NUI Galway
OÉ Gaillimh

Doctoral Thesis

Generating and Ranking Candidate Data Models from Background Knowledge

Daniela Oliveira

December 16, 2020

Supervisor
Dr. Mathieu d'Aquin

Co-supervisor
Dr. Ratnesh Sahay

External Examiner
Dr. Heiko Paulheim

Internal Examiner
Dr. John McCrae

Insight
SFI RESEARCH CENTRE FOR DATA ANALYTICS

Insight SFI Research Centre for Data Analytics
College of Science and Engineering, National University of Ireland, Galway

ABSTRACT

Knowledge graphs have emerged as a core technology to publish, share, and integrate data on the Web. Contrary to traditional data storage solutions, such as relational databases, one of the core functionalities of Knowledge graphs is that data is not required to adhere to strict pre-defined data models.

Nonetheless, ontologies are commonly used as a key technology to integrate information in knowledge graphs at the semantic level. Creating an ontology to model a domain is not a trivial task and requires significant investment of time and effort. Therefore, data publishers are encouraged to reuse existing ontologies by extending or modifying concepts already described in the domain. However, finding the right ontologies to model a dataset is a challenge since several valid, relevant data models are likely exist without clear agreement between them.

In this thesis, we developed a framework to ease the task of selecting the best data model for a dataset. The framework produces a ranked list of candidate data models that fit the data and are interoperable with a knowledge graph of published RDF data sources. This knowledge graph is obtained by aggregating freely available RDF datasets, extracting their underlying ontology graph, and then enriching its edges to produce a tightly connected graph. We exploit the content and graph structure of this knowledge graph to compute a score that considers the accuracy, interoperability, and consistency of the candidates. The output of the framework is the correspondence between a list of input triple patterns (i.e., ⟨`domain – property ⇾ range`⟩) and a ranked list of candidate triple patterns from the knowledge graph per input triple. These rankings are obtained by combining the three scores into a single triple score. This score combination is weighted and the user has the choice to decide the best weight for each score to best fit their use case or application.

Our experiments show that the framework produces a meaningful set of candidates for different use cases. In these experiments, we test the knowledge graph creation methodology and we present two domain use cases that demonstrate the usefulness of our approach. The experiments show that the framework is able to produce a set of reasonable candidate data models to be presented to the user, and the final choice of data model is controlled by a set of parameters that can be adjusted by a user to fit their use case and preferences. Therefore, our framework assists users in finding a data model that will make the task of annotating data less strenuous to support and sustain the (re)usability of ontologies when creating knowledge graphs on the Web.

CONTENTS

ACRONYMS

AD	Average Node Degree
AGraph	Axiom Graph
AKT	AKT Reference Ontology
AML	AgreementMakerLight
AOR	Ad-hoc Object Retrieval
AP@k	Average Precision@k
BFO	Basic Formal Ontology
BGraph	Base Graph
BIBO	Bibliographic Ontology
BOG	Biomedical Ontology Graph
CC	Connected Components
CCF	Clustering Coefficient
CEA	Column Entity Annotation
CMM	Class Match Measure
CPA	Column Property Annotation
CSV	Comma-Separated Values
CTA	Column Type Annotation
CV	Cohesiveness
DBO	DBPedia Ontology
E-R	Entity-Relationship
ETF	Entity Type Frequency
GO	Gene Ontology
GOG	General Ontology Graph
GT	Ground Truth
IR	Information Retrieval
JSON	JavaScript Object Notation
KGP	Knowledge Graph Patterns
LCC	Largest Connected Component
LOD	Linked Open Data
LOV	Linked Open Vocabularies
MA	Memory Alpha
MAP	Mean Average Precision
MB	Memory Beta

MGraph	Mappings Graph
MRR	Mean Reciprocal Rank
NDCG	Normalised Discounted Cumulative Gain
NER	Named-entity recognition
NP	Neighbourhood Proportion
OAEI	Ontology Alignment Evaluation Initiative
OLS	Ontology Lookup Service
OWL	Web Ontology Language
P@k	Precision@k
PGT	Probabilistic Ground Truth
RDB	Relational Databases
RDF	Resource Description Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language
STE	Star Trek Expanded
SW	Star Wars
SWG	Star Wars: Galaxies
SWTOR	Star Wars: The Old Republic
TF-IDF	Term Frequency-Inverse Document Frequency
URI	Uniform Resource Identifier
VSM	Vector Space Model
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

NAMESPACES

bibframe	http://id.loc.gov/ontologies/bibframe/
bibo	http://purl.org/ontology/bibo/
blterms	http://www.bl.uk/schemas/bibliographic/blterms/
bnb	http://bnb.data.bl.uk/id/
bne	http://datos.bne.es/resource/
bnf	http://data.bnf.fr/ark:/
busco	http://busco.ezlab.org/schema#
dcterms	http://purl.org/dc/terms/
ea	http://rdf.ebi.ac.uk/terms/expressionatlas/
edm	http://www.europeana.eu/schemas/edm/
efo	http://www.ebi.ac.uk/efo/EFO_
faldo	http://biohackathon.org/resource/faldo#
foaf	http://xmlns.com/foaf/0.1/
frbr	http://rdvocab.info/uri/schema/FRBRentitiesRDA/
gnd	https://d-nb.info/standards/elementset/gnd#
gwas	http://rdf.ebi.ac.uk/terms/gwas/
isbd	http://iflastandards.info/ns/isbd/elements/
ncit	http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#
oban	http://purl.org/oban/
obo	http://purl.obolibrary.org/obo/
owl	http://www.w3.org/2002/07/owl#
pgterms	http://www.gutenberg.org/2009/pgterms/
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
ro	http://www.obofoundry.org/ro/ro.owl#
schema	http://schema.org/
skos	http://www.w3.org/2004/02/skos/core#
ss	http://semanticscience.org/resource/
uniprot	http://purl.uniprot.org/core/

DECLARATION

I declare that this thesis, titled "*Generating and Ranking Candidate Data Models from Background Knowledge*", is composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Galway, December 16, 2020

Daniela Oliveira

Daniela Oliveira

ACKNOWLEDGEMENTS

I would like to give a special thanks to my parents and my sister for their continuous encouragement in all my endeavours, specially in this one that brought me so far away from them. A special thanks also to Hugo for his companionship and support through these years. You were the answer to the question I didn't know I had. I would also like to extend a thanks to all the friends that accompanied me on my journey and made it a fun and interesting one.

Finally, I wish to thank Mathieu for his contributions that were essential to make this thesis happen. I would also like to thank Ratnesh and Dietrich for their guidance in the earlier stages of this work. This research was supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 and SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund.

1 INTRODUCTION

Since its inception, the World Wide Web (the Web) has evolved to create a thriving global information space, where content can be published, stored, and accessed. Information is published in textual documents, using natural language, with hyperlinks providing connections to related pages. This information is discovered on the Web with search engines that index documents and retrieve them in response to user queries. Traditionally, search engines analyse the textual content of documents to find candidate web pages that match a query and rank them according to the quality of the links coming and going from each candidate page [1].

More recently, Google has proposed the use of a knowledge graph[1] to search for *things, not strings*. The concept of knowledge graph changed the search engine paradigm by expanding search results beyond textual matches with a connected graph that allows the identification of entities and their relationships. The task of retrieving an entity from a semantic context has been called entity search [2], semantic search [3], and Adhoc Object Retrieval (AOR) [4], i.e., the retrieval of a ranked list of entities from an unstructured keyword query in a knowledge graph. Since Google's announcement, companies have adopted the technology to enhance their services by connecting users and their interests, e.g. Facebook's Entity Graph[2] and LinkedIn's knowledge graph[3]. Efforts in non-profit organisations and research institutes have also been focused on the development of knowledge graphs such as DBpedia [5], YAGO [6], and Wikidata [7]. These open-source projects emphasise free access to data to enable sharing, reusing, and application in different scenarios, including academic research.

The modern concept of knowledge graph has been heavily influenced by the Semantic Web [8], which is an extension of the Web that provides approaches to representing the semantics of data and its metadata simultaneously. The Semantic Web is enabled by a set of best practices known as linked data [9] principles, which ensure that the data is available on the Web in a machine-readable format (e.g. Resource Description Framework (RDF)), following W3C best practices[4].

[1] `https://www.blog.google/products/search/introducing-knowledge-graph-things-not` (Accessed in September 2020)

[2] `https://www.technologyreview.com/2013/02/27/84077/facebook-nudges-users-to-catalog-the-real-world` (Accessed in September 2020)

[3] `https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph` (Accessed in September 2020)

[4] `https://www.w3.org/TR/ld-bp` (Accessed in September 2020)

Contrary to traditional data storage solutions, such as Relational Databases (RDB), one of the core functionalities of linked data, and specifically of knowledge graphs, is that data is not required to adhere to strictly defined data models. Nonetheless, the RDF model provides specifications for publishing data on the Web that not only describe entities and relationships but also focus on enabling interoperability between datasets. Conceptually, RDF data distinguishes between entities, entity types, and properties. Entities are distinct things that physically or conceptually exist in the real world and that can be uniquely identified. Entity types represent categories of entities. An entity can be an instance of one or multiple entity types. Properties define types of associations or interactions between entities (object properties) or between entities and literal values (datatype properties). Properties have a domain and a range. The domain defines the source entity type that is related to the range via the property. The range is another entity type in object properties and it is a literal in the case of datatype properties.

Ontologies, defined as *an explicit specification of a conceptualization* [10], are typically used to model RDF data. Various data publishing platforms and infrastructures advocate that if a database entity is described using a precise ontological resource, it eventually leads to efficient linking and querying over published datasets [11, 12]. When used to model data, ontologies ensure consistency between data sources by providing a schema that formally represents and precisely defines entities and their relationships [13]. These schemas use ontology classes and properties in the data model by assigning classes to entity types and ontology properties to relationships between entity types.

Key barriers to publish linked data, however, include a steep learning curve, difficulty with selecting appropriate ontologies to represent data, inadequate linked data tools, and difficulty establishing links [14–16]. Datasets published using the RDF data model and following the linked data principles require domain and knowledge engineering experts to model and link data sources. Most tools to model data following the RDF data model require a high level of understanding of underlying concepts or a broad knowledge of linked data datasets in a specific domain to fully exploit the potential of linked data. Especially in domains such as the library data domain with long histories of using standards to categorise data, adopting a new standard is not a trivial task, much less adopting a standard that requires a high technical and domain knowledge. Therefore, converting data between formats becomes a time consuming task if the data publishers are required to learn how to work with a new complex set of tools.

Another key barrier for a data publisher, however, is to find the ontologies that best fit their data, but also enable the integration with existing datasets in the domain. A survey conducted over several years with linked data providers found that the third most common barrier to publishing linked data was *selecting appropriate ontologies to represent our data* [14, 15]. If data publishers cannot find appropriate ontologies to model their use-cases, they tend to create new ontologies or extend upper-level ontologies to meet their requirements. Therefore, knowledge from similar domains ends up following different data models that not always focus on interoperability with other datasets in the same domain, making it challenging to integrate data from multiple, existing knowledge graphs, as well as to model new data consistently. This issue has been described in several domains such as life sciences [12], education [17], and library data [18].

We categorise approaches that address this issue as *internal*, where only the input data is considered to find a data model, and *external*, which also considers data models of existing datasets when choosing the data model for an input dataset. Figure 1.1 illustrates the high-level distinction between these categories. Internal approaches create a data model by finding direct correspondences between concepts in a dataset and ontologies considering only the needs of the dataset itself. Data models found with this approach can feature one or more ontologies, extended when necessary, and it is up to the publisher to design the schema that these ontologies will follow. Examples of this approach are found in [19, 20]. Internal approaches rely on the assumption that the ontological resources already exist in the domain and are properly indexed or easily findable to be reused or extended. However, in reality, the availability of ontology repositories with search functionalities is limited [21]. Often the search results over the ontology repositories are ambiguous, with dozens of synonyms matching in different ontologies, as well as a disagreements between search engines when ranking ontological resources in their search results [22]. Due to different naming conventions, textual descriptions, synonyms, and granularity of the ontologies, it is challenging to precisely identify an ontological resource that best describes a given concept.

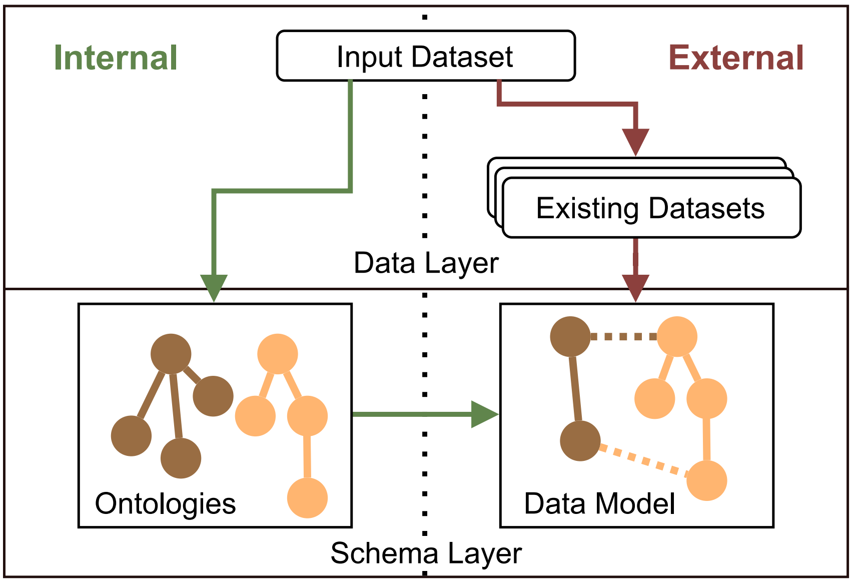


Figure 1.1: Data modelling approaches

External approaches, on the other hand, create a data model based on existing or influential data sources that overlap in domain with an input dataset, extending it to meet specific data requirements. This type of approach is based on the assumption that the data publisher has a clear view of existing domain knowledge and has a well-defined application of the data to match to a specific data model. Examples of this approach are found in [23, 24]. However, the data publisher is still faced with a wide variety of well-known resources, each applying their own data model [18, 25, 26] in the same domain with no clear agreement on which representation is more accepted or used. The limitations of this approach are two-fold: (1) specific data must have been modelled before using high quality resources and the data publisher needs to be aware of these resources, which might require an exhaustive search in the domain to find the most compatible datasets; (2) after selecting a subset of data models to use or extend, it is likely that the final data model will integrate very well with those datasets but might not be easily interoperable with new datasets or datasets unknown to the publisher. Therefore, the challenge for a data publisher is not only finding relevant modelled datasets but also finding the data model that will be most interoperable with existing and future resources to reduce the maintenance effort of the data.

Both approaches present numerous challenges of data and ontology integration. Nonetheless, surveys have found that data publishers want to publish linked data to expose the data to a larger audience, give better access to data, improve metadata sharing and interoperability, and harmonise multiple data sources [15, 16]. Therefore, in this thesis, we aim to investigate how to facilitate the process of finding well-fitted data models that enable interoperability in the domain and we develop a framework that enables this process. Ultimately, the goal of this framework is to reduce the entry level barrier to publish data following the RDF data model and to facilitate the process of creating interoperable data models between datasets in the same or related domains. We show that it is possible to extract consistent and interoperable data models from existing knowledge graphs by exploiting features of the data and the ontologies used to model it. We propose an approach that facilitates the task of finding a suitable data model in domains where disparate data models exist and there is no consensus over modelling standards. We focus on an external approach that gathers existing datasets in the domain to extract their data models. We match these datasets and their data models with the input data and rank candidate data models according to different measures. However, in the cases of missing concepts, we also consider the application of an internal approach to support the external approach in providing a complete integrated data model.

1.1 RESEARCH QUESTIONS

The overall research question of this thesis can be summarised by:

What measures and processes can be used to support the creation of a well-fitted and interoperable data model for a given data source, using a background knowledge graph in the relevant domain?

To answer this question, we propose a set of subquestions: that divide the problem in several smaller challenges:

(RQ1) What knowledge structures can support the extraction of ontology-based data models? (Chapter 3)

- **(RQ1.1)** Can increasing the connectedness of the ontology graph support the computation of interoperability measures for candidate data models?
- **(RQ1.2)** Is it possible to train a classification model to accurately predict datatype properties from literal values using multiple knowledge graphs?

(RQ2) What measures can be used based on the built knowledge structures to select and rank possible data models? (Chapter 4)

- **(RQ2.1)** Can the connections within the ontology graph be used to measure the accuracy and interoperability of possible data models, having a significant effect on the ranking of their components?
- **(RQ2.2)** Can the connections between components of the proposed data models be used to measure their consistency, with a significant effect on their ranking?

(RQ3) Can existing methods for ontology resource retrieval support the process of creating a data model using an internal approach? (Chapter 5)

- **(RQ3.1)** Are Information Retrieval (IR) algorithms effective for retrieving and ranking top-k ontology classes that match a given keyword?
- **(RQ3.2)** Are ontology resource search engines effective for retrieving and ranking top-k ontology classes that match a given keyword?
- **(RQ3.3)** What requirements should be considered when choosing the best technique to retrieve ontology resources when using internal approaches to create a data model?

1.1.1 RQ1: Building Background Knowledge

To extract data models from existing datasets, the knowledge contained in these datasets needs to be structured to facilitate discovery and scoring according to appropriate measures. Two types of knowledge are extracted from RDF data sources: (1) entities and their relationships, and (2) the underlying ontologies that model the data. This knowledge is parsed and structured to be used in the processes of selecting and ranking data models.

This research question is distilled into smaller subquestions that handle specific problems. RQ1.1 investigates the effect of improving the connections between ontology classes and properties to the overall cohesiveness of the graph. The improved connectivity is later exploited for the computation of interoperability in data model candidates.

Finally, RQ1.2 focuses on finding strategies to further enable the generation of data model component candidates, namely datatype properties. These strategies include training classifiers on the knowledge graphs and testing them against the multiple background knowledge graphs for the purpose of enabling the generation of datatype property candidates for input datasets.

1.1.2 RQ2: Generating and Ranking Data Model Candidates

Finding the best ontology resource to describe an entity type or property is a subjective task since concepts can be ambiguously described in an ontology or the same concept might be described differently in individual ontologies. These disparate definitions arise from cases where data publishers create new partial or complete data models, which have term overlap with existing ontologies instead of reusing the existing ontology classes and properties. The focus of this question is in designing methods to obtain and sort the top-k candidates from the knowledge graph to match the concepts in an input dataset. The subquestions focus on measuring the interoperability of the candidates in the knowledge and ontology graph (RQ2.1) and explore how to produce a data model that is consistent with the triple patterns observed in the knowledge graph and is also consistent in its entity type and property recommendations (RQ2.2).

1.1.3 RQ3: Benchmarking Ontology Resource Retrieval Methods

In this thesis, we focus on an external approach to find a data model for an input dataset. However, external approaches not always find data models that are complete, if entity types and properties in the input data are not represented in the background knowledge graph. Therefore, this last research question focuses on exploring the effectiveness of methods for ontology resource retrieval that can bridge the gap between an incomplete external method and internal methods for data model completion.

With this goal in mind, we benchmark different algorithms and tools that can be used for retrieving a ranked set of ontology resources that match a specific keyword. We provide a set of recommendations for the application of these algorithms and tools for different use-cases or as a complement to external approaches, such as the one presented in this thesis.

1.2 METHODOLOGY

We answer the research questions by developing a framework that, for an input dataset, proposes a set of candidate data models extracted from a background knowledge graph built from existing RDF data sources. The candidate data models are constructed by generating and ranking entity types, object properties, and datatypes properties. These elements are combined in consistent ⟨domain – property → range⟩ triples that are interoperable with the data sources included in the knowledge graph.

We experiment with ranking measures to provide an overview of the performance of the framework, with discussions about its potential applications when evaluating is not possible. The final product of the framework is a ranked set of data model candidates that can be used by a data publisher to aid in the task of finding the best data model to fit their specific use-case by providing recommendations of entity type and property annotations. The recommendation is further customisable by having parameters that are easily adjustable so that the data publisher can boost or penalise the ranking scores that are more or less suitable for their use-case.

