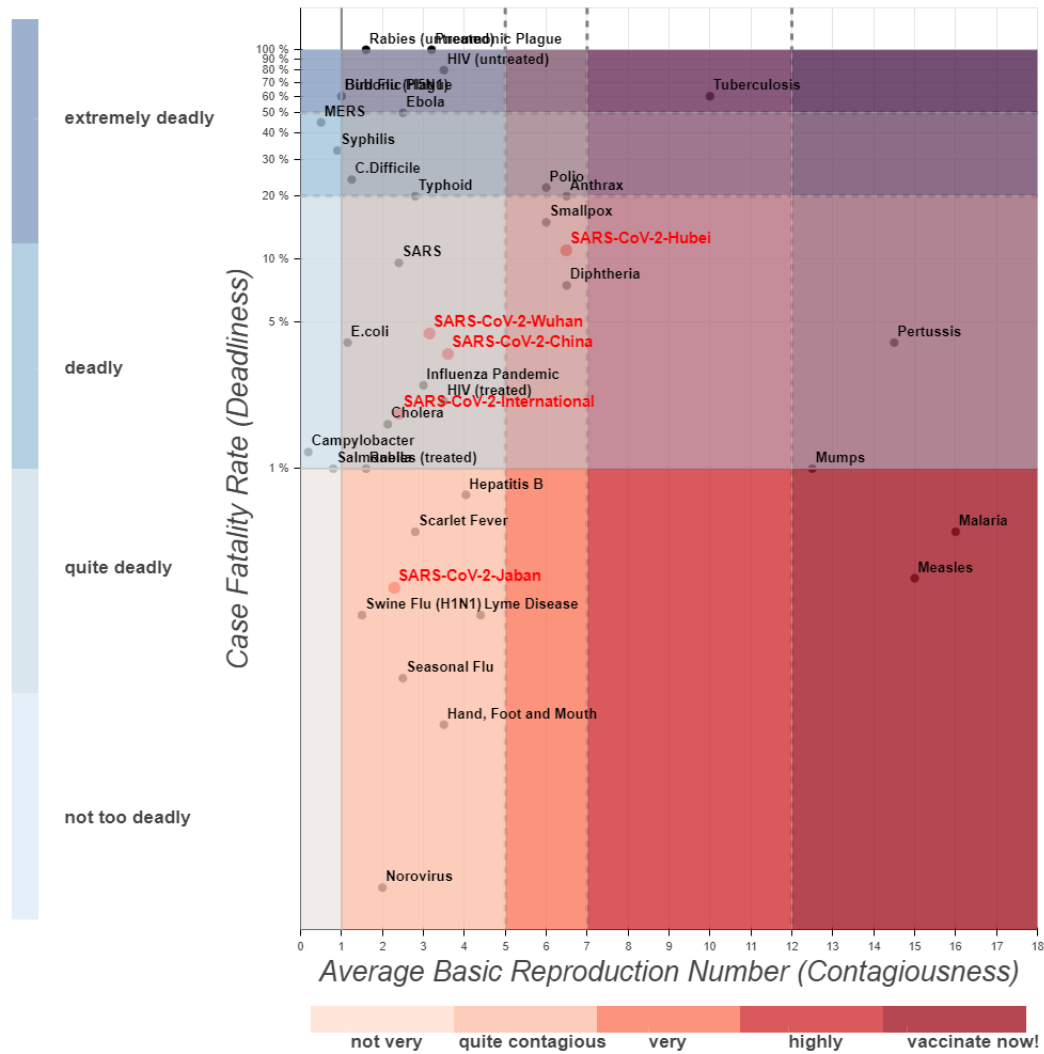


FIGURE 5.12: The contagiousness and deadliness of SARS-CoV-2 in different locations, relative to other infectious diseases.

Chapter 6

Invasion Biology Use Case

Our work in this use case is inspired by hi-knowledge¹ [24, 34]. The website provides an interactive network visualization of different hypotheses in invasion biology. The scholarly data of the articles related to the hypotheses is available on the website in multiple Excel files. In this use case, we first ingest the Excel data into the ORKG, create ORKG Comparisons, and then apply the presented scholarly data science architecture implementation to develop an interactive web application, interactive web page, and perform other analyses.

6.1 Methods

In addition to the common methods listed in Section 4.2.2, specific methods used in this use case include the following.

- Data ingestion into the ORKG
- Creating and publishing ORKG Comparisons
- Building an interactive web application and a web page on top of scholarly data

6.2 Importing articles and creating comparisons

In this section, we provide an overview of data ingestion and comparison creation in the ORKG.

Building on hi-knowledge, we work on ten hypotheses in invasion biology. Each of these hypotheses has been discussed in tens or hundreds of scientific

¹<https://hi-knowledge.org/invasion-biology-large/>

articles. Articles have tested the hypotheses by exploring and investigating several plant and non-plant taxa².

The starting point for us is scholarly data that was extracted from articles and published on the hi-knowledge website in separate files, one file per hypothesis. This data includes both article metadata and the extracted essential data, specifically:

- Article authors
- Publication year
- Article's stand of the hypothesis: Indicating whether it supports, is undecided, or questions the hypothesis
- The investigated taxa in the article, e.g., plants, birds, reptiles, etc.
- Number of investigated taxa in the article
- The continent in which the study was conducted
- Used research method: Experimental, observational, or correlational
- Experiment type: Enclosure, lab, or field.

We start by organizing and preprocessing the files, i.e., bringing them into the right format for ingestion into ORKG. For ingestion, we first create a CSV file for each hypothesis. The CSV file should be formatted in a way that it adheres to the following rules:

- The first row is the header: We specify here the properties that each contribution will have in the ORKG (these are columns in the CSV file and properties in ORKG contributions).
- One article per row: Starting with the second row, each following row corresponds to one contribution (one article could have one or more contributions).
- Cell values: These correspond to the property values, i.e., research contribution data. By default, each cell value is considered a literal (i.e., text). It is also possible to use an existing ORKG resource.
- For import into the ORKG, the CSV file should at least have the property for article title.
- Author names should be separated by a semicolon (e.g., Author1; Author2).

²As defined in Wikipedia, a taxon (plural taxa) is a group of one or more populations of organisms seen by taxonomists to form a unit. Members of a taxon are usually inferred to be related, they have characters in common that differentiate from other taxa.

Having prepared the CSV files, we can ingest them into the ORKG. To facilitate data ingestion, the ORKG provides an import tool³. The tool allows for checking how the data will be ingested; before starting the actual ingest, it is thus possible to first preview the data to ensure it is structured as expected. Following ingestion, we create ORKG Comparisons, one for each hypothesis.

To clarify this process, we provide an example of importing the CSV file of the limiting similarity hypothesis and creating an ORKG Comparison with data related to this hypothesis.

1. First, we create the CSV file with header as follows:

```
paper:title,paper:authors,paper:publication_year,  
paper:research_field,hypothesis,stand of hypothesis,  
measure of species similarity,measure of invasion success,  
number of plant species,continent,research method,  
type of experiment.
```

2. Using the import tool, we upload the CSV file and verify the data before the actual importing.
3. Once verified, we import the CSV file into the ORKG.
4. Since all the contributions share the same properties, we can compare them [48]. We select all the contributions we have just imported.
5. A new comparison appears listing all the contributions of the articles along with the properties. An example of a comparison of five contributions can be found here⁴.
6. Then, we publish the comparison to obtain an identifier for it.