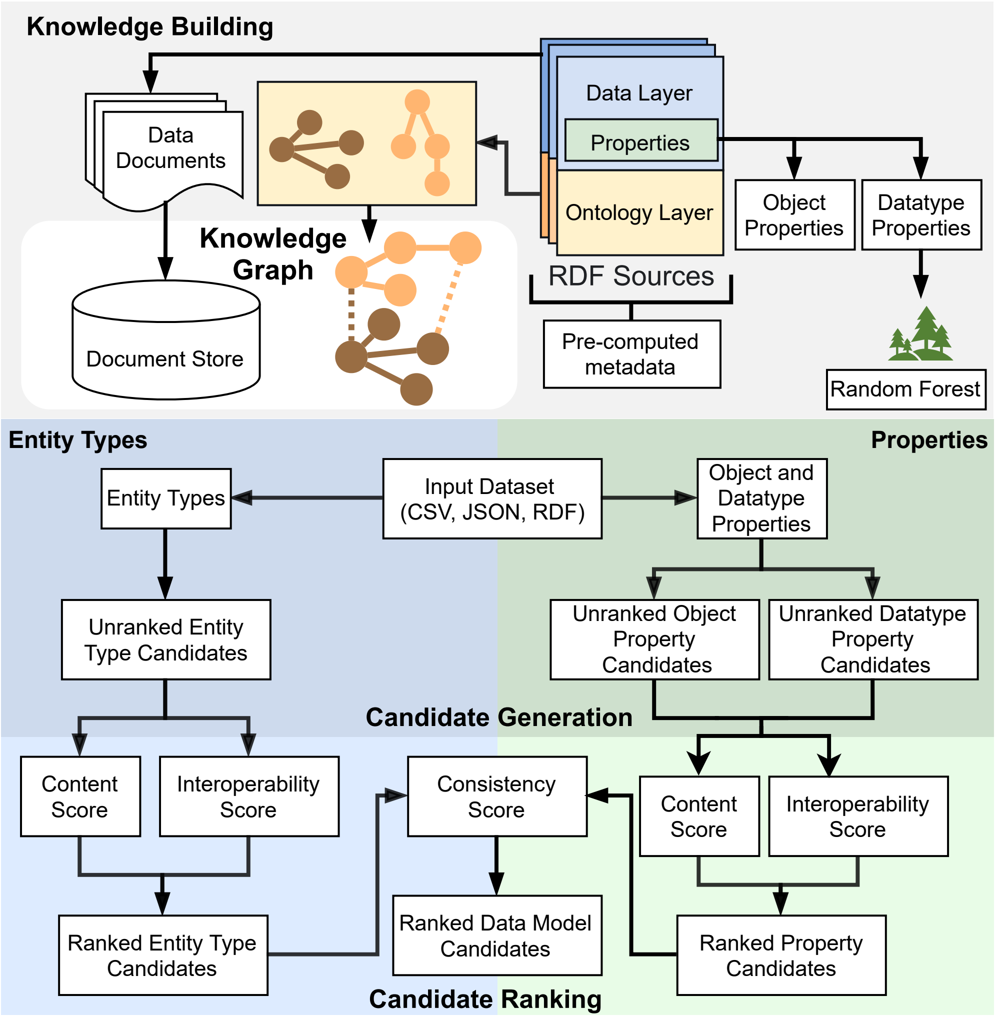


Figure 1.2: Framework diagram

The proposed framework, illustrated in Figure 1.2, has three stages: (1) knowledge building, (2) entity type and property candidate generation and (3) ranking. The **knowledge building stage** focuses on extracting and structuring the necessary knowledge from the RDF data sources to allow the recommendation of data models. From the data layer of the RDF data sources, we extract the entities and their relationships, storing and indexing them in a structured document store. The datatype properties are fed to Random Forest classifiers, creating feature vectors from the values of their properties, and training the models to allow the generation of datatype property candidates. From the underlying ontologies in the RDF data sources we extract the full ontology graph and, through edge enrichment methods, we increase the relevant connections between ontology classes and properties to create a more tightly connected graph that enables the generation and ranking of candidates considering structural properties of the ontologies. The **candidate generation stage** is concerned with extracting data model components (or approximations of these in the case of semi-structured formats) from an input dataset. For entity types this process involves extracting representative entity labels that can be matched against the knowledge graph to extract entity type candidates. For object properties, we identify if any exist in the dataset and match them against object properties in the ontology graph from the edges between ontology classes. Datatype properties are obtained by using the trained model to predict obtained a list of all datatype properties ranked according to the probability of matching an input literal value extracted from the input dataset. The **candidate ranking stage** ranks the generated candidates in terms of their appropriateness to match the input using metrics based on frequency, ontology graph distances, and string similarity. Then, it ranks the candidates according to their interoperability with the background knowledge graph, boosting candidates that are more frequent in the knowledge graph and are better connected in the ontology graph. Finally, the consistency score provides full data model recommendations by aggregating the individual scores of each data model component, together with the frequency of the proposed triple in the knowledge graph. The aggregated score is further refined to ensure that the same candidates are being suggested for the same input across the dataset. This final step allows for a full ranked data model recommendation, where each extracted triple of the input is matched with a ranked list of the most appropriate and interoperable candidates considering the RDF data sources provided as background.

The methodology is supported by experiments that evaluate and analyse the stages of the framework to provide an overview of the strengths and weaknesses and potential applications of the framework. The experiments were supported by test cases with library and biomedical data that could benefit from our framework.

Nonetheless, we also consider the case where the framework is not able to provide a complete data model recommendation based on the background knowledge provided. Therefore, to bridge this gap, Chapter 5 explores how internal approaches can be used instead or as a complement to our proposed framework. We also discuss how the framework developed can aid existing ontology repositories in producing ontology resources that are more targeted towards a specific domain or set of data sources.

1.3 CONTRIBUTIONS

During our research work, we produced several outcomes while answering the proposed research questions. The next sections detail the contributions produced in each phase of the process, including peer-reviewed publications describing our work.

1.3.1 Building Background Knowledge

The main contribution of Chapter 3 includes a methodology to create a knowledge graph from multiple sources supported by a tightly connected background ontology graph. The methods are separated into data indexing and schema extraction and enrichment. First, available RDF datasets are transformed into a common structure, then stored in a document store, and an inverted index is built to allow full-text search queries. We then extract the namespaces of the ontologies used in these documents and construct the underlying ontology graph. This graph is enriched with edges from data and ontology correspondences. The detailed methodologies of the chapter will be described in a paper to be submitted to the Knowledge-Based Systems journal.

We also propose a methodology to characterise the impact of ontology matching enrichment methods in an ontology graph. The results of this analysis were presented at the 1st International Workshop on Knowledge Graph Building co-located with the 16th Extended Semantic Web Conference (ESWC 2019) [27].

1.3.2 Generating and Ranking Data Model Candidates

The task proposed in RQ2 has two major challenges: finding entities in the knowledge graph that match an input entity and ranking their entity types according to set criteria. Therefore, the two main contributions of this task are a methodology to generate candidates and rank them in terms of their content, interoperability with a background knowledge graph, and consistency with the knowledge graph data. The detailed methodologies of the chapter will be described in a paper to be submitted to the Knowledge-Based Systems journal.

We participated in the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference (ISWC 2019)[5] which benchmarked systems matching tabular data to DBpedia. We obtained 3rd place in the entity annotation task, with overall high results in all tasks. We presented a system that shared similar principles to our final framework with a strong AOR component and high graph reliance [28]. The system benchmarked in this competition exemplified a basic application of the framework proposed in this thesis since it only considered one data source (DBpedia). Overall, the participation in this competition

[5] http://www.cs.ox.ac.uk/isg/challenges/sem-tab (Accessed in September 2020)

contributed to a wider understanding of the pitfalls and challenges of dealing with incomplete or ambiguous knowledge graph data.

A demo paper entitled *RICDaM: Recommending Interoperable and Consistent Data Models* [29] was presented at the 19th International Semantic Web Conference (ISWC 2020). The demo is described in this thesis in Section 4.6 and is framed as the output of the framework to show the potential of a user interface.

1.3.3 Benchmarking Ontology Resource Retrieval Methods

In answering RQ3, we identify key search factors for biomedical ontologies together with search requirements. We also provide a set of recommendations to help biomedical experts and ontology engineers in selecting the best-suited retrieval method in their search scenarios. This evaluation allows researchers and practitioners to apply the current search techniques more reliably and can help them to select the right solution for their daily work. More generally, we were able to extrapolate how the performance of ontology resource retrieval methods can be applied to internal approaches or to complement external approaches.

The methods and contributions related to this task are explained and detailed in Chapter 5 and were published in the journal Briefings in Bioinformatics [22].

1.4 THESIS OVERVIEW

This thesis is organised as follows: Chapter 2 provides an overview of the context and foundation for this thesis. Chapter 3 details the approach to build the background knowledge graph from multiple data sources, to extract its underlying ontology graph, and to construct the knowledge structures necessary for the framework. In Chapter 4, we present the methodology and experiments to generate and rank candidate entity types and properties, including an evaluation and analysis of the results. Chapter 5 includes the methodology, experiments, and recommendations reached as an outcome of the analysis of the studied AOR methods. Finally, we present the overall conclusions and future directions in Chapter 6.

2 BACKGROUND

This chapter provides the conceptual foundation for this thesis by defining and explaining fundamental concepts. The topics explored directly relate to the research questions proposed in Chapter 1, which focus on the two overarching topics of representation and retrieval of data, information, and knowledge.

Data, information, and knowledge are not always well distinguished and sometimes are used interchangeably. However, their meanings are distinct and their differences are commonly associated with the Data-Information-Knowledge-Wisdom (DIKW) hierarchy. Ackoff [30] is usually cited as the modern source of the hierarchy, but the definitions and relations between the DIKW concepts have remained contentious among peers. In this thesis, we will adopt the following definitions:

- **Data** is defined as a product of observations represented by symbols. Examples of data are: The Eye of the World; Robert Jordan; Fantasy; 1990.

- **Information** is processed and inferred from data, i.e., it is data with a meaning and context. Information answers questions such as who, what, and when. For example, *The Eye of the World* is a book title, Robert Jordan is a person, Fantasy is a genre, 1990 is a year.

- **Knowledge** is extracted from the combination of experience and insight with information, e.g., The Eye of the World is a book of the Fantasy genre written by the author Robert Jordan and released in 1990.

- Finally, **Wisdom** adds value to Knowledge by including higher-level judgement and not necessarily being applied in intuitive ways. For example, *Since a new TV show on the series is being produced, we should release a new edition of the book The Eye of World*. Wisdom can also include personal judgements such as *I like Fantasy books, The Eye of the World is a Fantasy book, therefore, I want to read The Eye of the World*.

These concepts have been represented in the form of the pyramid [31] shown in Figure 2.1. This hierarchy has been criticised for being basic and not accurately representing reality [32]. It has been proposed that the pyramid should be inverted since wisdom and knowledge drives the collection of information and data [33]. It has also been proposed that it should be bidirectional [34]. However, in practice, the difference between these concepts is not clearly defined which leads to the same term being used

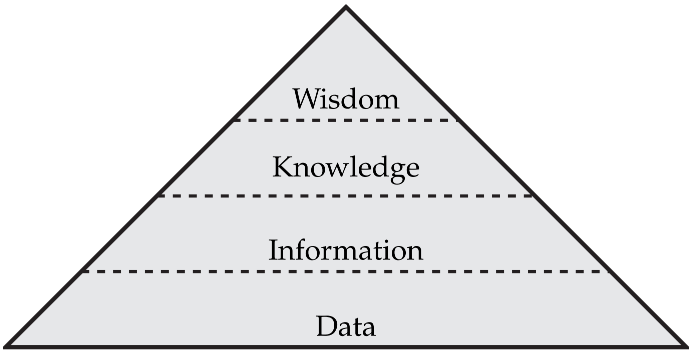


Figure 2.1: Data-Information-Knowledge-Wisdom pyramid

differently by different people. Raw data is rarely available on the Web because without any resemblance of context, the data is hard to find, use, and reuse. Information is more commonly available and shared on the Web. In the last few years, knowledge has become a main topic of discussion in terms of its creation, sharing, and application.

In this thesis, we use published knowledge to enable the structuring of existing information into knowledge that is interoperable with the initial published knowledge. In the next sections, we describe data modelling at different levels and how it influences the structure of data on the Web. Then, we detail ways to retrieve information using IR techniques. Finally, we describe the current state of knowledge representation in the context of linked data and knowledge graphs.

2.1 DATA MODELLING

The advent of the Web has created a thriving environment for publicly sharing data, information, and knowledge. The advances in science and technology, combined with the ease of use of the Web, brought about a data deluge that is consumed by third parties in academia, industry, and public sector alike. Organisations seek this data to extract knowledge that can further their specific endeavours. However, data on the Web is not always published in ways that enable access and reuse to infer information or gather new knowledge. For example, data can be published using proprietary file formats, or be shared in a format that does not lend itself well to the retrieval of data (e.g., PDF). These circumstances hinder the potential for data re-usability and interoperability with other sources on the Web. A challenge when publishing data on the Web is, therefore, to model data following practices that facilitate the use of the data by third parties. Accordingly, data is commonly distinguished between unstructured, semi-structured, and structured depending on its level of adherence to a model specified for a task at hand. The next sections will focus on understanding data models and distinguishing types of data from their level of adherence to a data model.

2.1.1 Data Model

> '[A data model] is a model about data by which a reasonable interpretation of the data can be obtained.' (D. C. Tsichritzis and Lochovsky, 1982)

The term data model was first formulated in the context of a RDB [36]. However, there is no standard or widely accepted definition of data model and the term has been used in the literature with slight variations but similar meanings. For example:

- architecture for data [35]
- set of data requirements [37]
- formal representation of information and means to manipulate said representation [38]

A three-perspective approach was proposed to distinguish between data model instances [39, 40] in conceptual, logical, and physical data models.

Conceptual Data Model

The conceptual data model focuses on providing high-level descriptions of the data and its semantics in the scope of the domain. These descriptions focus on *what* is described by the data. This model is independent of the application of the data and can be reused in different settings covering the same domain. The Entity-Relationship (E-R) data model [41] is commonly used to represent conceptual data models. This model distinguishes entities, entity types, relationships, and attributes. Figure 2.2 shows an example of a simplified view of these components.

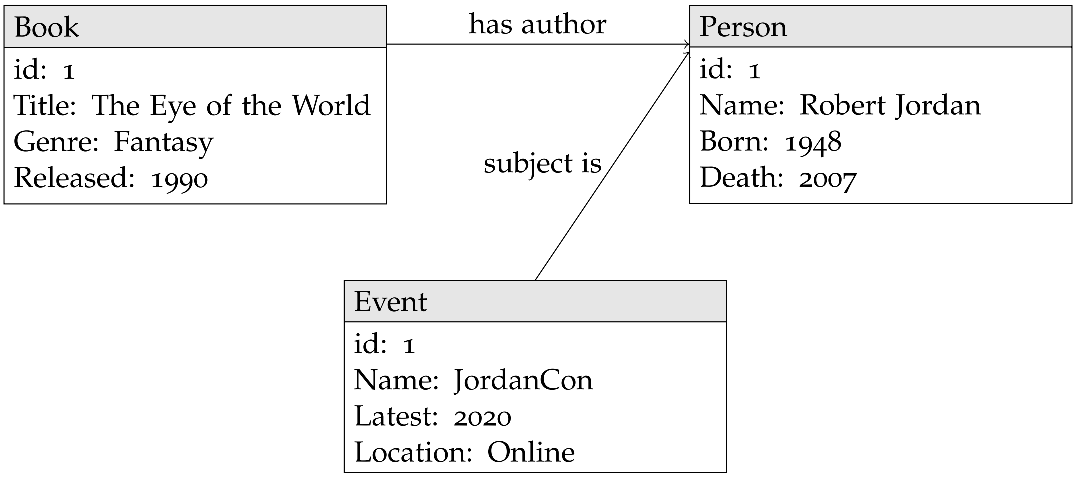


Figure 2.2: Simplified illustration of a conceptual model representing three entities and their relationships

Entities are defined as distinct things that physically or conceptually exist in the real world and that can be uniquely identified. A specific person, book, or event are examples of entities. Entities have instance categories called entity types. In Figure 2.2, three entities are represented. The top left entity is an instance of the type `Book`, while the right entity is of type `Person`. Contrary to the two previous entities, the bottom entity represents an event, which is not a physical entity but represents a real-world instance of an abstract object of type `Event`.

Relationships define associations or interactions between entities or entity types. These connections give further meaning to information. Figure 2.2 shows a relationship between entity types `Book` and `Person` and another between `Event` and `Person`. It is important to note that, depending on the application, a relationship might be considered an entity, e.g., *parent* can represent a `Person` entity with at least one other entity connected to it via a *is_parent* relationship, or it can be the relation between two `Person` entities, i.e. ⟨Child – parent → Father⟩. The strictness of relationships also varies with the application, e.g., the relationship *subject* in Figure 2.2 is applied between an `Event` and a `Person`. This might not always be the case since the subject of an `Event` can be something of a completely different type depending on the dataset.

Finally, **attributes** characterise the properties of an entity or relationship. Values assigned to attributes are used to distinguish one entity from another. Attributes can also act as primary identifiers to distinguish between instances of an entity type. For example, the instance of the entity type `Book` represented in Figure 2.2 has four attributes and the *id* attribute can act as the primary identifier of this entity.

Logical Data Model

The **logical data model** describes the data in terms of data-specific structures, i.e., *how* the data is structured regardless of the storage system. This data model considers the application intended for the data to specify the optimal structure of the data. The classical relational database model [36] is an example of a logical data model by structuring data in tables and relating tables via key constraints. The logical data models that will be prominently featured in this thesis include:

- **key-value model** with the data model being an associative array of key-value pairs. The key serves as a unique identifier for the value. This data model is designed for high performance and horizontal scalability. Table 2.1 shows the example of the `Book` entity from Figure 2.2 modelled with a key-value logical data model.

Table 2.1: Example of a key-value logical data model

key	Value
Book1:type	Book
Book1:title	The Eye of the World, Fantasy, 1990
Book1:genre	Fantasy
Book1:released	1990

- **document store model** is a specification of the key-value store that follows a standard format to encapsulate data into a *document* that is stored in the database with a unique key. Common formats are Extensible Markup Language (XML), YAML Ain't Markup Language (YAML), and JavaScript Object Notation (JSON). This data model extends the data storage capabilities to include additional data and metadata to be associated to the same document. Therefore, this model is desirable when features such as organisation or full-text search are fundamental for an application. Similarly to the Table 2.1, Listing 2.1 shows an example of an entity modelled following the document store model.

```
{"book1": {
    "type": "Book",
    "title": "The Eye of the World",
    "genre": "Fantasy",
    "released": "1990"
    }
}
```

Listing 2.1: Example of a document store logical data model

- **graph model** prioritises relationships between entities by using graph structures to store data. The representation of this model tends to approximate the representation in a E-R diagram since graph models are direct representations of entities and their relationships. Graph data models are particularly useful when data is heavily focused on connections [42] because it allows for fast querying of relationships and intuitive visualisations of the data. Figure 2.3 shows an example of data modelled into a graph structure, where the grey boxes represent nodes with labels and attributes, and the arrows represent edges also with attributes.

- **RDF model**[1] is a specialised graph model, where data is structured as a labelled, directed multigraph. The RDF data model closely approximates the RDB model [43] with some key differences. For example, schemas are created with the intention of being shared to enable interoperability between datasets. RDB schemas are commonly created independently for different databases, which hinders data integration. Additionally, relationships are first-class objects in the RDF model, described by a unique Uniform Resource Identifier (URI). Figure 2.4 shows the running example following the RDF data model. Entities, relationships, and entity types are represented by dereferenceable URIs (in this example URIs are fictional) and structured statements of ⟨subject – predicate → object⟩, known as a triple (see Section 2.2.1 for an extended description of this model). Attributes of subjects are represented as objects of the appropriate datatypes, e.g., string or integer.