The list of comparisons for the ten hypotheses we created for this use case can be found on ORKG by searching for 'invasion' and filtering by type 'Comparison'⁵.

6.3 Conducting scholarly data analysis

Describing scholarly knowledge in the ORKG opens up the opportunity to reuse knowledge in downstream data science. The following three sections describe how we reuse the hi-knowledge data we just ingested into ORKG in three different tools that demonstrate the variety of possibilities at hand.

³<https://www.orkg.org/orkg/csv-import>

⁴<https://www.orkg.org/orkg/comparison?contributions=R52072,R52074,R52076,R52078,R52080>

⁵<https://www.orkg.org/orkg/search/invasion?types=Comparison>

6.3.1 Analysis with Python Jupyter Notebook

After publishing, an ORKG Comparison gets a unique resource number. For example, the recourse number of the limiting similarity comparison is R52143⁶. Using the Python package and the comparison resource number, we can import comparison data into Jupyter Notebook.

With Jupyter Notebook, we perform data analysis across the articles related to six hypotheses, one notebook per hypothesis. As in the first use case, the Jupyter notebooks are accessible on Binder⁷.

We describe in more detail the notebook developed to analyze the data of 119 contributions related to the propagule pressure hypothesis. We formulate seven analytical questions and answer them by processing data and producing figures and plots, which can be explored online⁸.

It is worth mentioning that the main hypothesis in this notebook (propagule pressure) is divided into five sub-hypotheses according to the measure of propagule pressure. The first sub-hypothesis considers the total number released as the measure of propagule pressure. Propagule size, propagule frequency, distance from the source, and other proxies are measures of propagule pressure in the other sub-hypotheses.

Question 1: How many research contributions have discussed the propagule pressure hypothesis? How many contributions support, are undecided, or question the hypothesis?

After importing and preprocessing the comparison data, we plot a donut chart that shows the number of contributions supporting, being undecided, and questioning the main hypothesis.

Figure 6.1 shows that this hypothesis is discussed in 119 contributions and it is widely supported in the literature. More than 75% of the contributions support the hypothesis (i.e., 90 out of 119), while 15% (18 contributions) question it, and 10% (11 contributions) are undecided.

Question 2: How many contributions have discussed each of the sub-hypothesis of propagule pressure? How many contributions support, are undecided, or question each sub-hypothesis?

⁶<https://www.orkg.org/orkg/comparison/R52143>

⁷https://mybinder.org/v2/gl/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=invasion_biology%2Fhypotheses_notebooks

⁸https://mybinder.org/v2/gl/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=invasion_biology%2Fhypotheses_notebooks/propagule_pressure

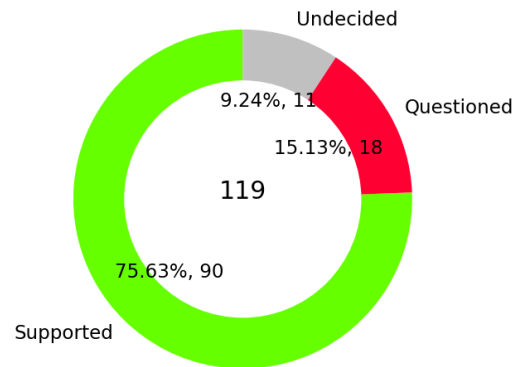


FIGURE 6.1: Propagule pressure hypothesis.

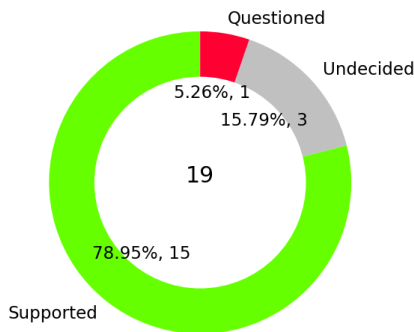


FIGURE 6.2: Total number released.

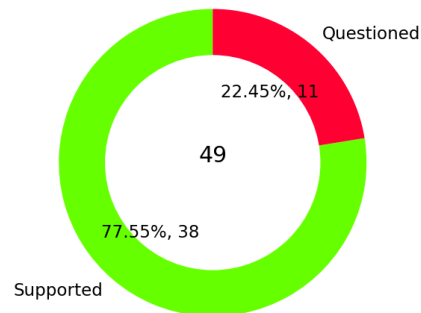


FIGURE 6.3: Propagule size.

The donut charts in Figures 6.2, 6.3, 6.4, 6.5, and 6.6 answer this question. As the figures suggest, most of the literature considers the propagule size as a measure of propagule pressure (49 out of 119 contributions), and as in the main hypothesis, the five sub-hypotheses are widely supported.

Using the Python Graphviz package, we can also demonstrate the hierarchy of the hypothesis. Figure 6.7 depicts a graph visualization of the hypothesis and its sub-hypotheses.

To test the hypothesis, some articles investigated plant taxa while others investigated non-plant taxa such as birds, reptiles, etc.

We first look at the articles that investigated plant taxa, specifically with the following question.

Question 3: How many plant taxa did each article investigate? Which articles investigated the highest numbers of plant taxa?

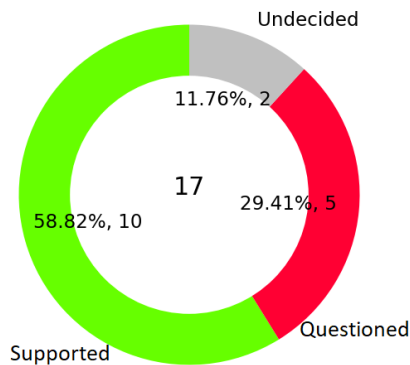


FIGURE 6.4: Propagule frequency.

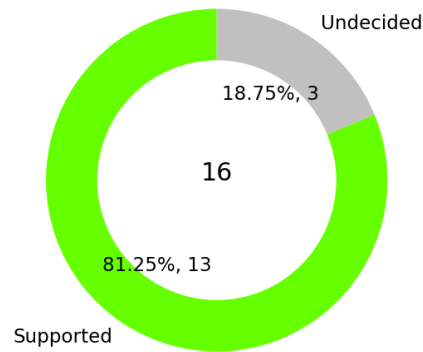


FIGURE 6.5: Distance from source.

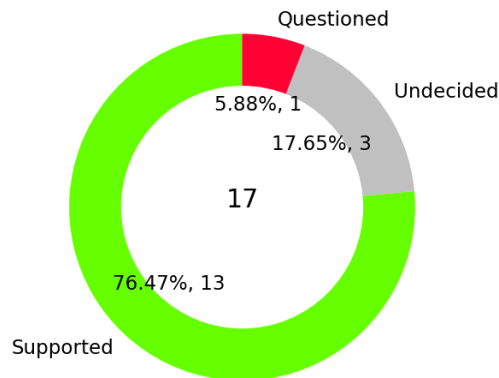


FIGURE 6.6: Other proxies.

To answer this question, in Figure 6.8 we plot a bar chart that demonstrates the number of investigated plant taxa in each article along with the article's stand for the hypothesis and the publication year. Each bar in the plot represents an article, while the bar height reflects the number of plant taxa investigated in the article. We disable the labels of the x-axis because the titles of the articles are too long to be represented as axis ticks labels. To see titles, interested readers can explore the notebook online, where plots can be hovered to reveal the relevant four elements of information for each article: article title, the exact number of investigated plant taxa, article publication year, and stand for the hypothesis.

Figure 6.9 depicts a plot that shows the top-10 articles with the highest numbers of investigated plant taxa.

Question 4: How many non-plant taxa did each article investigate? Which articles investigated the highest numbers of non-plant taxa?

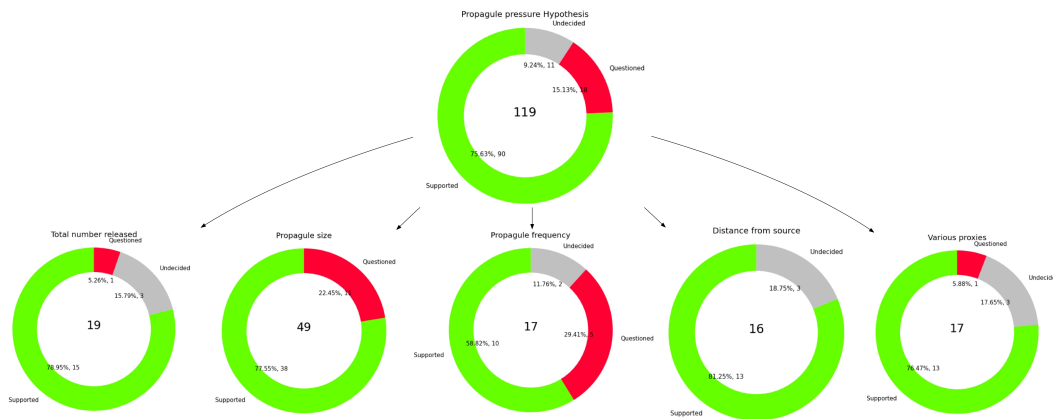


FIGURE 6.7: Hierarchy of the hypothesis.

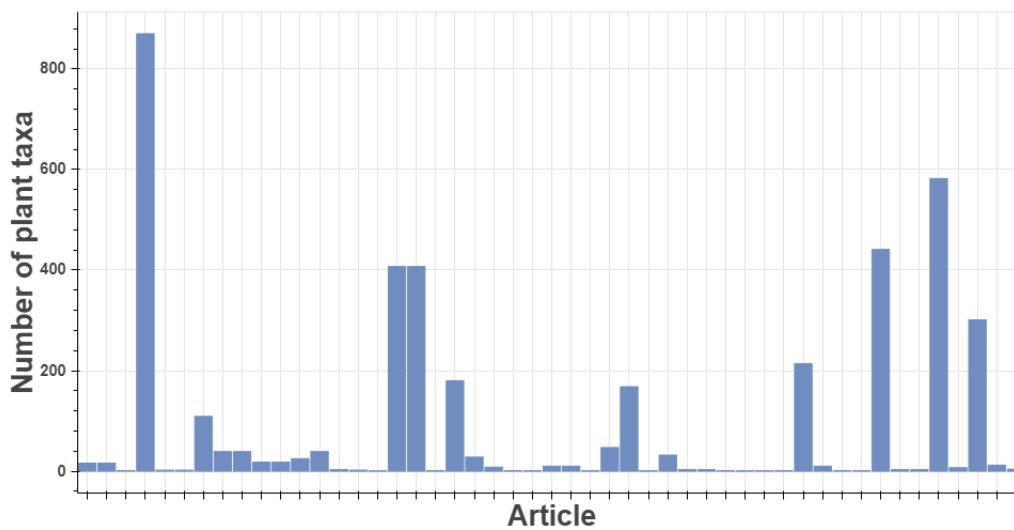


FIGURE 6.8: Number of plant taxa investigated in each article.

Using the same logic as in the last question, the bar charts presented in Figures 6.10 and 6.11 answer this question.

Question 5: The literature tests the hypotheses by examining plant and non-plant taxa. What exactly are these non-plant taxa? How many articles investigated each taxon?

We construct an overview of the different taxa examined in the articles by plotting a bar chart. Each bar represents an investigated taxon, while the bar height represents the number of articles examining each taxon. Figure 6.12

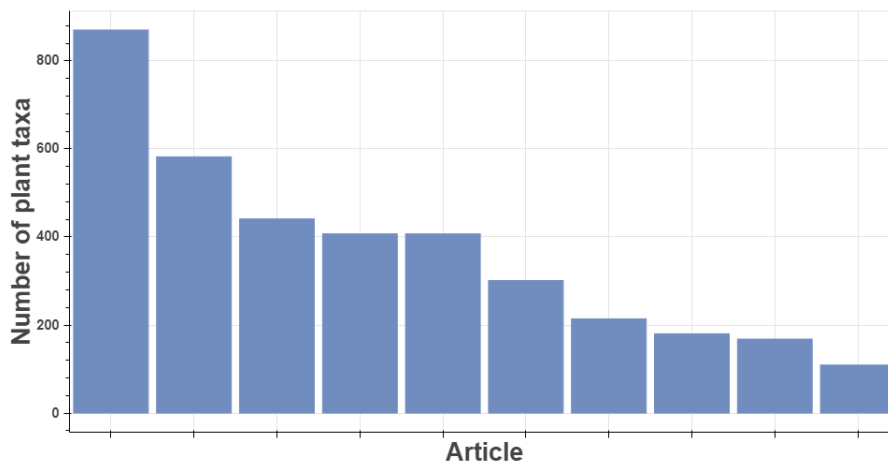


FIGURE 6.9: Top-10 articles with highest numbers of investigated plant taxa.

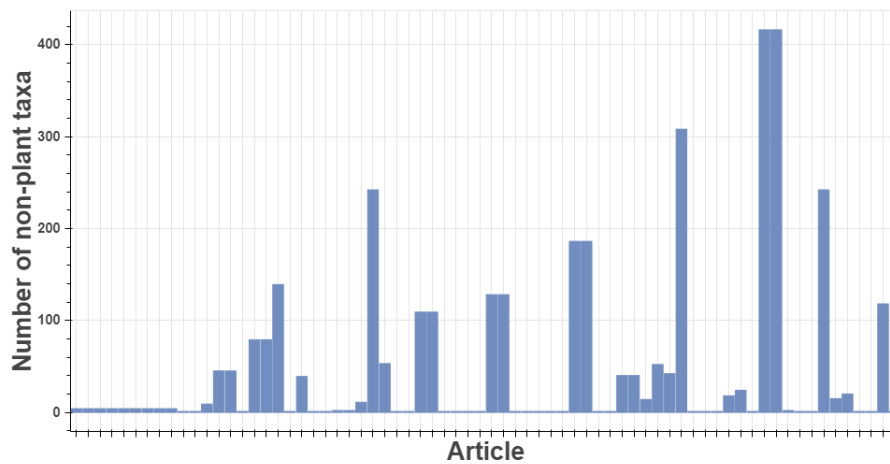


FIGURE 6.10: Number of non-plant taxa investigated in each article.

shows that plants were the most (49) investigated taxa. Other popular taxa include insects, birds, and fishes.