[1] https://www.w3.org/TR/PR-rdf-syntax (Accessed in September 2020)

Figure 2.3: Example of a graph logical data model

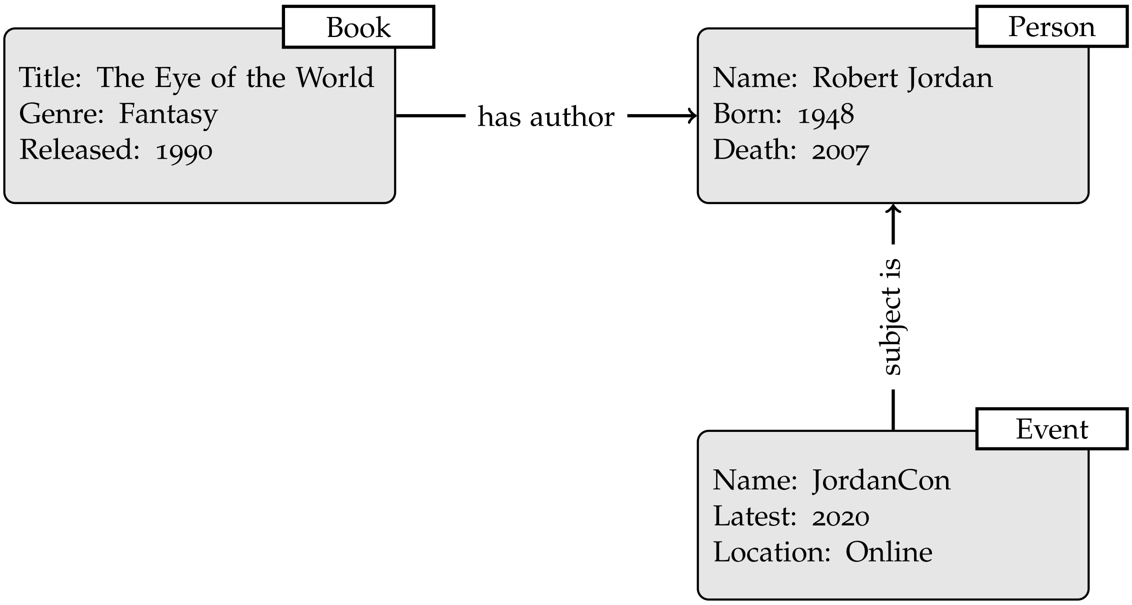
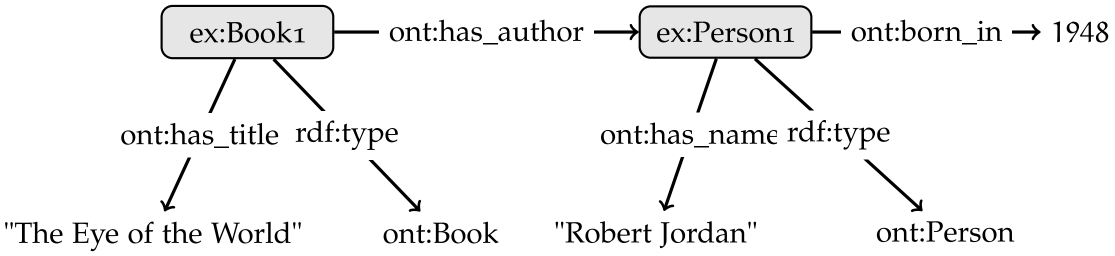


Figure 2.4: Example of RDF data model. The URI namespaces are replaced by prefixes indicated in the top of the figure.

ex: `http://example.org/`
ont: `http://ontology.org/`
rdf: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`



Physical Data Model

The **physical data model** describes data in regard to the requirements of a specific platform chosen to store it, including all artefacts required to create relationships between entity type instances. For example, when a relational data model is chosen, the physical data model will include a specific implementation of keys, constraints, columns names, data types that might differ between database systems.

For example, relational data models are used in Relational Database Management Systems such as Oracle Database[2], MySQL[3], and PostgreSQL[4]. Some notable **key-value**

[2] `https://www.oracle.com/database`
[3] `https://www.mysql.com`
[4] `https://www.postgresql.org`

stores include ArangoDB[5], Couchbase[6], and Redis[7]. **Document store** examples include ArangoDB[5], CouchDB[8], and Elasticsearch[9]. ArangoDB[5], Neo4j[10], and Virtuoso[11] implement physical **graph data models**. Finally, the RDF data model can be implemented in specialised **triplestores** such as Virtuoso[11], AllegroGraph[12], and Blazegraph[13].

The physical data model is the precursor of the schema, i.e., the translation of the physical model to a formal language supported by an implementation of a storage platform of the chosen logical data model. For example, in RDB systems, the common language is Structured Query Language (SQL), while systems implementing the RDF model usually use SPARQL Protocol and RDF Query Language (SPARQL).

Thesis Context

In the context of this thesis, we will be focusing on proposing logical data models, more specifically RDF data models. For the remainder of this thesis, we simplify the term and refer to the RDF data model just as data model. While referring to the logical data model, however, we will use the concepts defined in the E-R data model, which was given as an example of the Conceptual Data Model. The terms entity, relationship, and entity type are easily translatable to the RDF data model, i.e., entity refers to subjects, relationships are predicates, and entity types are the classes usually found as the object of the `rdf:type`[14] predicate.

While the output of our framework is ranked candidate RDF data models, the knowledge graph building process (Chapter 3) and the candidate generation and ranking (Chapter 4 use a document and graph models to structure and store the data.

2.1.2 Data Structuring

Data can be distinguished by its level of adherence to a data model as being **unstructured**, **structured**, and **semi-structured**. In structured data, the structure is clearly provided by a schema, while the opposite happens in unstructured data, i.e., there is no apparent schema to the data. Semi-structured data lies between the two opposites since it does not have an obvious schema, but one can be inferred from the structure of the data. However, these terms are not strictly defined. The next sections will provide more details to each distinction, including possible caveats.

[5] `https://www.arangodb.com`
[6] `https://www.couchbase.com`
[7] `https://redis.io`
[8] `https://couchdb.apache.org`
[9] `https://www.elastic.co`
[10] `https://neo4j.com`
[11] `https://virtuoso.openlinksw.com`
[12] `https://franz.com/agraph/allegrograph`
[13] `https://blazegraph.com`
[14] Namespace prefixes used throughout the thesis are expanded in prefix list in page ix

Unstructured Data

When taken literally, the adjective unstructured means *without structure or organization*[15]. Truly unstructured data would be useless since no one would be able to make any sense of it. Therefore, the term commonly refers to: (1) information that does not follow any perceivable data model, (2) information that cannot easily be stored or translated into any rigid data structure, or (3) structured data, but the data model is not helpful for a specific processing task. These definitions are broad, however, the most relevant point is that distinguishing between structured or unstructured data relies on the use-case.

DBpedia language text in books or emails, for example, follows a grammatical structure that can be inferred and exploited using natural language processing methods (e.g. Klein and C. D. Manning [44]). However, this is not always the optimal format for specific tasks such as applications that require direct access to the entities contained in the text. Extracting entities from natural language text is an active field of research, usually called Named-entity recognition (NER). NER techniques are used to extract meaningful information from text, however, this is not a trivial task. Therefore, it is common to refer to natural language text as unstructured.

Structured Data

Structured data follows a clear data model, which all current and future data must follow. The most common example of structured data is found in RDBs. Designing an RDB requires a database administrator to create a conceptual, logical, <and physical data model that culminates in a strict schema for the data in the database to follow. Structured data systems trade flexibility for reliability. The process of extending a data model beyond its initial design can be strenuous and can compromise the system. Therefore, designing a structured database is an elaborate process since the data model is almost final before the database is populated. This design process means that the flexibility of the system is restricted, making it difficult for the data model to evolve.

Semi-structured Data

Semi-structured data does not follow a strict schema rule, but a structure can be inferred from the data format. The presence of a schema is optional and can even be defined *a posteriori* or adapted as the data evolves. Any physical data model that does not adhere to strict schema rules can be considered semi-structured. Popular semi-structured data formats are Comma-Separated Values (CSV), JSON, and RDF.

One of the most common semi-structured formats on the Web is tabular data, with CSV being one of the most popular formats, especially among governmental open data portals[16]. CSV owes its popularity to the ease of manipulation and the variety of soft-

[15] `https://www.merriam-webster.com/dictionary/unstructured`

[16] As of November 2020, CSV is the most popular data format in the European Data Portal (`https://www.europeandataportal.eu/`)

ware available that can consume and export data in this format. A data file usually includes a record per line (or row) that represents an entity with attributes in subsequent columns, each column separated by a comma.

Due to its ubiquity, CSV is a fast and easy solution to publish data. However, it is a semantically poor format since complex relationships between columns can be lost without higher levels of expressivity.

JSON is a data format that stores data in an associative array of attribute and value pairs. This format facilitates human and machine readability, with a standard structure being exploited to provide higher expressivity to the data. It is programming language independent and can be easily converted to other formats. JSON data can also be considered structured if it follows a specific data model, however, even when it does not, some level of structure can be inferred from the data.

RDF data usually lies closer to structured data than the two previous examples since specifications are provided to publish data in this format. However, these specifications are not always followed, which can make the structure less obvious. In most cases, a structure can still be inferred even if the data model is not provided.

Thesis Context

In this thesis, we handle semi-structured and structured data. The knowledge graph building process handles RDF data that ranges from semi-structured to structured. The framework presented focuses on providing data model candidates for input semi-structured or structured data. The output is always data with a stronger structure, following an RDF data model.

2.2 KNOWLEDGE REPRESENTATION

When data is structured, it provides information from which knowledge can be extracted. Structuring data in ways that are machine readable enables knowledge to be more easily accessible and reused. More specifically, linked data uses RDF data models to structure data in ways that promote the connection of data sources to enable semantic queries from which knowledge is extracted. Ontologies play a major role in supporting information exchange, but standardising and integrating data models using ontologies is not a trivial task. In the next sections, we will provide descriptions of concepts that enable knowledge representation using the RDF data model.

2.2.1 Linked Data

Linked data refers to principles and standards to publish and connect data on the Web [45]. The four main principles of linked data are [9]:

1. **Use URIs to identify things** such as real-world and abstract objects and concepts, extending the scope of the Web to support the identification of more than documents and digital content.

2. **Use HTTP URIs** so that people can dereference objects and concepts online to obtain a description of the object or concept identified by the HTTP URI.

3. The description provided by the dereferenced HTTP URI should **follow a standard data model**, RDF being one of the most widely used when following these principles.

4. **Add links to other URIs** to enable data interconnection and interchange. These links can be between entity types or entities, forming relationships between different datasets and powering the acquisition of integrated knowledge.

Over the years, data publishers have released several datasets following these principles. In 2010, Tim Berners-Lee updated the linked data document to expand it with the description of Linked Open Data (LOD). This initiative encourages data publishers to not only follow linked data principles but also to release their data with an open licence to enable the reuse of the data. LOD should [9]:

1. be published with an open licence

2. be structured with a machine-readable format

3. use a non-proprietary format

4. follow open standards to identify things

5. link their data to existing data

The most notable effort to bring together open datasets is the LOD cloud[17], which displays open datasets published following LOD principles. As of September 2020, the LOD cloud includes 1260 datasets with 16 187 links between them. Data publishers from several domains have contributed with datasets, forming domain sub-clouds, for example, in the geography, government, life sciences, and linguistic domains.

Data analyses of the cloud have shown that well-known vocabularies are more widely used, with proprietary vocabularies becoming less popular with time [46]. However, data publishers do not always conform to linked data publishing guidelines, and only a few provide human-readable metadata for their resources or licensing information [47]. In terms of analysing the links in the cloud, data publishers tend to reuse ontologies. However, ontologies not always follow best practices and often include broken classes and property links [48].

[17] `https://lod-cloud.net` (visited September 2020)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/> .
@prefix ont: <http://ontology.org/> .