Question 6: When were the relevant articles published? In other words, what are the most productive years in terms of publishing articles related to this hypothesis?

Figure 6.13 summarizes the distribution of articles over the years. A large number of related articles were published between 2009 and 2013. The most productive year is 2012 with 16 relevant articles.

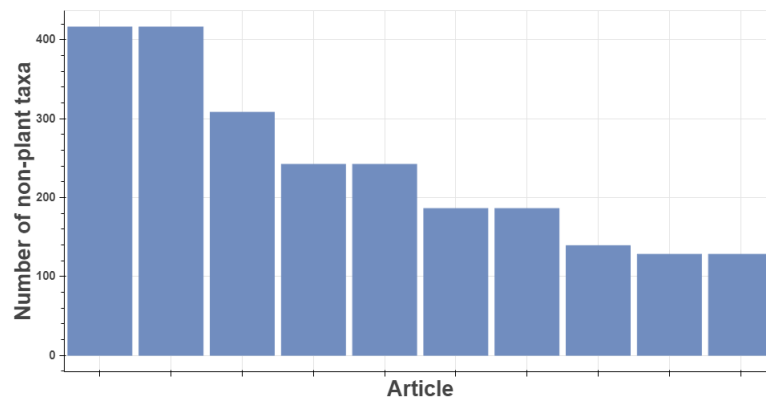


FIGURE 6.11: Top-10 articles with highest numbers of investigated non-plant taxa.

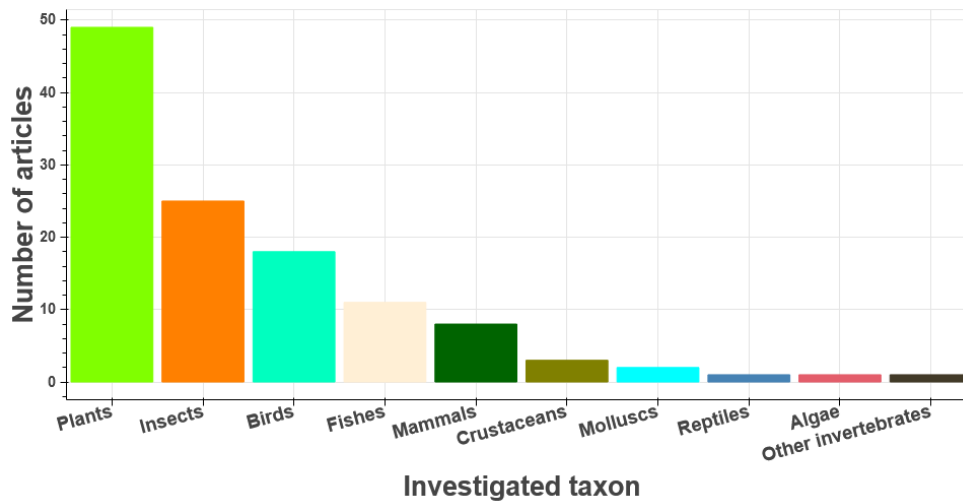


FIGURE 6.12: Number of articles that investigated each taxon.

Question 7: Where did the studies related to this hypothesis take place?

Another interesting point is to look at the distribution of experiments over the continents to discover if there are dominant continents. Figure 6.14 suggests that there is a focus on North America (49 out of 119 contributions). Furthermore, 3 studies were conducted in all continents except Antarctica.

6.3.2 Interactive web application with R Shiny

In the previous section, we exemplarily described one Jupyter notebook that analyses scholarly knowledge related to one hypothesis. As mentioned above,

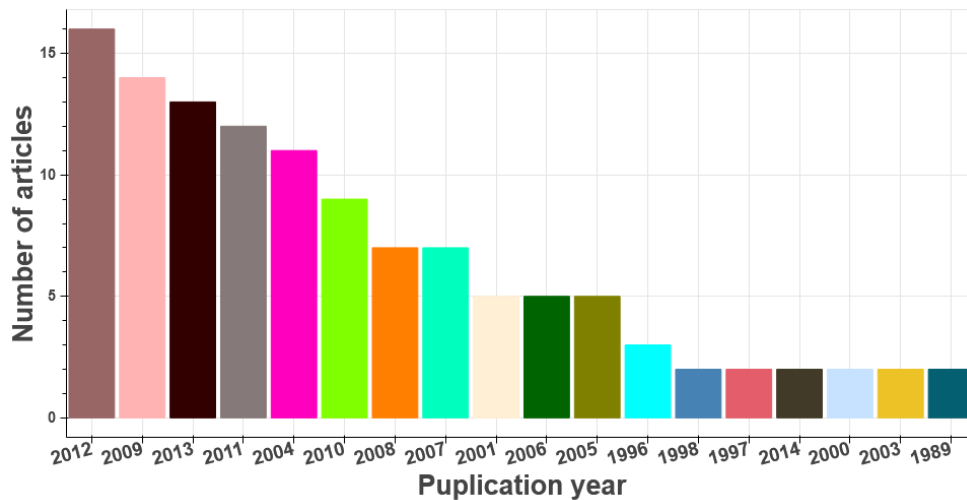


FIGURE 6.13: Distribution of articles over years.

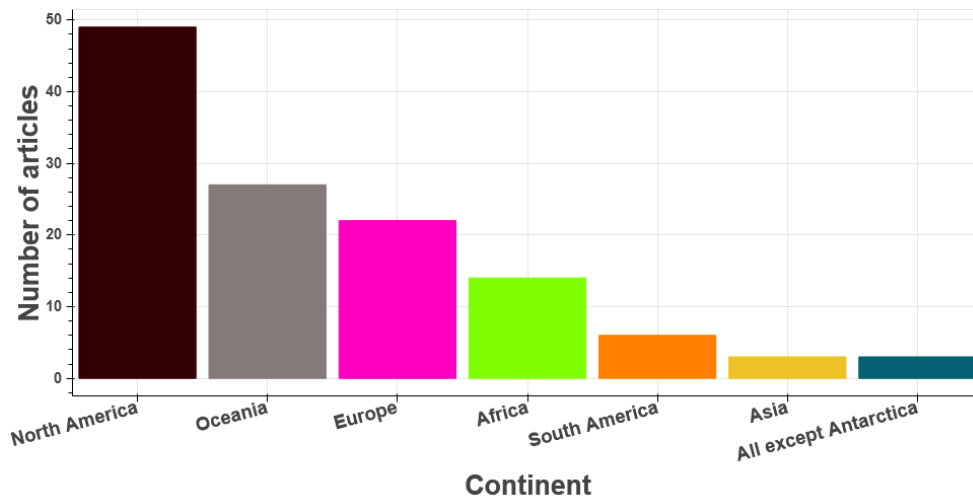


FIGURE 6.14: Distribution of articles over continents.

the analysis of other individual hypotheses can be explored online⁹.

With this approach, it is hard to get an overview of all hypotheses, since we need to open and compare notebooks, which is not a practical solution.

To address this, we have explored the possibility to build an interactive web application using R Shiny. The application is accessible online¹⁰. It provides an interactive overview of ten hypotheses, where users can select a hypothesis from a drop-down menu, then several plots about this hypothesis will be shown interactively. The plots include:

⁹https://mybinder.org/v2/gl/TIBHannover%2Forkg%2Forkg-notebooks/HEAD?filepath=invasion_biology%2Fhypotheses_notebooks

¹⁰https://invasion-biology.shinyapps.io/overview_hypotheses/

- Distribution of articles over continents
- Number of investigated taxa in each article
- Top-5 articles with highest numbers of investigated taxa
- Distribution of articles over years.

6.3.3 Interactive web page using JavaScript amCharts

As an additional and last variant to interact with hi-knowledge scholarly data, we also propose an implementation that supports exploring the connections between hypotheses. Assuming the inter-hypotheses relationships [24, 34] and hi-knowledge website¹¹, and based on the number of contributions supporting, being undecided, or questioning each hypothesis obtained in Jupyter notebooks analysis, we create a web page containing a network visualization implemented as a force-directed tree¹². The force-directed tree is a special type of chart for displaying multi-item data related in a linear, hierarchical, or mixed way. We build this interactive network visualization using JavaScript amCharts. It represents the different hypotheses as nodes and their relations to each other as connections in a graph network. The web page can be visited here¹³.

To clarify the network, we explain its components as follows:

- Each node represents a hypothesis. The name of the hypothesis is shown under the node.
- The size of the node corresponds to the number of scientific articles that discuss the hypothesis. Hence, the larger the node the higher the number of articles related to the hypothesis.
- The nodes with no pie charts represent hypotheses that are not yet tested in the literature.
- The nodes with pie charts represent hypotheses that were tested in the literature.
- Each pie chart is divided into three parts indicating the stand of the relevant articles (green = supporting, red = questioning, grey = undecided).
- When hovering over a node, we can see the name of the corresponding hypothesis, the number of scientific contributions that tested it, and the number of supporting, undecided, and questioning articles.

¹¹<https://hi-knowledge.org/invasion-biology-large/>

¹²<https://www.amcharts.com/docs/v4/chart-types/force-directed/>

¹³https://tibhannover.gitlab.io/orkg/orkg-notebooks/invasion_biology/hypotheses_network_vis.html

- As we saw in the Python Jupyter notebook, some hypotheses have sub-hypotheses. We include these sub-hypotheses in the network as node children. We can click on the node to expand it and explore its children (i.e., its sub-hypotheses).

Chapter 7

Discussion

Conducting scholarly data science is a challenging task due to the lack of structure of articles published in the literature. In this work, we introduced, implemented, and tested an architecture for applying data science to scholarly data. We demonstrated through two use cases that the SKGs, specifically the ORKG, can be exploited in scholarly data science. We performed numerous data science tasks using different tools and methods to draw new insights in the research fields of our use cases and to build new applications and visualizations on top of SKG data. Our work shows that the proposed approach applies to other SKGs and other use cases. This answers our first research question from Section 1.2.

Furthermore, this work assumes SKGs as a given data source. We did not discuss the construction process of SKGs or the technical approaches to build a SKG for data science. Thus, the presented architecture itself addresses our second research question. We discussed how to implement each component in the architecture. The interaction between these components demonstrates the technical approaches of exploiting scholarly databases in general and SKGs (e.g., the ORKG), specifically, in data science. Our implementation is only one way of realizing the architecture. Other implementations may use different set up of the architecture components.

Looking at the results from a larger distance, we can make a few interesting observations, discussed next in terms of advantages and limitations of our work.

7.1 Advantages

Flexibility. An advantage of the proposed approach is the flexibility in terms of conducting highly specialized data science tasks.

To address their research questions, scientists process data in very heterogeneous and typically complex manner. The research problem frames the strategy according to which the data is processed. A generic processing strategy

for all questions is of course not possible to devise. Thus, while specialized user interfaces optimally address one or a small set of tasks, the integration of a data science environment (e.g., Jupyter Notebook) with data sources such as the ORKG for scholarly knowledge offers more flexibility.

An example can be seen in the difference between our approach and the system provided in the hi-knowledge website. The website provides an interactive network visualization of different hypotheses in the invasion biology research area. The visualization is presented at two levels. The first level shows the relationships between the individual hypotheses. If we dive into a hypothesis, we are shown the hierarchy of the selected hypothesis, the number of articles related to the (sub) hypothesis, and the number of articles supporting, being undecided, and questioning each (sub) hypothesis.

While the website introduces a user-friendly interface and a sophisticated visualization of hypotheses, it is built for a specific purpose and does not provide opportunities to analyze the data about the hypotheses or the scientific articles related to them for other purposes. In contrast, our approach enables the data to be processed in arbitrary ways and, thus, to conduct a wide variety of data science tasks such as building applications and visualizations.

Efficiency. Another advantage of exploiting the ORKG in data science is the ORKG Python library (presented in Section 4.2.1). The library enables us to fetch ORKG content directly into Jupyter Notebook. In other words, once data is added to the ORKG, we can easily load ORKG data and directly begin our analysis. With this feature, we can update our Jupyter notebooks and the included visualizations and results on-the-fly by updating the ORKG content used in the analyses and re-running the notebooks scripts.

Furthermore, related to the data science pipeline (mentioned in Section 2.2), since the ORKG Python library supports reading ORKG content directly into native Pandas data frames, we argue that the library supports the acquisition stage and the first part of the preprocessing stage of our pipeline. This demonstrates one side of the efficiency in our approach. Activities including downloading data, importing data into the environment, and transforming data into Pandas data frames are not needed to conduct data science tasks in Jupyter Notebook.

On the other hand, exploiting the MAG or the OpenAIRE research graph, for instance, does not include this advantage. Although these graphs offer the possibility of conducting several analyses on their websites, these do not cover all the analyses a scientist may want to perform. To overcome this obstacle, the creators of MAG and OpenAIRE research graphs provide regular snapshots of their graphs for download. Since it is regular snapshots, a scientist who wants to update her analyses should wait until the next release of

the snapshots. Only then, she can download the graph data and update her results.

Data science on articles content. A third advantage in the implementation of our approach is that we worked on scientific article contents imported from a generic SKG (the ORKG). This enabled the variety and depth of data science tasks we conducted in different domains. This stands in stark contrast to numerous articles (mentioned in Section 3.3) that reported analyzing scholarly metadata.

Conducting data analysis on articles metadata helps scientists to gain insights about research entities described by this metadata such as authors, institutions, journals, and their relationships. Scientists can perform different kinds of analyses on this level such as conference series analysis, research trend analysis, the productivity of authors or institutions, and citation relationships. However, for science it is interesting to analyze the actual content of articles because this content represents the researchers' contributions to advancing knowledge. The work presented demonstrates some possibilities into such direction.