ex:Book1
   rdf:type ont:Book ;
   ont:has_title "The Eye of the World" ;
   ont:has_author ex:Person1 .
```

Listing 2.2: Example of RDF data serialised in Turtle format.

Resource Description Framework

Linked data relies on datasets published using the RDF[18] data model. This model allows a rich description of relationships between entities in the dataset and external concepts. These descriptions are provided in the form of statements with a ⟨subject – predicate → object⟩ structure, known as a triple. The subject identifies the resource that is related to the object via the relationship indicated by the predicate.

Elements in triples can be of three types: URIs, literals, or blank nodes. URIs are found in any element of a triple and contain a string of characters that uniquely identifies a resource in the data. Literals are values such as strings, numbers, and dates. A literal can have up to three parts: value, datatype, and language. The datatype should follow the specification to declare the type of the literal in question, and the language in which the literal is expressed can also be declared. Literals can only be represented in objects. Blank nodes indicate the existence of a resource without using a URI, allowing the representation of n-ary relationships between subjects and objects. Concrete syntaxes for the serialisation of the RDF data model have been proposed, such as RDF/XML[19], Turtle[20], N-Triples[21], or N-Quads[22]. Listing 2.2 shows the running example in RDF serialised in Turtle format.

The RDF model directly translates to a graph structure, where subjects are nodes connected to object nodes via predicate edges. An RDF dataset is, therefore, a collection of one or more associated graphs, controlled by the same data publisher body as a single or multiple files [48].

When well-structured, RDF data goes beyond information extraction and facilitates the easier extraction of new knowledge. The less strict schema structure supports the evolution of data since adding new concepts to the schema can be easily achieved. The RDF model focuses on enabling data interoperability and interchange on the Web by providing a set of standards that should be used when modelling data.

[18] https://www.w3.org/RDF
[19] https://www.w3.org/TR/rdf-syntax-grammar
[20] http://www.w3.org/TR/turtle
[21] http://www.w3.org/TR/n-triples
[22] http://www.w3.org/TR/n-quads

2.2.2 Ontologies

RDF focuses on enabling not only links to follow between data sources but should also provide ways to integrate these sources. Integrating data relies on the interoperability of the data models chosen for the data. When data models are not consistent, they increase the domain entropy by creating semantic heterogeneity [49], i.e., the same data in separate contexts is interpreted differently. Ontologies have been proposed as a key solution to semantic heterogeneity in the Web of Data [50, 51]. Ontologies (or vocabularies[23]) are usually defined as *an explicit specification of a conceptualization* [10], i.e., ontologies provide conceptualisations of domains of knowledge and the specification is the concrete representation of that conceptualisation. In information science, the word ontology is applied to a set of logical axioms that model a portion of reality [53].

Ontologies offer a unique combination of features that can benefit enterprise [54] and academia alike [55]. The biomedical domain has been especially keen to adopt ontologies to integrate their data and several applications have been developed based on core ontologies [56] ranging from clinical records [57] to gene nomenclatures [58].

Ontologies can be created with different levels of abstraction [59, 60]:

1. upper-level ontologies (or top-level) describe abstract concepts independent of domain or application, e.g., Basic Formal Ontology (BFO) [61];

2. mid-level ontologies were proposed [60] as the bridge between upper-level and domain ontologies to provide descriptions of more concrete concepts while still maintaining domain independence, e.g., FOAF [62];

3. domain ontologies (or task ontologies) define terms of a specialised domain (e.g., medicine, finance, or publications), e.g., Gene Ontology (GO) [58];

4. use-case ontologies (or application ontologies) define concepts in a specific domain for an individual use-case. Ideally, these ontologies extend domain ontologies with precise concepts needed for the task.

Ontologies can be expressed with different formal syntaxes such as RDF Schema (RDFS)[24] and Web Ontology Language (OWL)[25].

RDFS

RDFS includes upper-level classes and properties to describe categories and their relationships. `rdfs:Class`, `rdfs:Literal`, and `rdf:Property` are used

[23] The distinction between *vocabulary* and *ontology* is not clear [52]. The term ontology is usually reserved for a complex and formally defined set of terms, while vocabulary is used more loosely to refer to collection of terms that might not follow strict formalisms. In this thesis, we will generally prefer the term ontology.

[24] `https://www.w3.org/TR/rdf-schema`

[25] `https://www.w3.org/TR/2004/REC-owl-features-20040210`

when defining types of resources in an ontology. RDFS also defines properties that can be assigned a domain and range with `rdfs:domain` and `rdfs:range`, respectively. These properties allow the attribution of specific entity types and values relations, e.g., ⟨`ont:has_title` – `rdfs:domain` ⇾ `ont:Book`⟩, ⟨`ont:has_title` – `rdfs:range` ⇾ `rdfs:Literal`⟩. These properties ensure logical consistency and allow inferences to be extracted from the data even if not explicitly declared in the data model. In the running example, if `ex:Book1` did not have a declared entity type, it is possible to infer its type from the domain of the property `ont:has_title`.

Furthermore, RDFS defines the hierarchical properties `rdfs:subClassOf` and `rdfs:subPropertyOf` that define subsumption of classes or properties. For example, ⟨`ont:Book` – `rdfs:subClassOf` ⇾ `ont:Work`⟩ defines that `ont:Book` is subsumed by `ont:Work`. `rdf:type` is the property most commonly used to assign an entity type to an entity. An entity that is a member of a category indicated by `rdf:type` is called an instance. In the case of the running example, ⟨`ex:Book1` – `rdf:type` ⇾ `ont:Book`⟩, meaning that `ex:Book1` is an instance of the entity type `ont:Book`. The property `rdfs:label` is commonly used to provide a human-readable name to a resource. For example, ⟨`ex:Book1` – `rdfs:label` ⇾ "The Eye of the World"⟩.

OWL

OWL is built upon RDFS by expanding the ways to express the semantics of concepts. OWL provides a vocabulary to represent richer relationships, which enable higher levels of logical inference. On top of the relations defined in RDFS, OWL provides constructs to describe disjointedness, cardinality, or equality between classes, and provides richer descriptions for properties, such as symmetry and inverse.

Similarly to RDFS, OWL defines a top-level resource called `owl:Class`. This definition should be used instead of `rdfs:Class` when the ontology uses the OWL extended syntax, since RDFS class syntax will limit the inference potential of the ontology.

The OWL specification distinguishes between `owl:ObjectProperty` and `owl:DatatypeProperty` properties. `owl:ObjectProperty` define relations between instances of classes, e.g., ⟨`ont:has_author` – `rdf:type` ⇾ `owl:ObjectProperty`⟩ since `ont:has_author` has an ontology class in both the domain and the range, while ⟨`ont:has_title` – `rdf:type` ⇾ `owl:DatatypeProperty`⟩ forms a relation between an instance of a class and a literal. OWL provides several other constructors to generate complex axioms such as intersection, union, and complement logical relationships.

One of the cornerstones of ontologies is reuse. Therefore, OWL provides `owl:equivalentClass` to denote that two classes have the same instances, being essentially the same class defined in different contexts or ontologies. For example, ⟨`ont:Book` – `owl:equivalentClass` ⇾ `ont2:Textbook`⟩ denotes that `ont:Book` and `ont2:Textbook` have the same meaning even if used in different settings. Similarly, OWL has the same relation for instances where `owl:sameAs` declares two instances to be identical, e.g., ⟨`ex:Book1` – `owl:sameAs` → `ex2:Textbook35`⟩ declares that these two instances with different URIs are referring to the same entity.

Ontology Matching

A common challenge when using ontologies is finding correspondences between resources that have been developed in different contexts and were not specifically developed for interoperability with new or existing resources. For example, an ontology that describes books will have different concepts defined if it was developed by a library or by a service that sells books. While the core entities (e.g., book, author, publisher) should be represented in both, if the data models are designed in parallel, they do not always follow the best practices of reusing or integrating existing ontologies. This issue is found in different domains where several ontologies exist with significant overlap in concepts, but only a small percentage of classes are widely reused from a small group of ontologies [48, 63–65].

Ontology heterogeneity is also relevant when designing a data model that uses several ontologies. An ontology engineer creating a use-case ontology most likely will find the need to integrate different ontologies that have a degree of overlap between them. To create a single data model from different ontologies to extend with their use-case, they need to find correspondences between these ontologies.

Therefore, the process of finding correspondences between semantically overlapping entities in ontologies is called *ontology matching*. The correspondences can have different meanings with equivalence being the most obvious, but others such as subsumption or disjointness being equally relevant. A correspondence is called a mapping and can be found between classes, properties, or instances. Mappings have different levels of precision, usually associated with a mapping score that denotes the confidence on the correctness of the correspondence. The result of an ontology matching task is a set of mappings, which is often called an alignment.

Formally, the pairwise ontology matching process is defined as a function f that computes an alignment A' from a pair of ontologies o and o', considering an input alignment A, parameters p, and resources r as follows [66]:

$$A' = f(o, o', A, p, r) \tag{2.1}$$

The parameters A, p, r are used to extend the definition of the matching process by providing an input alignment to complete the alignment A, by providing a set of parameters p that affect the matching process (e.g., weights, or thresholds), or external resources r that support the matching process such as specific thesauri.

Several ontology matching tools have been developed over the years. The Ontology Alignment Evaluation Initiative (OAEI)[26] is an initiative that aims to help improve onto-

[26] `http://oaei.ontologymatching.org`

logy matching techniques by assessing the characteristics of existing ontology matching systems and performing comparative evaluations between different solutions. The systems are measured on different tasks that evaluate different facets of ontology matching, such as equivalence class matching, large ontology matching, and instance matching.

Over the years, many systems participated in this initiative, and different tasks have been added and excluded has the research field evolved, e.g., in 2018 a task called knowledge graph [67] was added to match instances and schemas between knowledge graphs. In 2019, overall, the OAEI had 20 participant systems [68], each participating in a different number of tasks.

In this thesis, we use methods from the AgreementMakerLight (AML) ontology matching system [69] which has participated in OAEI for several years, while being consistently one of the best performing systems in several tasks.

2.2.3 Representing Heterogeneous Data in RDF

Data transformation from heterogeneous data sources to RDF is a complex task and different solutions have been proposed to deal with it that focus on specific file formats or provide the flexibility to be adapted to several data publishing formats. Lefrançois, Zimmermann and Bakerally [70] shares a list of requirements for generating RDF from heterogeneous data sources, gathered from experience and their use-cases. These requirements include: (1) transform different file formats, including binary, and easily extend with new formats, (2) exploit existing RDF data sources for context, (3) be easy to use by Semantic Web experts, (4) integrate well in a data engineering workflow, and (5) be flexible and maintainable. Our framework aligns well with the use-cases proposed in Lefrançois, Zimmermann and Bakerally [70], and it fulfils the requirements or is easily extendable to fulfil them.

In the same article, Lefrançois, Zimmermann and Bakerally [70] propose SPARQL-Generate, an extension of SPARQL 1.1[27] to transform data into RDF, while fulfilling the requirements they have delineated. SPARQL-Generate works by creating an adapted SPARQL query that, from specific inputs and a RDF data source, maps the data source to RDF while answering the query to produce a data model. The data model is extracted by querying the data model of a RDF dataset that, together with a set of specific data inputs to convert, generates a query that creates a data model to fit the input data.

Similarly, mapping languages are a popular solution that have long been used to map data sources to RDF by creating a set of logical rules that express the relations between entities in a data source. These rules are formalised with a vocabulary that facilitates the translation to RDF. R2RML [71] was proposed as a language to express mappings between RDB and RDF datasets. The RML mapping language [72] expanded R2RML by making the language input format agnostic, while YARRRML [73] was proposed as

[27] https://www.w3.org/TR/sparql11-query

a human-readable format to write these mappings. Tools and mappings for different formats have been developed (e.g., [74–76]), and mapping languages have been applied to convert data sources in different domains (e.g. [77, 78]). In general, the aim of mapping languages is to create RDF data models that fit heterogeneous data sources.

However, the mappings created are either directly extracted from the data source without following a specific existing ontology or when creating the rules, an ontology or a set of ontologies needs to be chosen to map the concepts in the data sources. Our framework extracts these concepts from the existing RDF data sources, facilitating the processing of choosing the most appropriate ontology or set of ontologies for a given use-case. Furthermore, our solution could be integrated with mapping languages since our framework extracts specific vocabularies from existing data. Therefore, our framework could facilitate the creation of mapping rules by suggesting a data model or candidate entity types and proprieties to jump-start the creation of the mapping rules.

Complex systems have been developed on top of mapping languages. For example, LDIF [79] is a system that uses a mapping language to translate different data sources into the same data model. Sadeghi et al. [20] presents a system that builds a knowledge graph from heterogeneous data sources using processes that include a mapper (using a mapping language) and ontology matching to connect the ontologies used. However, the aim of this system is to generate a knowledge graph without apriori knowledge of other datasets already modelled in the same domain. Therefore, the system expects the user to provide a set of ontologies to be mapped to the data via a mapping language. These approaches differ from our proposed framework since they are closer to an internal approach where the desired ontologies are supplied and no background knowledge is provided to support the modelling process. Therefore, similarly to all approaches based on mapping languages, these approaches are based on finding a specific data model determined by an automatic mapping of ontology concepts to the data. Contrary to them, however, our approach focuses on providing a broad view of the possibilities to model specific data in a domain where similar data has already been modelled.

Several methods have also been focused on the internal approach that builds the data model from the ground up by starting with one or more specific ontologies that can model the problem, modelling the data in RDF, and providing post-processing methods to improve the interlinking of concepts. Datalift [80] is an example of such an approach, where several input datasets are converted to RDF, keywords are matched to Linked Open Vocabularies (LOV) [81] vocabularies that are used to model the data. Similarly, Datavore [82] is a framework that, from an input dataset, finds a schema by searching keywords in the LOV repository and uses a set of measures to improve the connections between concepts. KARMA [19] is a system that models a variety of data sources in different formats, with ontologies provided by the users. The system uses a learning-to-rank approach to provide a semi-automatic data modelling technique that improves suggestions as the user selects more concepts for their data model. This system speeds up the process of creating a data model using an internal approach. FuhSeh [23] takes several keywords, searches through RDF data sources, and produces a knowledge graph that matches those keywords and integrates with the data sources.

The biggest challenge with internal approaches to data modelling is finding ontology repositories that gather all relevant ontologies in a specific domain. While general-purpose repositories exist, such as the LOV portal, and some domains have specific ontology repositories, e.g., BioPortal [83] and Ontology Lookup Service (OLS) [84] for the life sciences domain, there is no guarantee that the optimal ontology or ontologies are indexed by these repositories. When following an external approach that starts from available datasets modelled in RDF, the nature of the method guarantees that the relevant ontologies are found since this approach is concerned with finding a data model based on existing ones. However, not all concepts have been modelled in a domain and some domains have greater RDF dataset coverage than others. In these cases, internal approaches are one of the solutions to find or complete a data model. Therefore, in this thesis, our final analysis includes evaluating the performance of internal approaches. In this evaluation, we start from a set of keywords and use different search methods to extract a set of entity type candidates from local or online ontology repositories.

2.3 KNOWLEDGE DISCOVERY

Until now, we have presented and discussed ways to store and structure data, information, and knowledge on the Web. This section focuses on discussing the approaches and problems related with the retrieval of relevant information and knowledge.

2.3.1 Information Integration

The main goal of the efforts to integrate information is to facilitate the extraction of knowledge, e.g., if information exists in isolation, no knowledge can be extracted from it. Therefore, this is an open research field with several challenges [85]:

1. datasets are produced and managed by different authorities and each publisher decides the needs of the data model for their dataset;

2. the same real-world concepts are referenced by different URIs, creating the need to perform disambiguation tasks;

3. complementary datasets in the same domain but modelling different aspects are not easily linked if not properly integrated;

4. datasets can contain errors or out-of-date information that makes the integration task more laborious;

5. datasets in the same domain follow different data models that can include different concept granularities or different names for entity types and properties;

6. integration has to be a continuous process that accounts for the evolution of data and ontologies.

In this thesis, we mostly focus on Challenge 5 with some tasks related to Challenges 1 and 3. In the knowledge graph research domain, these challenges are usually handled by refinement tasks, which are distinguished into completion and error detection [86]. While the latter focuses on adding missing knowledge to the graph, the last handles graph knowledge that is incorrect. These approaches are directed towards specific targets in the graph, e.g., entities, entity types, properties, or literals, and make use of internal and/or external methods, i.e., use only information contained in the graph or also rely on external sources, respectively. See [86] for a detailed survey of proposed approaches for knowledge graph refinement. In this thesis, we do not directly refine the knowledge in the graph, but indirectly, we integrate new data with knowledge graphs, therefore completing the knowledge in the domain. We use a combination of internal and external methods in our proposed framework by using a background knowledge graph to propose candidate models for input data, an ontology graph to facilitate the ranking of candidates, and supervised learning methods to extract information from the knowledge graph that can be used to complete the generation of data model candidates.

2.3.2 Information Retrieval

Information Retrieval (IR) is a broad concept that has been defined as *finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).* [87]

Traditionally, this definition could refer to a librarian searching for a reference within a library collection. With the advent of the Web, however, IR is more commonly understood as a computer process to provide access to unstructured or structured documents, usually within large collections of documents. The retrieval process has, in general, three main stages: (1) indexing, (2) searching, and (3) scoring/ranking. The main purpose of indexing is to improve the efficiency of the search process. An indexer collects, parses, and stores documents to make them easily accessible and matched. The search task starts when a query is received by the system and is matched against the index. The results from this querying process are scored using a variety of algorithms and allow the ranking of the search results to present in response to the query.

Three types of IR systems can be distinguished by their scale [87]:

- *Web search* includes search tasks that can encompass millions of documents available in assorted formats from unstructured to structured. This type of search has specific requirements in terms of indexing and efficiently retrieving relevant documents. Google[28] is one of the most used web search engines available.

- *Personal information retrieval* includes small-scale searches, for example, in a personal computer or email inbox. Here, the challenges include indexing and searching on a wide variety of document types without consuming too many resources to avoid interfering with the user experience in their personal computer.
- *Enterprise or domain-specific search* is usually developed with specific use-cases in mind. Therefore, the indexing and search processes are optimised for expected documents or data. The scale of this type of search is variable and resources are allocated depending on specific needs. This type of search tends to benefit the most from integration between indexes, which, in an enterprise setting, means storing data in databases optimised for that task.

However, arguably another possible type of search is called semantic search, i.e., an IR system that performs queries based on information beyond data symbols. This type of search is usually found in knowledge graph queries, where a search engine has access to semantic characteristics and graphical structure of entities. The task of *Ad-hoc Object Retrieval (AOR)* [4] has been defined has the task of retrieving a ranked list of resources from a knowledge graph in response to an unstructured keyword query. The intention of these queries is to retrieve different aspects in the knowledge graph, e.g., finding entities, entity types, attributes, or relations.

Early AOR approaches focused on entities present in Wikipedia [88], but more systems have been developed and evaluated [89]. AOR engines use well-established IR techniques to retrieve resources. For instance, Swoogle [90], Sindice.com [91], Watson [92], or Yars2 [93] allow searching of ontology resources through user queries. The ranking in these search engines follows traditional link-based ranking methods, in particular, adapted versions of the PageRank algorithm [1], where links from one source of information to another are regarded as a "positive vote" from the former to the latter. Falcons [94] uses a popularity-based scheme to rank concepts and ontologies. Facebook also developed a search system to retrieve entities from their social graph [95].

In this thesis, we use concepts from traditional IR and apply them to the AOR task. Chapter 4 takes advantage of the inverted index built from the knowledge graph to generate data model candidates. In Chapter 5, we apply IR algorithms directly to retrieve entity types from a collection of entity types candidates.

2.3.3 Evaluating Knowledge Discovery Techniques

There are several measures to evaluate knowledge discovery techniques. These are usually obtained by comparing the retrieved information against a ground truth and vary

[28] `https://www.google.com`

in their strictness. In IR, it is common to have measures that evaluate the performance based on the top-k results retrieved, while specific classification approaches usually rely on traditional accuracy measurements, commonly using precision and recall. In this section, we will present the the evaluation techniques that we use throughout this thesis.

Precision and recall are common evaluation measures used in different domains. Precision measures the number of correct results retrieved from all the results obtained, while recall returns the number of correct results retrieved from all possible correct results. These metrics are usually calculated in terms of True Positives (TP) and False Positives (FP), which together form the pool of retrieved results. The TP are the correct results and the FP are the results retrieved that are incorrect. Additionally, False Negatives (FN) represent correct results that were not retrieved. Following this intuition, precision (P) and recall (R) are defined as:

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

$$R = \frac{TP}{TP + FN} \tag{2.3}$$

In IR, it is common to use variants of these measures that consider evaluation only from the top-k results retrieved. For example, Precision@k (P@k) defines a precision measure that evaluates the top-k results in terms of correctness. In this case, the denominator of the precision P is always equal to k, defining P@k as:

$$P@k = \frac{\text{number of relevant resources in top-}k\text{ results}}{k} \tag{2.4}$$

Other precision-based measures include Mean Average Precision (MAP), which is the mean of the average precision for each query q in the set of queries Q. It also has a top-k variant, which considers a cut-off point at result k and is defined as:

$$AP@k(q) = \frac{\sum_{i=1} rel(r_i) \cdot P@i}{k} \,\forall\, q \in Q \tag{2.5}$$

$$MAP@k(Q) = \frac{\sum_{q \in Q} AP@k(Q)}{|Q|} \tag{2.6}$$

Where $rel(r_i) = 1$ if r_i is a relevant resource for the query Q and 0 otherwise, $P@i$ is the precision at top result i.

Finally, Normalised Discounted Cumulative Gain (NDCG) is a standard evaluation measure of ranking quality that allows graded relevance instead of the traditional binary relevance. NDCG involves a discount function to weight the rank for penalising relevant resources that appear in a low position in the search result. The Discounted Cumulative Gain (DCG) is calculated by:

$$\mathrm{DCG}(q) = \sum_{i=1}^{k} \frac{rel_i}{\log_2(1+i)} \tag{2.7}$$

The NDCG is the quotient between the obtained DCG value and the ideal DCG value (iDCG). The iDCG is calculated by sorting the results from most to least relevant.

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{iDCG}} \tag{2.8}$$

Mean Reciprocal Rank (MRR) is another statistical measure to evaluate the ranking obtained by a process for a set of queries. The MRR is the mean of the multiplicative inverse of the rank of the first correct answer, i.e., 1 for the first result, 1/2 for the second, until k for all queries $q \in Q$ as:

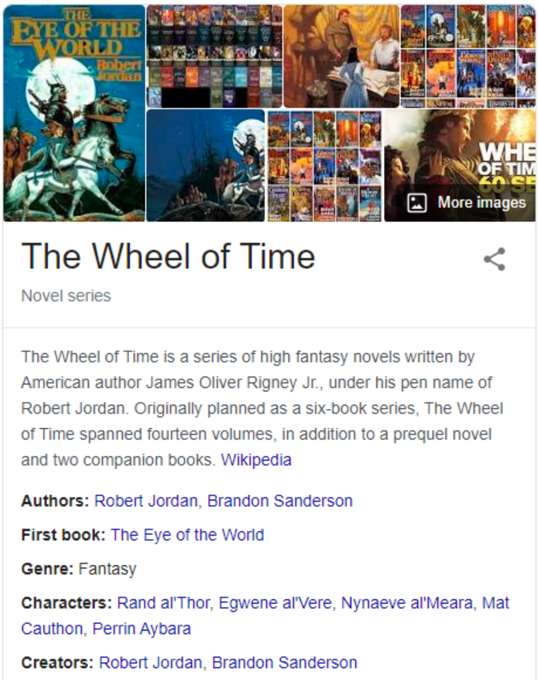
$$\mathrm{MRR@k}(Q) = \frac{1}{|Q|} \sum_{i=1}^{k} \frac{1}{rank_i} \tag{2.9}$$

2.4 KNOWLEDGE GRAPHS

The concept of knowledge graph emerged in 2012 when Google announced that its search engine would be supported by a background graph with semantic connections between entities [96]. The concept was introduced as a way to search *things, not strings*, meaning that keyword searches could now provide deeper results not only about the searched entities but also about the context of the queried entity.

The Google knowledge graph was initially powered by Freebase [97], a collaborative structured knowledge base, but quickly grew from 50 million entities [86] to ≈ 1 billion entities [98]. One of the most well-known applications of the Google knowledge graph is the knowledge panels. Knowledge panels (see Figure 2.5) display relevant information to the search query that can be found in the Google knowledge graph. They provide a quick overview of the context of the query by automatically extracting information from different sources on the Web related to the queried entity. This information can be enriched or verified by relevant authorities via user feedback.

Currently, there is no clear and agreed definition of what constitutes a knowledge graph. Several definitions have been proposed for knowledge graphs and the definitions are centred around different aspects of a knowledge graph. We indicate the following references for a more in-depth analysis of the proposed definitions [99–101], with Hogan et al. [102] proposing a 4-category distinction between the existing definitions: (1) graph where nodes are entities and edges are relationships, (2) knowledge base structured like a graph, (3) compliance with technical characteristics such as having instances, covering multiple domains, and providing reasoning capabilities, and (4) by example.



The Wheel of Time

Novel series

The Wheel of Time is a series of high fantasy novels written by American author James Oliver Rigney Jr., under his pen name of Robert Jordan. Originally planned as a six-book series, The Wheel of Time spanned fourteen volumes, in addition to a prequel novel and two companion books. Wikipedia

Authors: Robert Jordan, Brandon Sanderson

First book: The Eye of the World

Genre: Fantasy

Characters: Rand al'Thor, Egwene al'Vere, Nynaeve al'Meara, Mat Cauthon, Perrin Aybara

Creators: Robert Jordan, Brandon Sanderson

Figure 2.5: Google knowledge panel

Several of the definitions proposed are not compatible between each other and there does not seem to be any direction for consensus. In [98], several leaders of knowledge graph projects came together to publish an article about the direction of knowledge graphs in their settings comparing lessons learned and common challenges. In this article, they refer to a knowledge graph as a structure that *describes objects of interest and connections between them*, in line with the original blog post by Google [96]. They expand their description by saying that it is common to impose constraints on the knowledge graph via schema ontology in a way that can be shared within an organisation and with others and allows inference of new facts.

In this thesis, we adopt a broad definition of knowledge graph fitting in different categories of definitions, where a knowledge graph stores data about real-world abstract or physical entities and their relations structured in ways that enable the extraction of knowledge, e.g., graphs. Our knowledge graph follows the RDF data model and adheres to the Linked Data Principles. We use the same terminology as the E-R conceptual data model to refer to resources expressed by a knowledge graph, i.e., entities represent real-world physical things or abstract concepts, while entity types categorise these entities. The entity types are represented by ontology classes, associated to entities via `rdf:type` predicates. Relationships are indicated by properties directly connecting two entities or, indirectly, by edges connecting two entity types associated with entities.

In the past few years, several knowledge graphs have been openly published on the Web, but also several companies have privately made an effort to use the technology to boost their business. These knowledge graphs are being applied in different contexts with objectives ranging from obtaining a greater understanding of the data to providing a better experience for users and costumers.

2.4.1 Open Knowledge Graphs

Open knowledge graphs refers to knowledge graphs that are published following the Linked Open Data Principles. Some notable cross-domain open knowledge graphs include DBpedia [5], YAGO [6], Wikidata [7], and Freebase [97]. Both DBpedia and YAGO aim to automatically extract and structure information from Wikipedia. The knowledge contained in these knowledge graphs is further enriched with links to external resources and to each other.

Wikidata is operated by the same institution as Wikipedia, the Wikimedia Foundation, and provides a structured way of introducing data in Wikipedia articles. Contrary to DBpedia and YAGO, Wikidata does not automatically extract information from Wikipedia but, instead, provides means for users to use and extend its content, following restrictions that are in place to guarantee consistency across records. Wikidata supports the addition of references to the claims made in the edited data. Wikidata has received broad acceptance and is used by Google in their knowledge graph [103], and has been used in different domains of applications, including Apple's Siri [104].

Similarly to Wikidata, Freebase was a collaborative effort with users editing and expanding the knowledge about the entities and relations in the data. Freebase was acquired by Google in 2010 and propelled the initial launch of the Google knowledge graph, eventually having its content migrated to Wikidata [103].

Domain-specific knowledge graphs have also been released over the years. These knowledge graphs answer specific use-cases that broad domain knowledge graphs do not cover. Notable efforts include Bio2RDF [105] in the life sciences, OpenCitations in the publication domain [106], cultural heritage [107], and in the library data domain [26].

2.4.2 Enterprise Knowledge Graphs