7.2 Limitations

At the time of writing this thesis, the ORKG contains structured descriptions of about 5000 articles in more than 460 research fields¹. Since our implementation relies on the ORKG, the limited amount of content in the ORKG affects the variety of fields, in which data science tasks can be done. Furthermore, while generic KGs cover a wide range of domains and, thus, support conducting data science tasks in numerous domains, they cannot be expected to be comprehensive or aligned to any single domain in particular [32].

Another limitation of our work is that we only focused on a limited set of data science tasks. In other words, our approach is evaluated on standard data science tasks. We focused on descriptive analysis tasks to understand the available data and extract insights and valuable information from it. We did not consider predictive analysis methods in this work.

Moreover, as mentioned in Section 4.2.3, all data science tasks were performed on a local laptop with limited resources. Some data science tasks may include handling large amounts of data with its own challenges for data access and processing. For instance, exploiting several scholarly databases or SKGs such as MAG or the OpenAIRE research graph may introduce some of these challenges due to the large size of the dumps.

¹<https://www.orkg.org/orkg/stats>

7.3 Future work

As mentioned in the previous section, we evaluated our approach using only descriptive analysis tasks. In future work, we would consider extending the evaluation to predictive analysis tasks.

Furthermore, in this work, we exploited scholarly knowledge graphs in data science. Another idea is to exploit data science in SKGs research area. One of the research directions is to convert the huge amount of unstructured and semi-structured data published in numerous formats such as text and tables into graph data. As we discussed in this thesis, several initiatives are working on converting human knowledge into knowledge graphs, whether domain-specific or generic. This leads to a fast evolution of approaches for managing such data, i.e., collecting and processing. Data science can be used to address these issues.

Chapter 8

Conclusion

Nowadays, scientific articles are mostly published as PDF files containing unstructured and semi-structured text. This way of scholarly communication limits the possibilities of reusing scholarly knowledge for other purposes.

Scholarly knowledge graphs (SKGs) represent scholarly knowledge in a structured way. Thus, they enable researchers to navigate, analyze, and gain new insights from scholarly data. As we have described, domain-specific SKGs contain descriptions of articles content in specific fields. This enables conducting scholarly data science in these fields. In contrast, most of the generic SKGs focus on article metadata enabling metadata-level analysis including conference series analysis and trend analysis.

The Open Research Knowledge Graph (ORKG) is a generic SKG that allows describing research contributions in a structured and machine-readable way. The ORKG aims to implement the FAIR principles for scholarly knowledge. This allows for easier reuse in downstream data science of scholarly knowledge published in the literature.

In this thesis, we exploited the ORKG in data science pipelines. We presented and implemented an architecture for applying data science to scholarly data. Then, we tested the architecture and the implementation in two use cases in different domains (COVID-19 and invasion biology). We developed and demonstrated approaches that reuse the ORKG content and, when needed, data from other sources, to enable data science and discover new knowledge in the research areas of our case studies.

In the first use case, most of the needed data was readily available in the ORKG. We refined the ORKG data and then reused it in different tasks such as evaluating model performance using spatial analysis and discovering patterns in the model estimation. Additionally, a spatial analysis of the spread of COVID-19 across Chinese provinces was performed to reveal regional patterns. We also updated the results of an old Guardian study by reusing the ORKG content of different comparison tables. With this update, we compared the contagiousness and deadliness of COVID-19 to other viruses and diseases.

In the second use case, we started earlier in the scholarly data science workflow and firstly ingested scholarly data in the ORKG. Structured descriptions of hundreds of articles in the invasion biology research field were added to the ORKG. Then, we reused this data to analyze the support of hypotheses in invasion biology using an interactive web application and an interactive web page.

Furthermore, we presented several scholarly databases in different research fields. We discussed their technological differences and how we can exploit them in data science.

To summarize, publishing and storing scholarly knowledge in a machine-actionable way and according to the FAIR principles facilitates, among others, the reuse of this knowledge in data science. This enables gaining interesting insights and developing new applications and services across research fields.

Bibliography

- [1] Kritika Agrawal, Aakash Mittal, and Vikram Pudi. "Scalable, Semi-Supervised Extraction of Structured Information from Scientific Literature". In: (June 2019), pp. 11–20. DOI: 10.18653/v1/W19-2602. URL: <https://www.aclweb.org/anthology/W19-2602>.
- [2] Suad Alasadi. "Review of Data Preprocessing Techniques in Data Mining". In: *Journal of Engineering and Applied Sciences* 12 (Sept. 2017), pp. 4102–4107.
- [3] Hany Alashwal et al. "The Application of Unsupervised Clustering Methods to Alzheimer's Disease". In: *Frontiers in Computational Neuroscience* 13 (2019), p. 31. ISSN: 1662-5188. DOI: 10.3389/fncom.2019.00031. URL: <https://www.frontiersin.org/article/10.3389/fncom.2019.00031>.
- [4] Alberto Aleta et al. "A data-driven assessment of early travel restrictions related to the spreading of the novel COVID-19 within mainland China". In: *medRxiv* (2020). DOI: 10.1101/2020.03.05.20031740. eprint: <https://www.medrxiv.org/content/early/2020/03/10/2020.03.05.20031740.full.pdf>. URL: <https://www.medrxiv.org/content/early/2020/03/10/2020.03.05.20031740>.
- [5] R. Angles and C. Gutiérrez. "Survey of graph database models". In: *ACM Comput. Surv.* 40 (2008), 1:1–1:39.
- [6] M. Ashburner et al. "Gene ontology: Tool for the unification of biology". In: *The Gene Ontology Consortium. Nat Genet* 25 (Jan. 2000), pp. 25–29.
- [7] Adam Aspin. "Flat File Data Sources". In: *SQL Server 2012 Data Integration Recipes*. Berkeley, CA: Apress, 2012, pp. 61–131. ISBN: 978-1-4302-4792-0. DOI: 10.1007/978-1-4302-4792-0_2. URL: https://doi.org/10.1007/978-1-4302-4792-0_2.
- [8] Hazeline U. Asuncion. "SourceTrac: Tracing Data Sources within Spreadsheets". In: *Provenance and Annotation of Data and Processes*. Ed. by Paul Groth and James Frew. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–10. ISBN: 978-3-642-34222-6.
- [9] Sören Auer et al. "Towards a Knowledge Graph for Science". In: (June 2018), pp. 1–6. DOI: 10.1145/3227609.3227689.
- [10] S Bamford et al. "The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website". In: *British Journal of Cancer* 91 (June 2004), pp. 355–358. DOI: 10.1038/sj.bjc.6601894.

- [11] Simone Barbosa, Milene Silveira, and Isabela Gasparini. "What publications metadata tell us about the evolution of a scientific community: the case of the Brazilian human-computer interaction conference series". In: *Scientometrics* 110 (Oct. 2016). DOI: 10.1007/s11192-016-2162-4.
- [12] A. Batch and N. Elmqvist. "The Interactive Visualization Gap in Initial Exploratory Data Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 278–287. DOI: 10.1109/TVCG.2017.2743990.
- [13] Maria Biryukov and Cailing Dong. "Analysis of Computer Science Communities Based on DBLP". In: 6273 (Dec. 2010). DOI: 10.1007/978-3-642-15464-5_24.
- [14] Estelle Camizuli and Emmanuel John Carranza. "Exploratory Data Analysis (EDA)". In: (Nov. 2018), pp. 1–7. DOI: 10.1002/9781119188230.saseas0271.
- [15] S Carbon et al. "The Gene Ontology resource: enriching a Gold mine". In: 49 (Jan. 2021).
- [16] Wonhee Cho et al. "Big Data Analysis with Interactive Visualization using R packages". In: Aug. 2014. DOI: 10.1145/2640087.2644168.
- [17] E. F. Codd. "Relational Database: A Practical Foundation for Productivity". In: *Commun. ACM* 25.2 (Feb. 1982), pp. 109–117. ISSN: 0001-0782. DOI: 10.1145/358396.358400. URL: <https://doi.org/10.1145/358396.358400>.
- [18] Jia-Yu Dai et al. "An Efficient Data Mining Approach on Compressed Transactions". In: 40 (Jan. 2008).
- [19] Mariya Dimitrova et al. "OpenBiodiv: Linking Type Materials, Institutions, Locations and Taxonomic Names Extracted From Scholarly Literature". In: *Biodiversity Information Science and Standards* 3 (2019), e35089. DOI: 10.3897/biss.3.35089. eprint: <https://doi.org/10.3897/biss.3.35089>. URL: <https://doi.org/10.3897/biss.3.35089>.
- [20] Daniel Domingo-Fernández et al. "COVID-19 Knowledge Graph: a computable, multi-modal, cause-and-effect knowledge model of COVID-19 pathophysiology". In: *Bioinformatics* (2020). DOI: 10.1093/bioinformatics/btaa834. eprint: <https://academic.oup.com/bioinformatics/advance-article-pdf/doi/10.1093/bioinformatics/btaa834/34731464/btaa834.pdf>. URL: <https://doi.org/10.1093/bioinformatics/btaa834>.
- [21] Jason Dykes and David Unwin. "Maps of the Census: a rough guide". In: *Case Studies of Visualization in the Social Sciences, Technical Report 43* 43 (Jan. 1998).
- [22] Suhendry Effendy and Roland Yap. "Using Community Structure to Categorize Computer Science Conferences: Initial Results". In: July 2017, pp. 297–300. DOI: 10.1145/3110025.3110102.
- [23] Lisa Ehrlinger and Wolfram Wöß. "Towards a Definition of Knowledge Graphs". In: Sept. 2016.

- [24] Martin Enders et al. "A conceptual map of invasion biology: Integrating hypotheses into a consensus network". In: *Global Ecology and Biogeography* (Jan. 2020). DOI: 10.1111/geb.13082.
- [25] Michael Färber. "The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data". In: Oct. 2019, pp. 113–129. ISBN: 978-3-030-30795-0. DOI: 10.1007/978-3-030-30796-7_8.
- [26] Said Fathalla et al. "Analysing Scholarly Communication Metadata of Computer Science Events". In: (Sept. 2017), pp. 342–354. DOI: 10.1007/978-3-319-67008-9_27.
- [27] Said Fathalla et al. "Metadata Analysis of Scholarly Events of Computer Science, Physics, Engineering and Mathematics". In: (July 2018). DOI: 10.13140/RG.2.2.15536.05128.
- [28] Usama Fayyad et al. "The KDD Process for Extracting Useful Knowledge from Volumes of Data". In: *Communications of the ACM* 39 (1996), pp. 27–34.
- [29] Salil Gupta, Vipin Bhatnagar, and S.K. Wasan. "Architecture for knowledge discovery and knowledge management". In: *Knowledge and Information Systems* 7 (Mar. 2005), pp. 310–336. DOI: 10.1007/s10115-004-0153-x.
- [30] Tony Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Oct. 2009. ISBN: 978-0-9825442-0-4. URL: <https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/>.
- [31] M. Indrawan-Santiago. "Database Research: Are We at a Crossroad? Reflection on NoSQL". In: (2012), pp. 45–51. DOI: 10.1109/NBiS.2012.95.
- [32] Nitisha Jain. "Domain-Specific Knowledge Graph Construction for Semantic Analysis". In: June 2020.
- [33] Mohamad Yaser Jaradeh et al. "Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge". In: K-CAP '19 (2019), pp. 243–246. DOI: 10.1145/3360901.3364435. URL: <https://doi.org/10.1145/3360901.3364435>.
- [34] Jonathan Jeschke and Tina Heger. *Invasion Biology - Hypotheses and Evidence*. Apr. 2018. ISBN: ISBN-13: 978 1 78064 764 7.
- [35] S. Kandel et al. "Enterprise Data Analysis and Visualization: An Interview Study". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2917–2926. DOI: 10.1109/TVCG.2012.219.
- [36] J. Kumar and V. Garg. "Security analysis of unstructured data in NOSQL MongoDB database". In: (2017), pp. 300–305. DOI: 10.1109/IC3TSN.2017.8284495.
- [37] Kenneth Kvamme. "Spatial Analysis". In: (Dec. 2018), pp. 1–4. DOI: 10.1002/9781119188230.saseas0544.

- [38] Taesoo Kwon et al. "Next-generation sequencing data analysis on cloud computing". In: *Genes and Genomics* 37 (Apr. 2015), pp. 489–501. DOI: 10.1007/s13258-015-0280-7.
- [39] Ying Li et al. "Domain Specific Knowledge Graphs as a Service to the Public: Powering Social-Impact Funding in the US". In: Aug. 2020, pp. 2793–2801. DOI: 10.1145/3394486.3403330.
- [40] Zhen Liu, Beda Hammerschmidt, and Doug McMahon. "JSON data management-Supporting schema-less development in RDBMS". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data* (June 2014). DOI: 10.1145/2588555.2595628.
- [41] S. P. Lloyd. "Least squares quantization in PCM". In: *IEEE Trans. Inf. Theory* 28 (1982), pp. 129–136.
- [42] James B. MacQueen. "Some methods for classification and analysis of multivariate observations". In: 1967.
- [43] Thomas Mailund. *R Data Science Quick Reference: A Pocket Guide to APIs, Libraries, and Packages*. Jan. 2019. ISBN: 978-1-4842-4893-5. DOI: 10.1007/978-1-4842-4894-2.
- [44] Rosa Meo, Giuseppe Psaila, and S. Ceri. "A tightly-coupled architecture for data mining". In: Mar. 1998, pp. 316–323. ISBN: 0-8186-8289-2. DOI: 10.1109/ICDE.1998.655794.
- [45] Franck Michel et al. "Translation of Relational and Non-Relational Databases into RDF with xR2RML". In: May 2015. DOI: 10.5220/0005448304430454.
- [46] Barend Mons. "What gene did you mean?" In: *BMC bioinformatics* 6 (Feb. 2005), p. 142. DOI: 10.1186/1471-2105-6-142.
- [47] Finn Årup Nielsen, Daniel Mietchen, and Egon Willighagen. "Scholia and scientometrics with Wikidata". In: *Scientometrics 2017*. 2017. URL: <https://arxiv.org/pdf/1703.04222>.
- [48] Allard Oelen et al. "Generate FAIR Literature Surveys with Scholarly Knowledge Graphs". In: (June 2020).
- [49] Philipp Pagel et al. "The MIPS mammalian protein–protein interaction database". In: *Bioinformatics* 21.6 (Nov. 2004), pp. 832–834.
- [50] N.S. Patil et al. "A Survey on Graph Database Management Techniques for Huge Unstructured Data". In: *International Journal of Electrical and Computer Engineering* 81 (Apr. 2018), pp. 1140–1149. DOI: 10.11591/ijece.v8i2.pp1140-1149.
- [51] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic Web* 8 (Dec. 2016), pp. 489–508. DOI: 10.3233/SW-160218.
- [52] Lyubomir Penev et al. "OpenBiodiv: A Knowledge Graph for Literature-Extracted Linked Open Data in Biodiversity Science". In: (2019). DOI: 10.3390/publications7020038. URL: <http://dx.doi.org/10.3390/publications7020038>.
- [53] Usman Qamar and Muhammad Summair Raza. "Data Science Programming Languages". In: June 2020, pp. 153–196. ISBN: 978-981-15-6132-0. DOI: 10.1007/978-981-15-6133-7_8.