Google popularised the use of the term knowledge graph with their announcement to the migration of their web search to search for *things, not strings*. Since then, several other business have adopted the model for their web search (e.g., Microsoft Bing [108]).

Several other companies are also taking advantage of the technology for their own uses such as pharmaceuticals [109], financial [110], or automatic driving [111]. More specifically, Amazon [112] and eBay [113] have both developed their knowledge graph in the domain of commerce to accommodate for their need of describing products in ways that facilitate their discovery. In the domain of social networks, two prominent actors, Facebook [114] and LinkedIn [115], also announced their knowledge graph to better connect users and their interests and provide useful recommendations of content.

2.5 CHAPTER SUMMARY

In this chapter, we presented the defining concepts of this thesis, providing the foundation for understanding the motivation, contributions, and overall framework that is being proposed. We started by defining the high-level concepts of data, information, and knowledge, and how they can be modelled and structured within their data model. Next, we presented more specific concepts in the area of knowledge representation, including linked data principles, ontologies, and concepts associated with their representation and application. The last high-level conceptualisation was related to knowledge discovery, where we presented the facets of this broad research field that are relevant for this thesis, which include IR concepts and knowledge graph completion approaches. More specifically, we presented the concept of knowledge graphs, including examples of real-world applications of the concept in open and enterprise environments. Finally, we presented state-of-the-art work related to the overall problem being addressed by this thesis. We discussed how our methods and contributions fit in this state-of-the-art and the ways in which they are distinct and provide new contributions.

3 BUILDING BACKGROUND KNOWLEDGE

This chapter details the process of building the background knowledge that facilitates the generation and ranking of the data model candidates. Generating entity type and property candidates that not only accurately match a knowledge graph but also focus on maximising interoperability is not a trivial task. Therefore, in this chapter we describe the structures that enable and optimise the process and we assess their suitability for the task and evaluate the performance of specific key methods. This chapter addresses the following main research question:

RQ1 → *What knowledge structures can support the extraction of ontology-based data models?*

This question is answered by dividing it into the following sub-questions:

RQ1.1 → *Can increasing the connectedness of the ontology graph support the computation of interoperability measures for candidate data models?*

RQ1.2 → *Is it possible to train a classification model to accurately predict datatype properties from literal values using multiple knowledge graphs?*

The answers to these questions are explored by creating several knowledge structures that are extracted from the knowledge graph and hold the necessary knowledge to generate and rank entity type and property candidates. This stage of the framework has four main tasks: (1) extracting metadata to speed-up the lookup of essential information to generate and rank candidates (Section 3.5), (2) building the knowledge graph from heterogeneous RDF data sources (Section 3.6), (3) extracting the ontology graph (Section 3.7) and evaluating the edge enrichment methods (answers RQ1.1), and (4) training, validating, and testing a random forest classification model to generate datatype properties (Section 3.9, which answers RQ1.2).

Therefore, the contributions of this chapter include (1) methods to automatically create a knowledge graph from multiple RDF datasets, (2) supported by a tightly connected background ontology graph, and (3) methods and evaluation of random forest models to classify datatype properties.

3.1 INTRODUCTION

Background knowledge is a generic term to describe *a priori* information that helps to understand or contextualise a problem to facilitate the execution of a task. It helps to make sense of new information by comparing it to prior information and figuring out how the new information fits with the old. The term is applied to different fields from social sciences (e.g., pedagogy [116]) to machine learning and knowledge engineering. In the context of Pedagogy, for example, it may refer to the knowledge students acquire through life experiences and how it affects the execution of tasks such as reading and comprehension. On the other hand, in the field of machine learning, it may refer to a resource that is used in a supervised approach to train a model. In knowledge engineering, background knowledge usually refers to the use of external resources to improve a task. These resources can be, for example, existing datasets or ontologies.

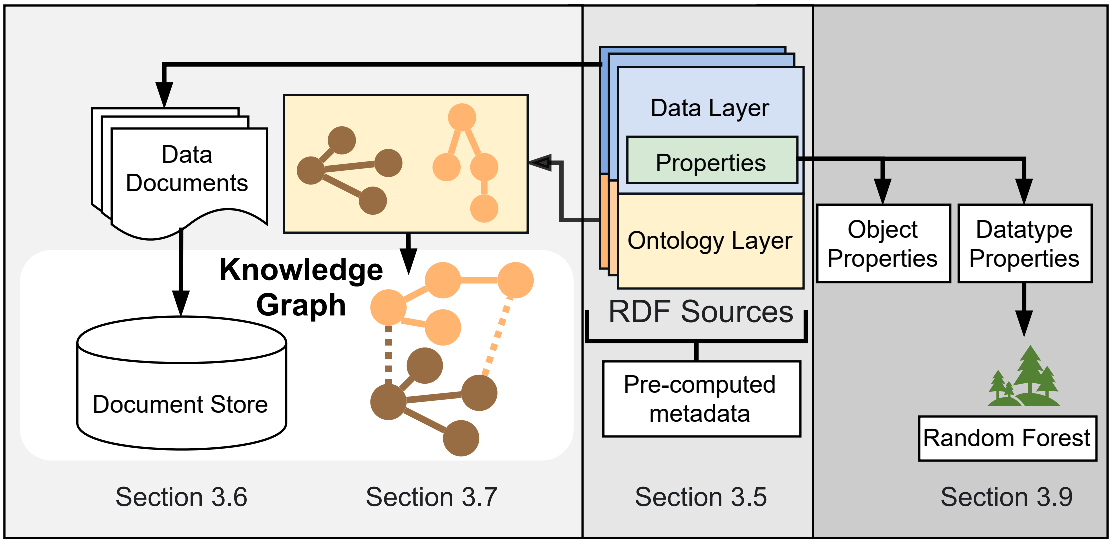


Figure 3.1: Overview of the framework components in this chapter with their respective sections.

In this chapter, we describe the knowledge structures we extract to support our framework, illustrated in Figure 3.1. These structures are built to facilitate the generation and ranking of entity type candidates and properties. The framework has the following requirements: (1) efficiently produce ranked data model candidates, (2) enable AOR search over the literals of the RDF data sources to generate entity type candidates, (3) easily traverse the ontology graph to understand the relations of each entity type and property to rank them based on their interoperability, and (4) generate datatype property candidates from literal objects.

We consider two main sources of knowledge that can be extracted from RDF data sources: the data layer and the ontology layer. The data layer contains the entities and their relationships, and the ontology layer contains not only the entity types and properties used in the dataset but also further logical relationships between entity types that might not be featured in the dataset. Therefore, to answer the requirements of the framework, from the RDF data sources, we extract maps of pre-computed metadata that contain essential information to generate and rank candidates more efficiently. From the data layer of the RDF sources, we build a document store with an inverted index of the literal field values of the RDF data sources that enables full-text search over these resources. This document store is the entry point to the knowledge graph since it facilitates the querying process to obtain valuable information to generate and rank candidates. The ontology layer is combined into a single ontology graph, further enriched with new edges that produce a tightly connected graph to be traversed, enabling the assessment of the distances between vertices and vertex neighbourhoods. The ontology graph contains information about the ontology layer only, with the knowledge graph functioning as the bridge between the data layer and the ontology layer. Therefore, the ontology graph has more extensive knowledge about the entity types and properties in the data models of the RDF data sources since it not only includes the entity types and properties represented in the data layer but also contains the extended relationships of these resources to other entity types and properties in the same domain.

Finally, the values of datatype properties can describe any type of literal data. Assuming that datasets in the same domain have similar datatype property requirements, we train a random forest model with the goal of identifying which properties fulfil each requirement in the source RDF data sources. In this chapter, we describe the methods used to build each one of these structures. We also present experiments that explore their impact and evaluation, and test the effectiveness of the proposed structures.

In the following sections, we will first discuss the state-of-the-work related to these methods. Then we will present the different parts of the methodology, i.e., extracting metadata, building the knowledge graph, creating the ontology graph, and training the datatype property model, followed by experiments and evaluations of the methodology. Finally, we present the conclusions, including limitations and future work.

3.2 RELATED WORK

Several solutions have been developed to create knowledge graphs from a single or multiple data sources and to enrich the links between the entities or entity types in the graph. However, the right approach to create a knowledge graph depends on numerous factors, including the domain of the data, the application, and the actors. For example, DBpedia [5] is a knowledge graph built from information in Wikipedia, with an underlying ontology semi-automatically extracted from the most commonly used Wikipedia information boxes. YAGO [6] is another knowledge graph that extracts knowledge from Wikipedia, but its ontology combines the Wikipedia category system with the WordNet [117] taxonomy. The relations in YAGO are manually evaluated and some also provide links to DBpedia concepts. Both of these knowledge graphs contain a broad domain of knowledge but miss specific definitions of domain concepts necessary in some applications, especially when these concepts are not very well covered by Wikipedia. For example, the DBpedia entity for the book *The Eye of the World* (http://dbpedia.org/page/The_Eye_of_the_World) is well-covered, including entity types from popular ontologies used to model bibliographic data, such as the Bibliographic Ontology (BIBO). However, several other data models have been proposed in the bibliographic domain and, when modelling specifically this type of data, the data publisher might prefer to have their data well integrated with other bibliographically data, which would not always be the case if using the DBpedia data model. In this thesis, we propose an approach that not only extracts a schema from an RDF data but also facilitates the integration of the data with published knowledge graphs.

RDF data sources have a long history of being exploited as background knowledge to perform and enhance other tasks. More specifically, it is common to find ontology matching strategies that use background knowledge to improve the mappings discovered between ontologies [118–120]. These ontologies can be later exploited to provide schema mappings that improve linking in knowledge graphs [27, 121]. In this thesis, we use RDF data sources as background knowledge that allows the extraction and ranking of candidate matches for entity types and properties.

Background knowledge extracted from knowledge graphs has also been applied to perform semantic labelling of numerical values. Approaches have been proposed to identify them [122, 123] using classification and clustering methods. Commonly, tabular data does not follow any strict data model. Therefore, data publishers provide table headers that, even though might refer to the same concepts, are provided with different names. For example, in Z. Chen et al. [124] the authors provide the example of a *latitute* column *versus* the abbreviated *lat* header. The authors of this work use a supervised learning method to predict alternative or missing schema labels to integrate tabular data. They extract different features from the values of each column and train a random forest model to predict future values. The authors conclude that the method performs better on float values and worse on string values. In our proposed datatype property prediction approach, we also apply a random forest model. In addition to similar features proposed by Z. Chen et al. [124], we also extract some features that focus on distinguishing string values, such as number of letters, letter casing, and number of non-alphanumeric characters. However, due to the variety of string datatypes, we do not expect the datatype model to obtain a high precision@1, but we aim for high precision@5 since the models are used to generate an extensive list of datatype property candidates that are further ranked by content- and interoperability-based scores. In this thesis, we propose a method to generate datatype property candidates closely related by the methods developed by Z. Chen et al. [124] since we use a random forest classifier to identify the datatype properties. However, differently from Z. Chen et al. [124], the aim of our work is not to find missing schema labels but instead to produce a list of datatype properties that match a background knowledge graph and can support a data publisher to create a full data model. Therefore, besides considering the results from the random forest prediction, we apply additional content- and graph-based scores that re-rank the candidates to facilitate the task at hand.

3.3 EXPERIMENTAL USE-CASE DATASETS

In the experiments with the framework, we use two running examples as potential use-cases: a library use-case and a life sciences use-case. The library use-case focuses on integrating data across libraries, while the life sciences use-case looks at data from biology repositories and aims to find potential data models for this data. The next sections present both of these use-cases in more detail and include descriptive statistics of the data and metadata of the RDF data sources and input data in each use-case.

3.3.1 Library Use Case

Traditionally, libraries expose their catalogue using Machine-readable cataloguing (MARC)[1] standards, which have been criticised for their restrictions in the description of relationships between entities [125, 126]. Libraries have, therefore, been shifting towards more open, reusable, and interoperable formats by exposing their catalogues in RDF, following LOD principles [45]. However, several schemas have been proposed to structure bibliographic data, but none are widely adopted [18], with different libraries modelling data in distinct ways [25, 26], hindering data interchange between published datasets. Considering all different options, it is also challenging for the library data publisher to select the most appropriate data model when transitioning from traditional standards to LOD models.

Datasets

In our experiments, we distinguish between RDF and non-RDF (CSV and JSON) datasets. The RDF datasets are used to build the knowledge graph but also illustrate the use-case of a data publisher that intends to update a data model. The non-RDF datasets represent the use-case of finding an interoperable data model for data not currently following the RDF data model. Table 3.1 presents the datasets chosen for our experiments.

The RDF group has datasets from 5 European libraries, the Project Gutenberg digital library, and data from the Hardiman Library at NUI Galway. All data is publicly available, except for the University data which was provided in RDF/XML format, automatically converted from MARC 21 to BIBFRAME[2]. The Gutenberg and University data were chosen as the experimental use-cases since they follow data model practices not completely or easily linked to the models of the other libraries. The non-RDF datasets

[1] http://www.loc.gov/marc (Accessed in September 2020)
[2] https://www.loc.gov/bibframe/mtbf (Accessed in September 2020)

Table 3.1: Summary of the source RDF libraries chosen. The dashed line separates RDF (top) from non-RDF (bottom) datasets

Library name	Handle	URL	Downloaded
British Library	British	`https://www.bl.uk`	Nov 2019
Bibliothèque Nationale de France	French	`https://www.bnf.fr`	Dec 2019
Deutsche Nationalbibliothek	German	`https://www.dnb.de`	Feb 2020
Project Gutenberg	Gutenberg	`https://www.gutenberg.org`	Nov 2019
James Hardiman Library	University	`http://www.library.nuigalway.ie`	N/A
Biblioteca Nacional de Portugal	Portuguese	`http://www.bnportugal.gov.pt`	Feb 2020
Biblioteca Nacional de España	Spanish	`http://www.bne.es`	Feb 2020
Open Library (JSON)	OpenL	`https://openlibrary.org`	Mar 2020
DSI Library (CSV)	Institute	`https://dsi.nuigalway.ie`	N/A

include a JSON dataset from the Open Library and a local small-scale CSV example with books and magazines from the library of the Computing and Communications Museum of Ireland located in the Data Science Institute (DSI).

Overall, the datasets contain a variety of records such as books, audio records, and periodicals. Table 3.2 illustrates the problem addressed in this work by showing the entity type chosen in each library to describe book entities. The table shows that each library developed different data models and a book entity has multiple potential entity types. The table also includes the definition of a book in the JSON dataset and the column name with books in the CSV dataset.

Table 3.2: Entity types that describe books in the chosen libraries

Library	Book source types
British	`dcterms:BibliographicResource` / `bibo:Book` / `schema:Book`
French	`skos:Concept` / `frbr:Work`
German	`bibo:Document`
Gutenberg	`pgterms:ebook`
University	`bibframe:Work` / `bibfram:Text`
Portuguese	`edm:ProvidedCHO`
Spanish	`bne:C1003`
OpenL	`/type/work`
Institute	`Title`

Metadata

Table 3.3 presents descriptive statistics of the RDF data sources and target datasets. The columns include the number of entities, entity types, datatype properties, object properties, and data model triple patterns (i.e., ⟨`domain` – `property` ⇾ `range`⟩). Overall, the largest dataset comes from the Open Library, followed by the German Library, and French Library. Our smallest dataset is the DSI library with only 34 unique documents. Despite being a outlier in terms of size, this dataset is useful to showcase that the framework can be applied to small scale cases, even when compared to a large knowledge graph. Furthermore, it shows how the framework can be directly applied to CSV input files. Most libraries use between 15 and 20 entity types, except for the University library which has 43 unique entity types. Both French and German libraries include relators in their properties, i.e., specific properties that relate to a name and a bibliographic resource [127], which is why they show such a large number of object properties. The last column of the table reports the number of unique triple patterns from the Knowledge Graph Patterns (KGP) list.

Table 3.3: Descriptive statistics of the datasets in the library use-case. **# Dt Properties** refers to the number of datatype properties and **# Obj Properties** to the number of object properties

Library	# Entities	# Entity Types	# Dt Properties	# Obj Properties	# Triples
British	17 289 195	15	34	17	51
French	38 100 563	15	93	602	695
German	50 340 156	20	236	187	423
Gutenberg	856 476	7	40	30	70
University	5 236 482	43	293	74	367
Portuguese	2 437 096	2	28	0	28
Spanish	20 752 087	16	163	38	201
OpenL	54 678 367	15	324	5	329
Institute	34	N/A	N/A	0	32

3.3.2 Life Sciences Use Case

Life sciences research is increasingly focused on integration with research areas such as Systems Biology and Translational Medicine, bridging distinct domains to provide novel insights. The need for data integration across domains coupled with the massive amounts of data being produced both by biological and clinical domains poses new challenges. A common strategy to deal with this data deluge involves linking the information to ontologies, making it easier to search through databases and to develop algorithms to process information. Ontologies have been remarkably successful in the life sciences, especially in the biomedical domain, where the Gene Ontology [58] is the most notable success case. The NBDC RDF Portal [128] includes a collection of datasets in the life sciences domain in RDF format. These datasets are characterised by their large size with the NBDC Portal, as of August 2020, reporting 99 billion triples in their portal. The significant computational resources necessary to handle this data is one of the biggest challenges of the domain. Similarly, life sciences ontologies follow the same trend by describing thousands of concepts with complex relationships between them.

Datasets

Due to the challenges of the domain, we selected a small use-case to exemplify the potential of the framework in the domain. However, by being small, and not including all available resources, this use-case has a higher potential of underperforming. Nonetheless, Table 3.4 shows the resources selected for the use-case.

Table 3.4: Summary of the source life sciences resources used. The dashed line separates RDF (top) from non-RDF (bottom) datasets

Library name	Handle	URL	Downloaded
DisGeNET	DisGeNET	`https://www.disgenet.org/`	June 2020
Expression Atlas	EA	`https://www.ebi.ac.uk/gxa/home`	July 2020
GenAge Database	GenAge	`https://genomics.senescence.info/genes/`	August 2020
GWAS Catalog	GWAS	`https://www.ebi.ac.uk/gwas/`	June 2020
Monarch Initiative	Monarch	`https://monarchinitiative.org/`	June 2020
Uniprot	Uniprot	`https://www.uniprot.org/`	August 2020
GDC Data Portal (JSON)	GDC	`https://portal.gdc.cancer.gov/`	August 2020
PharmGKB (JSON)	PharmGKB	`https://www.pharmgkb.org/`	August 2020

The RDF data sources include 6 datasets publicly available. The datasets cover different sub-domains within the life sciences with some overlap between datasets. Overall, the datasets contain genomic, proteomic, and disease data. DisGeNET, Expression Atlas, GWAS, and Uniprot are directly provided by the data publisher in RDF format. The Monarch Initiative gathers data from different data sources to create an integrated data platform and makes this data available in RDF format. The GenAge dataset was obtained via the Bio2RDF project [129], which is an open source project that applies simple conversions to transform data in heterogeneous formats to RDF. Despite being available in RDF already, GenAge does not follow data modelling practices adopted by manually curated datasets in the same domain. Therefore, we apply our framework to propose a data model to bring it in line with the datasets in the knowledge graph.

The non-RDF datasets include case data from projects publicly available in the Genomics Data Commons (GDC) Portal of the National Cancer Institute, USA. The PharmGKB dataset includes drug, pathway, and clinical annotations to study the relationship between genetic variations and how the human body responds to medications. We use the publicly available dumps of this data as input of our framework.

Metadata

Table 3.5 includes descriptive statistics of the data sources of this use-case. Overall, the largest dataset in terms of entities is the Expression Atlas (EA). The largest in terms of data model is the Monarch dataset, which is explained by their integration of multiple datasets in their platform.

Table 3.5: Descriptive statistics of the datasets in the life sciences use-case

Dataset	# Entities	# Entity Types	# Dt Properties	# Obj Properties	# Triples
DisGeNET	4 842 024	22	16	12	28
EA	35 309 936	15	10	11	21
GenAge	3301	8	18	5	23
GWAS	320 224	3	11	6	17
Monarch	24 007 559	66	56	57	113
Uniprot	2 824 754	26	30	18	48
GDC	84 139	12	12	0	12
PharmGKB	26 836	33	33	0	33

3.4 OVERVIEW

The knowledge building methodology of this chapter has four main tasks:

1. extracting metadata: pre-computes the metadata maps that facilitate candidate generation and ranking;

2. building the knowledge graph: extracts the RDF data layer and stores it in a document store to be indexed;

3. creating the ontology graph: extracts the ontology layer and connects it via hierarchical and logical relationships and enriches the graph with edges inferred from the data layer;

4. fitting the datatype property classification model: training and testing the random forest model to predict datatype properties.

The next sections describe each task, including the methods to obtain the background knowledge structures and their analysis or evaluation.

3.5 METADATA EXTRACTION

This task focuses on extracting metadata from the RDF data sources that is not easily retrieved either from the document store or the ontology graph. Therefore, we extract and store useful information in intermediary structures that will later be used to more efficiently generate and rank entity types and property candidates.

First, we create Entity Type Frequency (ETF), a map of the number of times t each resource r (e.g., entity type or property) appears in each of the RDF data sources ds, in the form of $\mathsf{ETF}[ds][r] \leftarrow t$. Considering that some RDF data sources assign more than one type per entity, the sum of frequencies for each data source is greater than the total number of documents stored.

We extend this frequency map to create the Neighbourhood Proportion (NP) map. This map takes each entity type and its frequency and adds the frequency of its direct ancestors and descendants. For example, the frequency of bibo:Book will include the frequency of its superclass bibo:Document and its subclass bibo:Proceedings. Then, we normalise this frequency by the total number of entities in the dataset to obtain the proportion of entities that belong to each neighbourhood cluster.

Finally, we extract the KGP list of tuples, where each tuple contains the triple patterns of the data model of a RDF data source, in the form of ⟨domain – property ⇾ range⟩. Each tuple also includes the frequency of each triple pattern in the data sources.

3.5.1 Library Use-case

Table 3.6 shows the top-3 most common entity types and properties per RDF data in the knowledge graph. Table 3.7 shows how the top-3 frequencies change when applying the methodology to obtain the NP map. Overall, the best connected entity types increased their frequency with related concepts contributing to each other's frequencies and boosting themselves to the top-3. For example, in the German Library, both new entity types in the top-3 are connected with bibo:Document and, therefore, share its raw frequency.

Table 3.6: Top-3 frequencies per library

	Classes		Properties	
Library	Entity Type	Frequency	Property	Frequency
British	dcterms:BibliographicResource	4 305 814	rdfs:label	13 208 476
	schema:Book	4 061 670	owl:sameAs	9 620 153
	blterms:PublicationEvent	4 054 537	schema:name	5 757 535
French	skos:Concept	13 719 911	owl:sameAs	14 120 540
	frbr:Manifestation	10 026 284	dcterms:created	13 963 339
	frbr:Expression	10 026 284	dcterms:modified	13 619 055
German	bibo:Document	13 229 893	owl:sameAs	25 426 926
	gnd:DifferentiatedPerson	5 138 194	dcterms:license	25 029 667
	bibo:edition	4 352 339	dcterms:modified	24 895 254
Portuguese	edm:aggregatedCHO	1 218 548	edm:dataProvider	1 218 548
	ore:Aggregation	1 218 548	edm:provider	1 218 548
	edm:ProvidedCHO	1 218 548	edm:type	1 218 548
Spanish	bne:C1004	10 425 921	bne:OP4001	10 425 921
	bne:C1003	4 598 298	bne:P4001	10 425 921
	bne:C1001	2 014 818	bne:P4016	10 425 921

3.5.2 Life Sciences Use-case

Table 3.8 shows the top-3 most common entity types and properties per RDF data sources in the knowledge graph, while Table 3.9 shows the changes in the entity type

Table 3.7: Top-3 neighbour frequencies per library

	Classes	
Library	`Entity Type`	Frequency
British	`dcterms:BibliographicResource`	25 055 055
	`blterms:PersonConcept`	24 351 957
	`blterms:OrganizationConcept`	21 481 046
French	`foaf:Person`	16 123 825
	`skos:Concept`	16 005 727
	`foaf:Organization`	16 005 727
German	`bibo:Document`	20 970 398
	`lib:BrailleBook`	18 451 383
	`bibo:AudioVisualDocument`	18 451 383
Portuguese	`edm:aggregatedCHO`	1 218 548
	`ore:Aggregation`	1 218 548
	`edm:ProvidedCHO`	1 218 548
Spanish	`bne:C1006`	12 063 212
	`bne:C1005`	12 063 212
	`bne:C1004`	12 063 212

frequencies after applying the methods to obtain the NP map. These tables show an overview of the types of entities featured in life sciences use-case datasets, which describe particular concepts within the same or related domains. Similarly to the library use-case, there is some overlap between the concepts described in the datasets, but they use different data models to describe the same concepts (e.g., Gene in DisGeNET is annotated with `ncit:C16612`, while Monarch uses `obo:SO_000704`). However, Table 3.8 shows the broader variety of concepts that are encompassed in the chosen datasets. For example, the EA dataset includes entity types that describe gene expression entities, while Uniprot focuses on annotating concepts relevant in the domain of proteins. The goal of constructing a broader knowledge graph is to allow for less specific input datasets to be supplied to the framework and increase the potential of obtaining a data model.

Table 3.9 shows that, for the most part, the frequencies of the entity types do not change as much as in the library use-case. This is a consequence of these datasets focusing their entities on a smaller number of relevant entity types, leading to the most frequent entity types not having a rich neighbourhood in terms of frequent entity types. For example, the EA dataset has $\approx$ 35M entities and $\approx$ 25M of those are of type `ea:BaselineExpressionValue`. Another factor influencing the neighbourhoods is the difference in domains. Since more domains are being covered, frequent entity types are more likely to be separated in the knowledge graph.

Table 3.8: Top-3 frequencies per life sciences datasets

	Classes		Properties	
Dataset	Entity Type	Frequency	Property	Frequency
DisGeNET	ncit:C25338 (Score)	839 183	rdfs:label	4 758 076
	ss:SIO_001121 (gene-disease biomarker association)	737 781	rdfs:comment	4 758 076
	ss:SIO_001122 (gene-disease association linked with genetic variation)	469 471	dcterms:title	4 755 657
EA	ea:BaselineExpressionValue	25 363 764	rdfs:label	35 305 428
	ea:IncreasedDifferentialExpressionRatio	3 885 675	ea:pValue	7 585 072
	ea:DecreasedDifferentialExpressionRatio	3 699 397	ea:tStatistic	5 530 508
GWAS	owl:NamedIndividual	320 223	rdfs:label	320 223
	gwas:TraitAssociation	179 364	ro:part_of	180 176
	gwas:SingleNucleotidePolymorphism	131 639	oban:has_object	179 364
Monarch	oban:association	10 199 705	oban:association_has_subject	10 199 705
	owl:NamedIndividual	2 526 713	oban:association_has_object	10 199 705
	faldo:Region	1 413 810	oban:association_has_predicate	10 199 703
Uniprot	uniprot:Proteome_Component	316 971	rdfs:comment	1 731 585
	uniprot:Proteome	292 464	rdfs:seeAlso	615 111
	busco:Score	190 503	foaf:page	448 465

Table 3.9: Top-3 neighbour frequencies per life sciences datasets

	Classes	
Dataset	Entity Type	Frequency
DisGeNET	ncit:C25338 (Score)	839 183
	ss:SIO_001121 (gene-disease biomarker association)	737 781
	obo:GENO_0000476 (variant)	492 664
EA	ea:BaselineExpressionValue	25 363 764
	ea:IncreasedDifferentialExpressionRatio	3 885 675
	ea:DecreasedDifferentialExpressionRatio	3 699 397
GWAS	gwas:Chromosome	640 690
	gwas:TraitAssociation	640 690
	gwas:SingleNucleotidePolymorphism	640 690
Monarch	owl:NamedIndividual	12 136 294
	obo:GENO_0000002	12 135 702
	obo:SO_0001583	11 682 197
Uniprot	uniprot:Proteome_Component	316 971
	uniprot:Proteome	312 064
	uniprot:Reference_Proteome	312 064

3.6 BUILDING THE KNOWLEDGE GRAPH

The knowledge graph is built from several RDF data sources that are assumed to (1) be previously published, (2) follow the RDF data model, (3) cover a similar or related domain to the data that needs to be modelled and (4) include equivalences for entities and properties to be modelled.

However, finding a dataset that fits all these parameters is not a trivial task. Nonetheless, a data publisher is usually an expert on the data being published and should be aware of similar datasets already published. In cases where this condition is not verified, significant research has been published over the years covering the topic of finding similar or related datasets on the Web. Different approaches have been proposed using content and metadata to extract similarities between datasets, for example, using probabilistic classifiers [130] or dataset profiles [131]. These works usually follow a dataset recommendation task that, for a given input dataset, rank candidate datasets in order of similarity to that dataset by extracting characteristics of the data and metadata. For a review on this subject, please refer to Mountantonakis and Tzitzikas [85].

The research developed for dataset recommendation can be used in conjunction with our approach since the first task is to find relevant RDF datasets to match some input data. Dataset recommendation tasks facilitate this search by enabling the data publisher to obtain several relevant datasets that will maximise the similarity, integration, and consistency of a data model to existing published datasets modelled in RDF.

In the next task, the set of chosen reference datasets are used to build the document store. Our implementation uses Elasticsearch[3] to store, index, and save the documents. This implementation was chosen due to the effective search engine integrated with Elasticsearch, which enables AOR of entities in the knowledge graph. Therefore, due to the requirements of this platform, we converted the RDF data sources to a JSON format, where the attributes are the properties and the values are the objects of those properties. In our implementation, we used the Raptor RDF Syntax Library[4] which converts several RDF file formats into a common JSON structure. Each document is further pre-processed to facilitate the extraction of relevant information about each entity. The main task in this pre-processing step is adding each blank node to its source entity and storing both in the same document. Blank nodes are anonymous entities, i.e., an entity without URI, that can be considered independently of other entities. However, in the context of our framework, gathering the entities with their blank nodes facilitates the aggregation of information that needs to be matched with entities during the candidate generation stage. Listing 3.1 shows an example of a fictional extract of RDF data, serialised in Turtle format, with a triple with an anonymous blank node ⟨`blank_node – ont:has_name –▹ Robert Jordan`⟩. The pre-processing step guarantees that when creating a data model, we consider entities as a whole, i.e., include all properties and values. Therefore, each document is directly independent of other documents but maintains the relationships to other documents through object properties. Additional pre-processing includes, excluding documents that do not have a `rdf:type` property and creating an inverted index from the literal objects in the triples. Listing 3.2 shows the same example in the document store after parsing and pre-processing.

[3] `https://www.elastic.co/elasticsearch` (Accessed in September 2020)
[4] `http://librdf.org/raptor` (Accessed in September 2020)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/> .
@prefix ont: <http://ontology.org/> .

ex:Book1 rdf:type ont:Book , ont:Work ;
   ont:has_title "The Eye of the World" ;
   ont:has_author [
      ont:has_name "Robert Jordan"
              ] .
```

Listing 3.1: Example of source RDF data serialised in Turtle format with one blank node.

```
{
    "uri": "http://example.org/Book1",
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#type":
        [{"value": "http://ontology.org/Book", "type": "uri"},
        {"value": "http://ontology.org/Work", "type": "uri"}],
    "http://ontology.org/has_title":
        [{"value": "The Eye of the World", "type": "literal"}],
    "http://ontology.org/has_author": [
        {"http://ontology.org/has_name":
            [{"value": "Robert Jordan", "type": "literal"}]
        }
    ]
}
```

Listing 3.2: Example of document store data model.

3.7 CREATING THE ONTOLOGY GRAPH

The knowledge graph is supported by a background ontology graph that provides a wider data model but also includes relationships and entity types that might not be represented in the data. This ontology graph, contrary to the knowledge graph, is independent of the data layer and references only the ontologies and relationships between ontologies and their resources. This ontology graph, therefore, allows for broader searches that consider not only the entity type of a certain entity but also the neighbours and relationships of that entity type. The graph can then be exploited to compute paths between nodes and find more relationships between concepts.

We obtain this graph by automatically extracting and retrieving from the Web the ontologies used by each source RDF dataset. If the input datasets to be modelled are provided in RDF format, then its ontologies are also retrieved to be added to the ontology graph. First, we extract all namespaces referenced in the data by adapting the `computeq_name` function from the Python rdflib package[5]. This function takes a URI and decomposes it into its parts. Using the fragment of the name that points to the name space, we automatically download the ontologies by crawling the dereferenced URI to find the download URL in an acceptable RDF format. Similarly to the knowledge graph building step, we convert these ontologies to JSON using the Raptor RDF Syntax Library and store and index them in the document store.

Then, we construct the first layer of the ontology graph by considering the relationships in and between the ontology classes and properties. A weighted directed ontology multigraph G is defined as $G = (V, E, W)$, where V is the set of vertices, E is the multiset of edges that connect the vertices, and W is the weighting function for the edges. The vertices represent ontology classes or properties that are connected to at least one edge. Edges represent relationships between ontology classes or properties. Edges created from `owl:equivalentClass` or `owl:equivalentProperty` predicates are added in both directions between the pair of vertices. Similarly, for the predicates `rdfs:subClassOf` and `rdfs:subPropertyOf`, we create the inverse edges `superClassOf` and `superPropertyOf`, respectively, to allow graph traversal through the children vertices. These edges are assigned a weight of 1.

The second layer of edges is obtained with an enrichment step that creates new relationships between ontology classes using: (1) co-occurring entity types, (2) `owl:sameAs` links, (3) ontology matching, and (4) extended string matching. The *co-occurring entity type* edges are obtained by extracting entities that have more than one type assigned in the knowledge graph. The relationship between the two entity types is usually equivalence or subsumption and, therefore, an edge is added between the two ontology classes with an assigned weight of 1. Next, we extract new relationships from *`owl:sameAs` links*.

[5] https://github.com/RDFLib/rdflib

Therefore, we extract the types of both entities connected via `owl:sameAs` and create a new equivalence edge between them with a weight of 1.

The final enrichment step adds edges from string matching techniques. First, for RDF datasets, we use *ontology matching techniques* to discover new relationships between ontology classes. In this work, we use the AML [69] ontology matching system since it is consistently one of the best-performing systems in the OAEI [132]. We adapted AML's workflow to efficiently process bulk matching requests from the pairwise combinations of ontologies. We use the AML WordNet Matcher to extend the lexicon of the labels in the ontologies and use the Word Matcher to obtain the mappings. This matcher uses a bag-of-words strategy to find word overlaps between two ontology class labels and scores these mappings with a modified Jaccard similarity. The mapping score $\mathtt{score}_m$ ranges between 0 and 1, where 1 represents the highest similarity between two mapped classes. A threshold can be set to exclude mappings below a certain score. The weight of a new edge e is calculated with the weighting function $W(e)$ as $W(e) = 1 + (10 - (10 \cdot \mathtt{score}_m))$. We apply a linear transformation to the score so that the edge weight ranges from 1 to 11 and lower mapping scores have significantly heavier edge weights than higher scores, making them harder to traverse during graph searches. For example, an edge of weight 1.0 (perfect match) has an edge weight of 1, while a mapping which achieves only a score of 0.5, will have an edge weight of 6. Due to the bag-of-words strategy, a mapping with a similarity of 0.5 is still likely to be between related concepts with overlapping words, however, it is less likely to be between equivalent concepts. Therefore, when transversing the ontology graph to find the most interoperable candidates, the algorithm should prefer closer concepts, which the edge weighting scheme takes into account when traversing algorithms are used.

In addition to ontology matching, we perform *extended string matching* between data sources and ontologies. In this step, we match ontologies extracted from the RDF data sources with the data model inferred from the semi-structured input datasets. From RDF input data, we extract entity type and property labels. For datasets that do not have these resources explicitly described, we use attributes or table headers as input for this matching task. Using the labels previously obtained, we search them in this ontology index and retrieve the top-10 matches. We score the matches using the string similarity score described in Chapter 4, Section 4.4.1. We add these matches as new edges and weigh them using the same linear transformation used for ontology matching edges.

3.8 ONTOLOGY GRAPH ENRICHMENT EXPERIMENTS

This section presents the results of the evaluation of the ontology matching strategy and an experimental analysis of the impact of the string matching (ontology matching and extended) enrichment edges on the ontology graph. We also present the descriptive statistics of the ontology graphs of the use-cases.

3.8.1 Matching Evaluation

We perform two edge enrichment matching tasks: ontology matching and extended string matching. We evaluate the ontology matching strategy with common state-of-the-art methods and evaluate string matching using OAEI reference ontologies as described below.

Ontology Matching Evaluation

The ontology mappings are evaluated in terms of accuracy using precision and recall metrics. We evaluate the mappings against varying confidence thresholds to understand how this parameter affects the results. We performed a baseline evaluation of the chosen ontology matching strategy by selecting a subset of two (cmt and conference datasets) of the manually curated reference alignments of the OAEI Conference track [133]. However, since this reference alignment evaluates only equivalent mappings, we added the COMPOSE reference [134], where the author extended some of the conference reference mappings to also include subsumption correspondences.

Figure 3.2 shows the precision and recall results of this evaluation. We verify that, as expected, the total number of mappings decreases with the increase of the confidence threshold, and the precision increases. Contrary to the expected result, recall is also relatively low, even when the threshold is 0. This is an unexpected result since AML is one of the best performing systems in the OAEI in the conference track of the competition. This performance can be explained by considering that our edge enrichment strategy used only a single matcher, the Word Matcher, as opposed to the combination of matchers that AML uses for the OAEI competition. The main methodology of the chosen matcher is a bag-of-words match, which in the case of this dataset does not obtain a good performance since there is a high variance between equivalent terms in the mappings. For example, `http://cmt#SubjectArea` is equivalent to `http://confOf#Topic`. The Word Matcher is not designed to pick up on these matches, therefore, the recall for this dataset is slow. Nonetheless, we decided that, for the case of our framework, this is a simple matcher that will efficiently obtain relevant edges since matches will need to have at least one overlapping word, increasing the chance that the concepts are related. The additional matchers of AML add to the complexity of the matching algorithm and, in our case, we had approximately 40 ontologies in each use-case, with the biomedical use-case having large ontologies that make the matching a complex task. Therefore, we opted for a less complex solution which ended up resulting in a lower performance for the AML matching system. Furthermore, since naturally recall decreases with the increase of the threshold and our goal is to support the generation of extensive lists of candidates, in our case, it is preferable to maximise recall instead of precision. Since we use the confidence value of mappings to put a higher weight to weaker matches in the ontology graph, making them harder to traverse during path computations between vertices, the impact of incorrect mappings should be reduced.

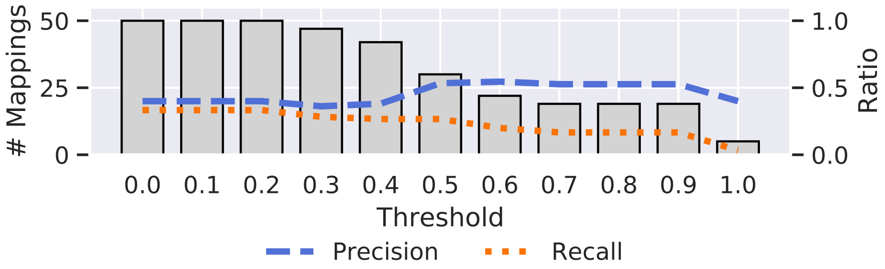


Figure 3.2: Ontology matching evaluation

This evaluation considers only equivalence and subsumption relationships. However, other correspondences exist between ontology classes, such as between `bibframe:Status` and `bibo:DocumentStatus`, which are not equivalent or subsumed but are conceptually related. Therefore, in our experiments, we kept every mapping by setting the confidence threshold to 0 and relied on the edge weights to reduce the influence of incorrect mappings without discarding possible relatedness correspondences.

Extended String Matching Evaluation

The extended string matching was evaluated using the full OAEI Conference track data from the 2019 edition[6]. Considering the consistent format of URIs in this dataset, for this evaluation, we used the last fragment of each URI (i.e., the remaining of the URI after the # symbol) as label and searched for each label in all other conference ontologies. We then applied the extended similarity methods over this set of labels and ontologies.

Following these methods, we obtained 1336 class mappings with 129 missing from the reference produced in the previous section, and 144 property mappings with 77 missing from the reference. Figure 3.3a shows the number of mappings, the precision, and the recall for class mappings and Figure 3.3b shows the same for property mappings. This extended string matching technique performs well when compared with the adapted AML ontology matching strategy but performs better with class names than property names in this dataset. In the case of the property matching, precision and recall are affected by the high percentage of `rdfs:label` values which are in dromedary case and are not matched against equivalent properties in other ontologies that are stylised with spaces in the label. Furthermore, through manual inspection, we found several cases of false negatives, e.g., in the reference, `http://confOf#hasTopic` is equivalent to `http://conference#has_a_track-workshop-tutorial_topic` but the extended string

[6] `http://oaei.ontologymatching.org/2019/conference` (Accessed in September 2020)

matching only finds the match to `http://edas#hasTopic`, which is arguebly correct. Finally, the last contributor to the low performance with properties are cases similar to the ones that degraded the performance of the ontology matching system, i.e., words that are synonym but have no overlapping terms. Therefore, for both classes and properties, we keep the threshold at 0 and weight the edges added with a linear transformation to increase the weight of mappings that are less likely to be correct.

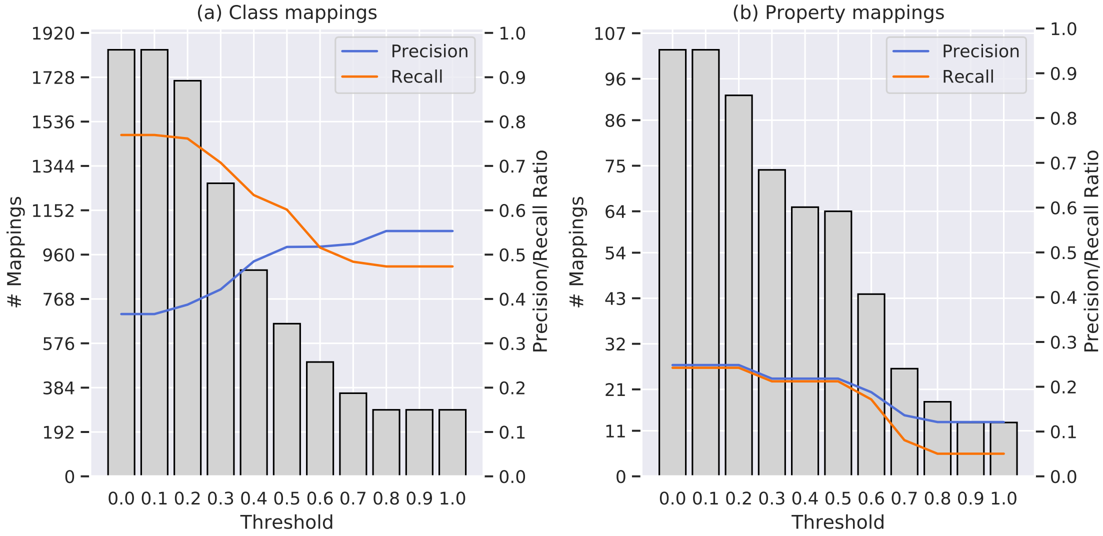


Figure 3.3: Extended matching evaluation

3.8.2 Ontology Graph Enrichment Analysis

We present an extended analysis of the impact of the selected ontology matching technique. We focus on this particular step because ontology matching is a complex process that can be expensive to compute and, therefore, it is important to understand its value in the process of creating a tightly connected ontology graph.

In this section, we present experiments that assess the impact of the ontology mappings on the overall connectivity of the graph by performing an in-depth analysis of the characteristics of the ontology graph following the ontology mapping enrichment. We excluded properties from this analysis to simplify the process.