- [54] Muhammad Mahbubur Rahman. "Mining Social Data to Extract Intellectual Knowledge". In: *International Journal of Intelligent Systems and Applications* 4 (Sept. 2012). DOI: 10.5815/ijisa.2012.10.02.
- [55] Bibhuti Regmi. *Neo4j Graph database*. Jan. 2021. DOI: 10.13140/RG.2.2.35357.33764.
- [56] Jochen Schirrwagen et al. "Data Curation in the OpenAIRE Scholarly Communication Infrastructure". In: *Information Standards Quarterly* 25 (Jan. 2013), pp. 13–19. DOI: 10.3789/isqv25no3.2013.03.
- [57] V. Senderov et al. "OpenBiodiv-O: ontology of the OpenBiodiv knowledge management system". In: *Journal of Biomedical Semantics* 9 (2018).
- [58] Rebecca Shepherd et al. "Data mining using the Catalogue of Somatic Mutations in Cancer BioMart". In: *Database* 2011 (May 2011). bar018. ISSN: 1758-0463. DOI: 10.1093/database/bar018. eprint: <https://academic.oup.com/database/article-pdf/doi/10.1093/database/bar018/1259630/bar018.pdf>. URL: <https://doi.org/10.1093/database/bar018>.
- [59] Giuliana Spadaro et al. "The Cooperation Databank". In: (2020). DOI: 10.31234/osf.io/rveh3.
- [60] Matthew Swain and Jacqueline Cole. "ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature". In: *Journal of Chemical Information and Modeling* 56 (Sept. 2016). DOI: 10.1021/acs.jcim.6b00207.
- [61] Mohammad Taye. "Understanding Semantic Web and Ontologies: Theory and Applications". In: *Journal of Computing* 2 (June 2010).
- [62] Dominika Tkaczyk et al. "CERMINE – Automatic Extraction of Metadata and References from Scientific Literature". In: *Proceedings - 11th IAPR International Workshop on Document Analysis Systems, DAS 2014* (Apr. 2014). DOI: 10.1109/DAS.2014.63.
- [63] Manabu Torii et al. "RLIMS-P 2.0: A Generalizable Rule-Based Information Extraction System for Literature Mining of Protein Phosphorylation Information". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 12 (Sept. 2015), pp. 17–29. DOI: 10.1109/TCBB.2014.2372765.
- [64] C. Tyagi and S. Batra. "Comparative Analysis of Relational Databases and Graph Databases". In: 2012.
- [65] Sahar Vahdati et al. "OpenResearch: Collaborative Management of Scholarly Communication Metadata". In: Nov. 2016, pp. 778–793. ISBN: 978-3-319-49003-8. DOI: 10.1007/978-3-319-49004-5_50.
- [66] Sahar Vahdati et al. "Unveiling Scholarly Communities over Knowledge Graphs". In: Sept. 2018, pp. 103–115. ISBN: 978-3-030-00065-3. DOI: 10.1007/978-3-030-00066-0_9.
- [67] Wil Van der Aalst. "Data Science in Action". In: Apr. 2016, pp. 3–23. ISBN: 978-3-662-49850-7. DOI: 10.1007/978-3-662-49851-4_1.
- [68] Jake VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. 1st. O'Reilly Media, Inc., 2016. ISBN: 1491912057.

- [69] Chad Vicknair et al. "A comparison of a graph database and a relational database: A data provenance perspective". In: *The 48th ACM Southeast Conference* 10 (Jan. 2010), p. 42. DOI: 10.1145/1900008.1900067.
- [70] Chad Vicknair et al. "A comparison of a graph database and a relational database: a data provenance perspective". In: *ACM SE '10*. 2010.
- [71] Kuansan Wang et al. "Microsoft Academic Graph: When experts are not enough". In: *Quantitative Science Studies* 1 (Jan. 2020), pp. 1–18. DOI: 10.1162/qss_a_00021.
- [72] Lucy Wang et al. "CORD-19: The Covid-19 Open Research Dataset". In: *ArXiv* (Apr. 2020).
- [73] J. Webber. "A programmatic introduction to Neo4j". In: *SPLASH '12*. 2012.
- [74] Mark Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3 (Mar. 2016). DOI: 10.1038/sdata.2016.18.
- [75] David Wood, Richard Cyganiak, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C, Feb. 2014. URL: <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- [76] Hui Yang. *Data preprocessing*. 2018. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.29&rep=rep1&type=pdf>.
- [77] Jingjing Ye et al. "Statistical method on nonrandom clustering with application to somatic mutations in cancer". In: *BMC bioinformatics* 11 (Jan. 2010), p. 11. DOI: 10.1186/1471-2105-11-11.
- [78] He YueShun and Xu Wei. "A Study of Spatial Data Mining Architecture and Technology". In: Sept. 2009, pp. 163–166. DOI: 10.1109/ICCSIT.2009.5234974.
- [79] Asadulla Zaki. "NOSQL DATABASES: NEW MILLENNIUM DATABASE FOR BIG DATA, BIG USERS, CLOUD COMPUTING AND ITS SECURITY CHALLENGES". In: *International Journal of Research in Engineering and Technology* 03 (May 2014), pp. 403–409. DOI: 10.15623/ijret.2014.0315080.
- [80] Andreas Zanzoni et al. "MINT: a Molecular INTeraction database". In: *FEBS Letters* 513.1 (2002), pp. 135–140. DOI: [https://doi.org/10.1016/S0014-5793\(01\)03293-8](https://doi.org/10.1016/S0014-5793(01)03293-8). URL: <http://www.sciencedirect.com/science/article/pii/S0014579301032938>.
- [81] Yangyong Zhu, Ning Zhong, and Yun Xiong. "Data Explosion, Data Nature and Dataology". In: 5819 (Oct. 2009), pp. 147–158. DOI: 10.1007/978-3-642-04954-5_25.
- [82] Jozef Zurada and Waldemar Karwowski. "Knowledge Discovery Through Experiential Learning From Business and Other Contemporary Data Sources: A Review and Reappraisal". In: *IS Management* 28 (June 2011), pp. 258–274. DOI: 10.1080/10580530.2010.493846.