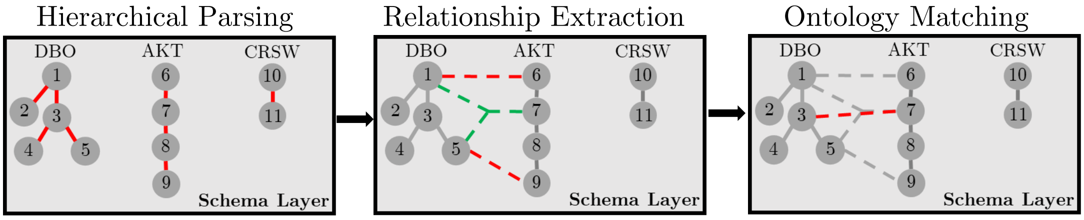
Our evaluation approach was divided in three stages, illustrated in Figure 3.4, which are:

1. *Hierarchy Parsing*, where the hierarchical relationships, i.e., `owl:subClassOf`, are extracted. The graph resulting from this step is named Base Graph (BGraph).

2. *Relationship Extraction*, where equivalence relationships are extracted. Not only `owl:equivalentClass` but also logical definitions were considered. Logical definitions [135] are complex axioms that refer to the relationship between a defin-

ing class x and a general class g that is discriminated from other classes of x with a class d. For example, the Cell Ontology (CL) [136] contains a logical definition for the class *cardiac neuron* (`obo:CL_0010022`), where g is a *neuron* (`obo:CL_0000540`) that is *part of* d, the *heart* (`obo:UBERON_0000948`). Logical definitions are translated into two edges: $E_1 = (x, g)$ and $E_2 = (x, d)$. Figure 3.4 illustrates `owl:equivalentClass` axioms, in red, connecting DBPedia Ontology (DBO) *Organization* with AKT Reference Ontology (AKT) equivalent class. Similarly, the axioms connect *University* between the two ontologies. The diagram also shows an hypothetical logical definition (green line) to connect the intersection between DBO's *Organization* and *University* with AKT's *Higher Education Organization*. This graph is an increment over the *Base Graph* and was called Axiom Graph (AGraph).

3. *Ontology Matching*, where mappings from ontology matching techniques are added. For these experiments, we empirically chose a threshold of 0.4. Contrary to the ontology mappings added to the ontology graph of the main framework, for efficiency reasons, here we do not use edge weights instead with simplify the methodology by cutting off mappings with a score lower than 0.4. If conflicting mappings were found, only the highest scoring ones were selected for the final alignment. Figure 3.4 illustrates a possible mapping (red line) between DBO's *Educational Institution* with AKT's *Educational Organization*. This graph is an increment over the *Axiom Graph* and was named Mappings Graph (MGraph).



Classes
1. Organization
2. Company
3. Educational Institution
4. School
5. University
6. Organization
7. Educational Organization
8. Higher Education Organization
9. University
10. Abstract Information
11. Course

Figure 3.4: Ontology graph enriching stages

Characterisation and Evaluation

To the best of our knowledge, no benchmark or comparable approach exists to evaluate the edge enrichment of a multi-source ontology graph. However, the graphs obtained from the hierarchy parsing, axiom extraction, and ontology matching phases (BGraph, AGraph, and MGraph, respectively) are increments of each other, i.e., the AGraph is built from the BGraph and the MGraph is enriched from the AGraph. This incremental building process allows us to assess how the more complex ontology axioms and the ontology matching process affect the base structure of the ontology graph.

We adopt the BGraph as the baseline for the evaluation since it represents the minimum set of existing edges between the considered ontologies. We then evaluate each step as an increment over the previous graph with characterisation measures that provide an overview of the structure. This evaluation strategy assesses the impact of each step, giving a better understanding of the graph's evolution. Ideally, the results of this evaluation present graphs that are increasingly more cohesive and connected, translating into graphs that can relate concepts more efficiently and effectively. The metrics used are defined as follows:

- **Connected Components (CC)** refers to sub-graphs where every node has a path to all other nodes in the same sub-graph.
- **Clustering Coefficient (CCF) [137]** measures the degree at which the nodes in the graph cluster together, based on triads.
- **Average Node Degree (AD)** is the average number of edges that are connected to a node.
- **Cohesiveness (CV) [138]** measures how difficult it is to split the graph.
- **Distribution of Shortest paths** measures the frequency of the lengths of the paths between two nodes in the graph such that the number of edges in these paths is minimised.

The number of CC and the AD are directly affected by the addition/removal of edges. In the ontology graph, these metrics evaluate the overall connectedness of the graph. Unchanging values between stages of building the ontology graph mean that the number of edges added had a low impact on the graph structure. CCF and CV have a range of $[0, 1]$ and measure how closely connected the nodes are in the graph and how robust those connections are. Ideally, each step of the building process should increase the value of these measures, therefore producing an increasingly connected graph where related concepts are more clustered and harder to separate into isolated structures.

The distribution of shortest paths evaluates the impact of the edges added in each step in terms of efficiency when querying for similar or related concepts. An increase in the number of shortest paths demonstrates that new edges are creating new paths between the concepts in the graph.

Results

We performed experiments over ontologies of diverse domains found in the LOV [81] portal. Ontologies featured in this portal are assigned subject tags. The three most common tags are "Methods", "Metadata", and "Geography". From the 659 ontologies available in the LOV in February 2019, we were able to download and parse 340. We refer to the graphs obtained from this set as the General Ontology Graph (GOG).

We also used a domain-specific ontology set extracted from the OLS [84], a repository of biomedical ontologies. Out of 220 ontologies available through the OLS REST API in February 2019, we were able to download and parse 206. The most common reason for discarding an ontology was the presence of outdated information, e.g., `owl:import` statements referring to ontologies that do not exist anymore. We refer to the graphs obtained from this set of ontologies as the Biomedical Ontology Graph (BOG).

These two test sets provide a rich environment for testing with ontologies with different characteristics. Biomedical ontologies are domain-specific ontologies and, therefore, are more likely to have equivalent or related terms between them. These ontologies are also characterised by their well-defined standards [139], large size, and complex axioms. Ontologies found in the LOV have diverse domains and, therefore, the alignment of the topics is not guaranteed. These ontologies have different levels of formalism and mostly follow LOD guidelines.

Table 3.10: Results of the ontology matching process. O is the set of ontologies, A is an alignment resulting from ontology matching, $med(M)$ is the median number of mappings in an alignment, and $avg(score)$ is the average of the confidence scores of the mappings

Ontology Dataset	$\|O\|$	$\|A\|$	$med(\|M\|)$	$avg(score)$
General ontologies	314	33 619	2	0.66
Biomedical ontologies	181	14 292	15	0.53

Table 3.10 shows the results of the pairwise ontology matching over the general and the biomedical sets of ontologies. The general set of ontologies obtained more alignments than the biomedical set. However, this difference is due to the difference in the number of ontologies in each set since the median number of mappings in the general set was significantly lower than the biomedical set. These results show that, as expected, ontologies in a more restricted domain will find a more significant number of overlapping or related concepts. The average mapping score in the general set of ontologies was 66%, while the biomedical set obtained 53%.

Table 3.11 compares the ontology graphs in different stages of construction. In GOG and BOG, the number of nodes shows a small increase from the BGraph to the AGraph due to axioms that reference ontology classes outside of the scope of the ontology set. In these cases, the new ontology classes are added as new nodes to the AGraph.

The GOG shows an increase of ≈21% in the number of edges between the BGraph and the AGraph and ≈56% more edges in the MGraph than in the AGraph. In the BGraph, ≈48% of the nodes are in the Largest Connected Component (LCC), but this number increases to ≈62% in the AGraph and ≈88% in the MGraph. Most of the elements not connected to the LCC are isolated, forming a new connected component with a single node and no incoming or outgoing edges. The most common reason for discon-

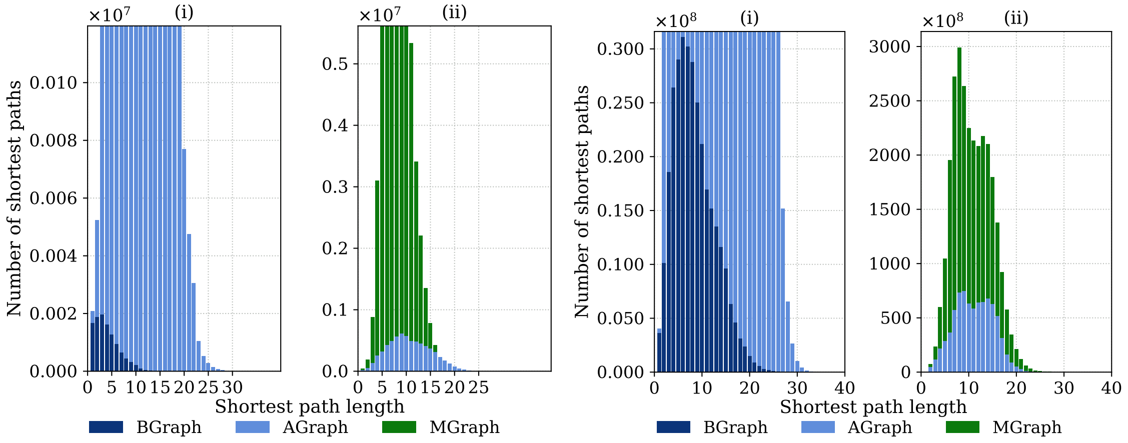
Table 3.11: Characterisation of the GOG and BOG. V is the set of nodes and E is the set of edges. LCC - Largest Connected Component; CC - Connected Components; 1-CC - single node CC; CCF - Clustering Coefficient; AD - Average Node Degree; CV - Cohesiveness

Graph	Stage	|V|	|E|	|LCC|	|CC|	|1-CC|	CCF	AD	CV
GOG	BGraph	16 510	16 755	7889	3210	2867	0.022	2.450	0.433
	AGraph	16 580	21 141	10 307	2252	2071	0.065	2.757	0.576
	MGraph	16 580	47 727	14 634	1571	1553	0.177	5.639	0.886
BOG	BGraph	4 502 980	6 967 165	4 223 817	66 270	65 392	0.049	3.089	0.562
	AGraph	4 502 981	7 456 127	4 227 375	63 522	62 693	0.050	3.160	0.575
	MGraph	4 502 981	14 595 189	4 458 547	41 317	40 581	0.129	5.978	0.833

nected nodes in the GOG is inconsistencies in the definition of classes, properties, and their relations. For example, the class `http://purl.obolibrary.org/obo/HAO_0002311` is a root class with no descendants and, therefore, is isolated in the graph.

Overall, in the BOG, the BGraph and the AGraph are structurally similar since only a small number of edges were added (≈7% more edges). However, between the AGraph and the MGraph, the differences are more prominent due to an ≈48.9% increase in the number of edges. The BGraph and AGraph have ≈94% of the nodes in the LCC. In the MGraph, ≈99% of the nodes are connected in the LCC. Most of the single elements connected components are due to ontologies that do not follow common standards and formalisms to build an ontology or reuse classes. For example, the ontology *Flora Phenotype Ontology*[7] was created only with classes and contains no intra- or inter-links between classes.

Figure 3.5a and Figure 3.5b show the distribution of shortest paths for the ontology graphs. Due to the large size of the BOG, the distance histogram was computed for a sample of 60% of the total number of nodes of this graph.



(a) GOG

(b) BOG

Figure 3.5: Distribution of shortest paths

[7] `http://purl.obolibrary.org/obo/flopo.owl`

In both figures, Plot (i) corresponds to the overlap of the distance histogram of the BGraph with the AGraph, and Plot (ii) corresponds to the overlap of the AGraph and the MGraph. Overall, both axiom and mapping edges have a significant impact in the number of shortest paths that connect the nodes, which is directly related with the increase of the number of elements connected in the LCC, i.e., more nodes connected, new shortest paths created. Despite the lower impact of the edges added to the AGraph in terms of structural properties in comparison to the MGraph, the distribution of shortest paths shows a comparable increase.

These new paths facilitate the discovery of relationships between concepts in the graph, but also help shorten the distances between them. More concretely, the changes in the distribution of shortest paths can be illustrated, for example, by the changes in the shortest path between the node with the label *cancer* (DOID:162) and *malignant cell* (`CL:0001064`). In the BGraph, the shortest path between the two nodes is 10, in the AGraph it is 4, and in the MGraph the concepts nodes are directly connected, i.e., a shortest path length of 1.

In summary, the characterisation of the proposed ontology graph demonstrates the contribution of each of the building steps to the structure of the graph. The addition of axioms and mappings improves the connectedness of the graph and creates new paths between nodes. The ontology matching step proved to be significant, which shows that, in the context of building a foundation for a knowledge graph, ontology matching is an approach that can connect previously disconnected parts of the graph facilitating the process of finding related concepts. Furthermore, the significance of this step is reinforced in Chapter 4, where ontology matching was used to enrich the ontology graph of the use-cases and facilitate the process of obtaining candidates based on their interoperability. The measure of interoperability would be hindered if it considered only axioms originally in the ontologies. For example, most of the ontologies in the library use-case provide no direct equivalence links between concepts. Having edges provided by the ontology matching facilitates content scoring measures based on graph distances, such as distance to the input entity type or property (see Section 4.4.1: Distance to Source Resource). In terms of interoperability, the ontology matching edges provide a more accurate view of potential interoperability of the concepts, given that without these edges, interoperability would be more limited to the relationships contained in each individual ontology and, therefore, the most interoperable concepts would only be more severely weighted by frequent concepts within the same library. These concepts and consequences will be furthered discussed in Chapter 4. Overall, the new edges obtained through ontology matching allow concepts in the graph to be more easily reachable from each other, facilitating the discovery of new relationships that would otherwise be missing in the ontologies.

3.8.3 Library Use-case Descriptive Statistics

The library ontology graph includes 36 ontologies, described in Table A.1. More details on these ontologies are found in Appendix A. Table 3.12 shows descriptive statistics for the library use-case ontology graph before and after enrichment steps. The same conclusions as in the previous edge enrichment analysis can be observed in this table, where, after enrichment, the graph becomes more tightly connected and it is more difficult to separate its components.

Table 3.12: Characterisation of library ontology graph before and after enrichment

Graph	|V|	|E|	|LCC|	|CC|	|1-CC|	CCF	AD	CV
Before	6621	22 058	1779	3071	2745	0.052	6.663	0.968
After	7972	53 381	5191	2080	1947	0.164	13.392	0.988

3.8.4 Life Sciences Use-case Descriptive Statistics

The life sciences ontology graph includes 33 ontologies, described in Table A.2. Table 3.13 shows descriptive statistics for the life sciences use-case ontology graph before and after enrichment steps. Due to the size of the ontologies, we did not add the ontology matching enrichment in this use-case. Once again, the same observations are verified in the ontology graph of this use-case.

Table 3.13: Characterisation of life sciences ontology graph before and after enrichment

Graph	|V|	|E|	|LCC|	|CC|	|1-CC|	CCF	AD	CV
Before	2 554 233	8 077 856	2 483 566	42 674	42 400	0.000	6.325	0.996
After	2 554 310	8 106 591	2 511 306	40 501	40 254	0.000	6.347	0.997

3.9 DATATYPE PROPERTY CLASSIFICATION MODEL

Datatype properties are found in ontologies and refer to properties that have literal values in their range, e.g., is common for dates in datasets to be associated to an entity via a datatype property. In the machine learning field, classification is a supervised learning task, where a model is trained to identify the categories for input data. Similarly to Z. Chen et al. [124], we treat the datatype property generation as a multiclass classification task, where each datatype property in the knowledge graph is a class and its value is processed into a feature vector to use as training data. We use a random forest classifier, which is an ensemble learning method for classification that works by training several decision trees and averaging the predictions to produce a final model. This method helps in correcting the tendency of decision trees to overfit their training set. We use the scikit-learn implementation[8] of the random forest classifier.

Table 3.14: Datatype property examples in the British Library dataset

URI	Property	Value

`bnb:resource/011326467`	`schema:datePublished`
	`bibo:isbn10`
	`schema:name`
	`isbd:P1053`
		1990
		0356190684
		The Eye of the world
		xiv,670p.

`bnb:person/JordanRobert1948-2007`	`foaf:name`
	`schema:deathDate`
	`foaf:givenName`
	`foaf:familyName`
		Robert Jordan
		2007
		Robert
		Jordan

We train models for each individual RDF source in the knowledge graph and for the knowledge graph as a whole. The single RDF data source model will have the tendency to overfit since their data model is more homogeneous and, therefore, might have difficulties with data that is similar but not the same. However, if a dataset is provided that uses the same format for the datatype property values, these individual models excel over the complete knowledge graph model. In slightly different datatype property values, the complete knowledge graph model will likely perform better since it provides a broader view of values for datatype properties that are used by multiple RDF data sources. Table 3.14 shows an example of the variety of datatype properties found in the British Library dataset.

The model training and test methodology used in our approach (1) defines features to extract from the datatype property values, (2) performs hyperparameter tuning with randomised and exhaustive parameter optimisation, (3) finds an optimal document sample size from the knowledge graph to train the model, (4) fits the models for each RDF source in the knowledge graph and the whole knowledge graph.

3.9.1 Feature Selection

Our approach to datatype property prediction assumes that the values of datatype properties are good predictors of the datatype property of a new instance. We performed an empirical analysis of the data in the library and biomedical use-cases to better understand the kind of features that had the potential to more accurately distinguish between datatypes properties. We concluded that the classification model should be able to clearly distinguish between textual and numerical datatype properties. For example, in the library dataset it is common to have numerical codes to uniquely identify books and,

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

at the same time, textual properties describe attributes such as title and author. However, the classification model not only has to distinguish between generic datatypes of properties but, among the same datatype, has to be able to discern more subtle differences between property subtypes. For example, the model has to be able to distinguish between an author's name and a title. In this example, the distinction is not trivial, since it is not uncommon for book titles to overlap with people's names. Nonetheless, in our scoring methodology, frequency of a candidate datatype property is taken into account to minimise overlapping cases, such as the title and author name. To capture these characteristics, we defined the following features:

1. Total number of characters in the literal
2. Number of digits
3. Number of letters
4. Number of white spaces
5. Number of other characters
6. Number of uppercase letters
7. Is date? (yes or no)

Features 1 through 6 are baseline distinctions between literals since they differentiate by the size and type of characters featured in the literal. These follow the intuition that different types of datatype property values exhibit different patterns in terms of their alphanumerical arrangement. Features 1 captures the intuition that the values of the same property will have similar lengths, e.g., date *vs.* International Standard Book Number (ISBN). Features 2 and 3 help distinguish between numerical and string values. Features 4 through 6 are more focused on string-based values and help distinguish between string literals that form sentences, e.g., a book title *vs.* an author name, which will usually have only 1-2 white spaces. Furthermore, a person's name is likely to have no non-alphanumeric characters, while a book title might include some punctuation such as a colon.

Several NER systems provide functionalities to identify dates in a corpus. However, these models are designed to provide optimal results when a context is provided. Therefore, we created Feature 7, which is a naive approach to discern if a literal is a date or not. First, we use the dateutil[9] Python package to parse the literal. If the literal is recognised as a date, the following rules are checked:

1. Is Feature 3 > 0?
2. Is the value decimal?

[9] `https://dateutil.readthedocs.io`

3. Is it a negative number? (i.e., starts with a minus sign)

4. If there are not letters or spaces, does it have more than 3 numbers?

If any of the answers is positive, then the value is not considered a date, otherwise it is a date.

Is date Feature

We evaluated the *Is date* feature by manually creating a ground truth with all the datatype properties in the knowledge graph that have date datatypes. Then we obtained a random sample of 100 documents from each dataset in the library knowledge graph, and extracted all the pairs of datatype property values in each document. The value was parsed by the *Is date* feature methodology described earlier in this chapter and compared with the ground truth to evaluate the precision and recall.

Table 3.15 compares our approach (dateutil + rules) with the NER modules of SpaCy[10] and Facebook's Duckling[11]. Our approach achieves significantly higher performance when identifying dates in our use-case. The NER modules tested falter in this case because they are trained in full-text corpora where dates are used in the context of a sentence. RDF property ranges do not include any context for the date, which leads to several false positives and negatives.

Table 3.15: Evaluation of date extraction approach

Approach	**Precision**	**Recall**
SpaCy	0.52	0.85
Duckling	0.44	0.16
dateutil + rules	0.96	0.92

In our approach, false positives are mostly caused by the `bibo:issue` property. Some documents mistakenly have a date as the issue number, despite the property not being considered a date property. When this property is added to the date properties, the precision increases to 0.99.

3.9.2 Hyperparameter Optimisation

The scikit-learn implementation of the random forest model provides several tuning parameters. We perform two hyperparameter optimisations: randomised and exhaustive. Through empirical observation of the data we found that some datatype properties were poorly represented in the data, having significantly less overall representation than the average property. Through experimentation, we also found that, due to the size of

[10] `https://spacy.io`
[11] `https://github.com/facebook/duckling`

the data, the hyperparameter optimisation was a slow task, therefore, together with the empirical observation of the data we used a 3-fold cross-validation approach to maximise P@5. For each of the optimisations, we extract 100 random value samples of each datatype property in the knowledge graph.

We tune the following hyperparameters:

- `n_estimators`: number of trees in the forest
- `criterion`: measures the quality of each split using either Gini impurity [140] (gini) or information gain [141] (entropy)
- `min_samples_split`: number of samples in node necessary to split it, given as a fraction of the total number of samples in the training set.
- `min_samples_leaf`: number of samples required to be a leaf node, given as a fraction of the total number of samples in the training set.
- `max_features`: number of features to consider for the best split
- `min_impurity_decrease`: a split node has to decrease the impurity of the parent node by this amount, given as a fraction of the total number of samples in the training set.
- `bootstrap`: whether to use a sample or the whole dataset when building trees.
- `max_samples`: provides the fraction of samples to draw from the training data for each Decision Tree estimator, if `bootstrap` is True.

We run the randomised grid search with a wide selection of values to discern which hyperparameters affect the model performance the most. In this grid search, we provide the number of iterations n and, instead of exhausting the possible combinations of hyperparameter values, the search is limited to n number of combinations randomly selected. We repeat the random grid search three times and empirically choose a reduced set of hyperparameters to pass to the exhaustive grid search. In our experiments, we perform 3-fold cross-validation with $n = 100$.

The exhaustive grid search obtains the final hyperparameters by fitting the model with all combinations of values for the supplied parameters and choosing the ones that maximise P@5.

Results

Table 3.16 presents the three hyperparameter tuning steps. We start with a broad range of hyperparameters in the randomised grid search. The results reported in the "Exhaustive" column represent the empirically chosen agreement between the 3 repetitions of the random search. The Final column includes the hyperparameters that maximised P@5 in the exhaustive grid search and were used to fit the random forest models.

Table 3.16: Hyperparameter search

Hyperparameter	Randomised	Exhaustive	Final
`n_estimators`	$[100, 1000]$	$[100, 500]$	300
`criterion`	{gini, entropy}	{entropy}	entropy
`min_samples_split`	$[0.01, 0.40]$ step $= 0.01$	$[0.01, 0.21]$ step $= 0.05$	0.01
`min_samples_leaf`	$[0.01, 0.40]$ step $= 0.01$	$[0.01, 0.10]$ step $= 0.02$	0.01
`max_features`	$[2, 7]$	$[2, 4, 5, 7]$	4
`min_impurity_decrease`	$[0.0, 0.2]$ step $= 0.01$	$[0.0, 0.1]$	0.0
`bootstrap`	$[\mathsf{True}, \mathsf{False}]$	$[\mathsf{False}]$	False
`max_samples`	$[0.7, 0.9]$ step $= 0.1$	N/A	N/A

In all runs, the entropy criterion was chosen over Gini and lower samples split and leaf were preferred over higher values. The `max_features` parameter was the one that oscillated the most, but, after the exhaustive search, 4 was the value that maximised P@5. The `bootstrap` parameter was always preferred to be *false*, therefore, for the Exhaustive and Final fittings, since `max_samples` depends on `bootstrap` being *true*, this parameter was removed.

3.9.3 Model Fitting

Finally, we fit the random forest model to each RDF data source in the knowledge graph and to the knowledge graph as a whole. We fit each model using parameters extracted from the hyperparameter optimisation step, using a sample of size s with 40% of the data being separated in the test set. We evaluate the model with Precision@1, Precision@3, and Precision@5. Since the goal is to obtain an exhaustive list of candidates, Precision@5 is favoured over the other measures.

Due to the large size of the datasets in the knowledge graph, we selected a sample of size s of values of each datatype property in the knowledge graph. We evaluated the impact of the sample size on the P@5 metric. For that, we generated random document samples of sizes 250 to 5000 in increments of 250.

Figure 3.6 shows that the sample size does not have a significant impact on the P@5 performance, which indicates that the features extracted from the datatype property values are relatively stable and even small sample sizes achieve good performance. On the other hand, the time it takes to fit a model increases linearly with the sample size. Therefore, since sample size does not have a significant impact on performance, we kept the sample size value low and chose $s = 2000$.

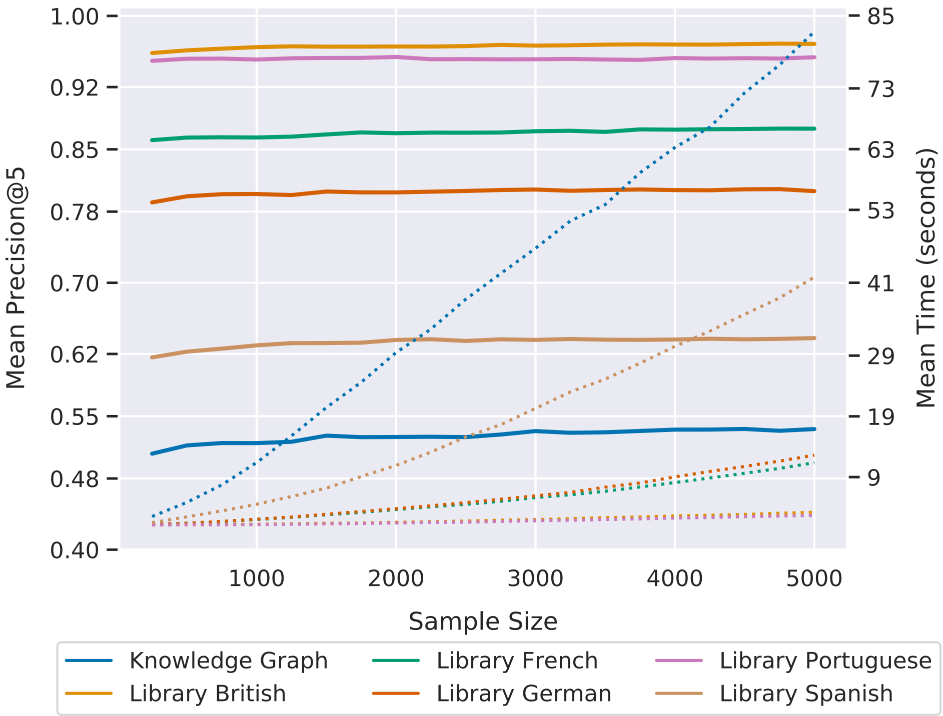


Figure 3.6: Impact of sample size on model fitting. The dotted lines indicate the time it takes for the model to be trained with each sample size

Library Use-case

Figure 3.7 shows the feature importance determined by each of the trained models in the library use-case. Feature importance is a measure used with decision trees which gives an indication of the contribution of each feature to reduce the entropy of each branch of the tree. Overall, the "Is date" feature is given less importance by every model with different features ranging in importance in different models. This outcome is most likely because the remaining features are already distinguishing between dates and other strings and, therefore, this feature is not strictly necessary. The most important features seem to be focused on the alphanumeric composition of the datatype property values, i.e., string length, number of digits, and number of letters. The uppercase feature is given high importance by most libraries, except for the French one.

Despite the different levels of significance given by the different datasets, we used all features for every dataset. Even though the added feature *Is date* did not performing well in this particular use-case, it performs well overall when identifying dates. Therefore, we consider important to report its usage and keep it for possible future use-cases.

Table 3.17 shows the precision results over the test set for each RDF source and the whole knowledge graph in the library use-case. Overall, the models obtained a high P@5 with low P@1, with the more diverse knowledge graph obtaining lower results than any model individually. The Spanish library obtained a considerably lower performance than the remaining individual libraries because, besides a higher than average

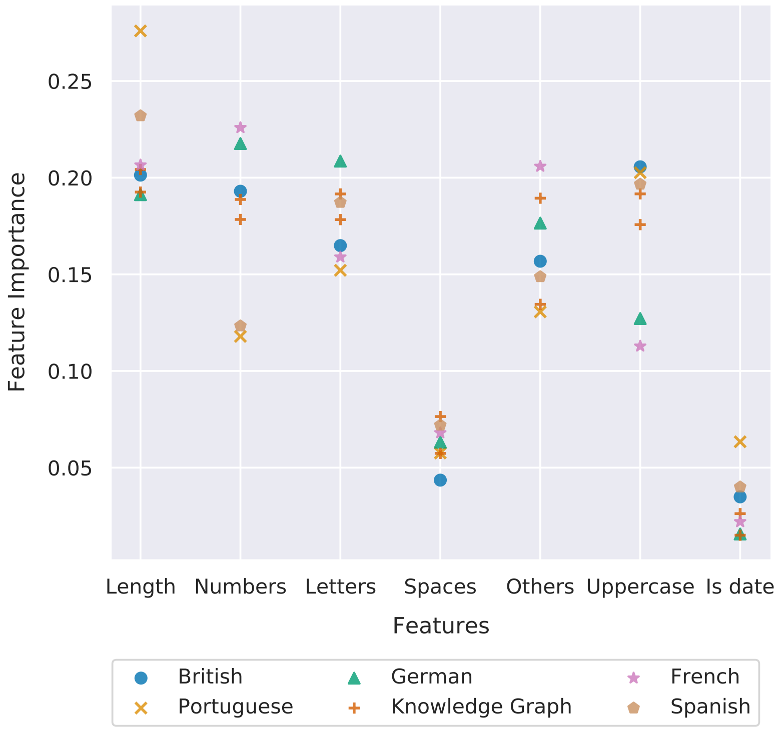


Figure 3.7: Distribution of feature importance in the library datasets

total number of properties, it also includes several properties with similar values, making it harder for a random forest model to predict the properties correctly. For example, for dates there are at least 9 datatype properties in the BNE ontology and it includes others from external ontologies such as `http://rdaregistry.info/Elements/a/P50038`. The same situation is verified in other properties, where values between properties are similar, making it hard for the model to distinguish between datatype properties. Nonetheless, since the aim of the random forest model is to produce an exhaustive list of datatype property candidates, we give preference to P@5. The high performance of the individual models is likely connected to overfitting. However, considering that the datatype properties to be extracted from these models are expected to be in the same domain, following the same structure and semantics, we do not necessarily consider overfitting an issue. The overfitted models are desired in the case of a dataset with data very similar to any of the RDF sources individually. Nonetheless, the whole knowledge graph model is expected to suffer less from overfitting, providing a broader view of the domain to match cases that might not exactly follow any particular data model included in the knowledge graph.

Overall, we consider that the performance is adequate for the task at hand since exact matches are not the goal. The trained model will allow the framework to retrieve the datatype properties in each model in ranked order as scored by the random forest model.

Table 3.17: Random forest model evaluation for the library use-case

RDF Source	P@1	P@3	P@5
Library British	0.57	0.87	0.97
Library French	0.47	0.77	0.86
Library German	0.43	0.70	0.80
Library Portuguese	0.71	0.9	0.95
Library Spanish	0.28	0.52	0.63
Knowledge Graph	0.21	0.41	0.52

Life Sciences Use-case

In the life sciences RDF data sources, most of the datatype properties are URLs linking to external resources or identifier codes. For example, in the Monarch dataset, the property `obo:RO_0002200` (has phenotype) has the value `http://omim.org/entry/612083`, which is an external resource not found in the knowledge graph. Therefore, since this value is not an entity in the knowledge graph, the property `obo:RO_0002200` is considered a datatype property. Nonetheless, we tested using the same hyperparameter values as before and obtained good results, as seen in Figure 3.8 and Table 3.18.

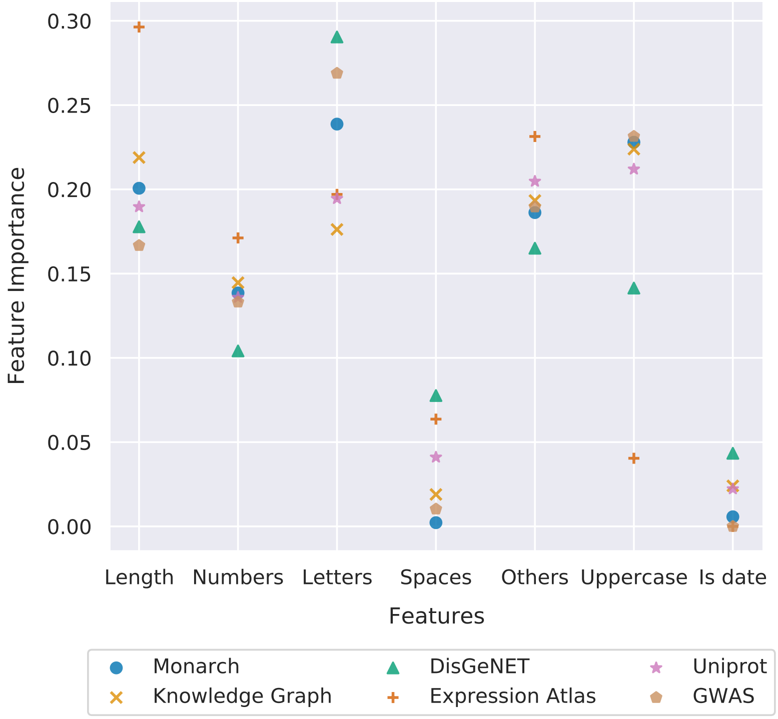


Figure 3.8: Distribution of feature importance in life sciences datasets

In Figure 3.8, we can see that yet again the *Is date* feature is not relevant for this date because dates in the data are rare. The remaining features help distinguish between text labels, identifiers, and URLs. Except for text labels, identifiers and URLs usually do not contain spaces, which can explain the lower importance of the *Spaces feature*.

Table 3.18 shows the precision results for the evaluation of life sciences models. Similar conclusions to the library use-case can be drawn, where the individual models perform well due to overfitting, while the broader models have the lowest performance. In this case, due to the nature of the Monarch dataset, this model also performs worse due to the wider domain it covers and, therefore, with the proposed features, it is not easy to distinguish some of the datatype property values.

Table 3.18: Random forest model evaluation for the life sciences use-case

RDF Source	P@1	P@3	P@5
DisGeNET	0.89	0.99	1.0
EA	0.74	0.98	1.0
GWAS	0.78	1.0	1.0
Monarch	0.49	0.79	0.89
Uniprot	0.66	0.93	0.98
Knowledge Graph	0.48	0.78	0.88

3.10 CONCLUSIONS

In this chapter, we focused on building knowledge structures to support the generation and ranking of data model candidates. This task was motivated by the question:

RQ1 → *What knowledge structures can support the extraction of ontology-based data models?*

We used a search engine that allows full-text search over the literals indexed from the RDF data sources. Therefore, we devised a series of pre-processing tasks to ensure that the triples in the RDF data sources had a format easily exploitable by the framework. These tasks converted the RDF data into JSON and resolved blank nodes to include them in the document to which they belonged. We also chose a document store with a strong search functionality as the storage facility.

RQ1.1 → *Can increasing the connectedness of the ontology graph support the computation of interoperability measures for candidate data models?*

The ontology graph was extracted from the ontologies in the ontology layer of the RDF data sources. This graph was enriched with data inferred from the RDF sources and with ontology matching techniques. Through an extensive analysis of the enrichment approaches, we concluded that the ontology matching enrichment has a high impact in the connectivity of the graph. The ontology graph, after enrichment, becomes more tightly connected (increased LCC and CCF) with vertex neighbourhoods more difficult to separate (increased CV). These conditions are ideal to support the ranking of candidates based on interoperability since this score relies on a tightly connected graph.

Chapter 4 details interoperability in the context of our framework and how the ontology graph connectivity is an integral part is ranking candidates based on this property.

RQ1.2 → *Is it possible to train a classification model to accurately predict datatype properties from literal values using multiple knowledge graphs?*

We extract relevant metadata from the RDF data sources to facilitate the generation and ranking of candidates. The purpose of these structures is to provide metadata support that speeds up the access to essential properties of the data based on frequency. We also compute the patterns that make up the data and that need to be modelled for easy access throughout the framework generation and ranking process.

Finally, we also produce a classification model with training data extracted from the knowledge graph to predict datatype properties when generating this type of resource candidates.

Therefore, returning to the overall question: *What knowledge structures can support the extraction of ontology-based data models?*

In this chapter, we described knowledge structures that will specifically optimise the generation and ranking of candidates using the framework proposed in this thesis. We took a modular approach to producing the background knowledge and, therefore, each solution can be used independently and applied to other tasks that require similar background knowledge. For our specific question of supporting the extraction of ontology-based data models, the background knowledge is composed of a knowledge graph, a tightly connected ontology graph, metadata maps, and a trained random forest model.

Future implementations of the background knowledge building process described in this chapter could be further optimised by selecting a store for the documents that provides both robust full-text indexing and search, but also maintains an optimised graph structure. However, due to the size and characteristics of the knowledge graph, finding a solution that complies with both requirements is not trivial. For the implementation presented in this thesis, we selected ElasticSearch for its full-text search capabilities, but since this system does not implement logical joins, retrieving necessary information, such as the entity type of the value of an object type property, is inefficient. SPARQL engines are optimised to do this type of join operation, so in the future, introducing a SPARQL engine could optimise the building process.

In relation to the ontology graph enrichment step, further edges between entity types could be discovered by exploiting more relationships between ontologies. For example, as mentioned in Section 3.8.2, ontologies can have complex relations such as logical definitions, or even union and disjoints. These relationships are not currently taken into account when creating the ontology graph and defining the relationships between entity types. Further steps can also be taken to make the matching strategies more robust, such as investigating which ontology matching techniques perform better under different use-cases (e.g., matching entity types *vs.* matching properties). Improving the edge enrichment step can improve the confidence in the relationships in the ontology graph, leading to better ranking in the later steps of the framework.

Finally, the classification model to predict datatype properties can also be optimised by introducing a pre-processing step that identifies the datatype of the property and extracts different features based on that information. Numerical values have different characteristics when compared with string values. These characteristics can be exploited to create independent models for different datatypes. Another limitation of this method is that it considers the datatype property independently of the entity it is describing. Introducing this information in the feature vector could improve prediction because, for example, person entities will inherently have different attributes than book entities. Furthermore, in the future, other classification techniques can be explored and compared against the baseline created using the proposed features and the random forest classification models.

4 GENERATING AND RANKING DATA MODEL CANDIDATES

This chapter details the tasks involved in generating and ranking entity type and property candidates to recommend and facilitate the creation of an RDF data model that fits a given input dataset. Using the background knowledge constructed in Chapter 3, we develop methods that generate entity type, datatype property, and object property candidates. The generated candidates are then ranked according to the computed scores that are proposed in this chapter. We propose two scores that focus on optimising different aspects of the individual candidates, and a third one that re-ranks considering the whole candidate data model.

This chapter addresses the following main research question:

RQ2 → *What measures can be used based on the built knowledge structures to select and rank possible data models?*

This question is answered by dividing it into the following sub-questions:

RQ2.1 → *Can the connections within the ontology graph be used to measure the accuracy and interoperability of possible data models, having a significant effect on the ranking of their components?*

RQ2.2 → *Can the connections between components of the proposed data models be used to measure their consistency, with a significant effect on their ranking?*

These questions are answered by first presenting the methods for each one of the generation and ranking steps, and then providing experiments to validate the design and performance of the framework. This stage of the framework has the following tasks: (1) generating entity type, datatype, and object properties (Section 4.3), (2) ranking entity type, datatype, and object properties (Sections 4.4.1 and 4.4.2 and answers RQ2.1), (3) ranking data models (Section 4.4.3 and answers RQ2.2), and, finally, (4) experimenting and discussing the results of the framework.

Therefore, the contributions of this chapter include methodologies to (1) generate entity type, datatype, and object property candidates from a background knowledge graph, (2) rank these candidates using the background knowledge in the knowledge graph and ontology graph, (3) gather the individual candidates into a single ranked and consistent data model.

In the following sections, we will first contextualise and introduce the particular challenges related to the tasks described in this chapter. Then we will present work related to the process of generating and ranking entity types and properties for a data model.

The methods are presented in two sections, separated by generation and ranking. In Section 4.5, we present extensive experiments and evaluations that allow us to assess and discuss the methods proposed. We also present the demonstrator created to illustrate the output of the framework and, finally, we present the conclusions of the chapter, including answers to the research questions.

4.1 INTRODUCTION

The advent of linked data has given data publishers new tools to expose their data on the Web using standards that ensure support for resource linking and knowledge discovery. A survey [14, 15] found that key motivations for publishing linked data are: (1) exposing data to larger audiences on the Web, (2) demonstrate the potential of linked data applications, (3) require data published as linked data to enable its consumption, (4) improving Search Engine Optimisation (SEO) for local resources, and (5) requirement from administrative bodies. The authors also added other reasons, including increased interoperability and linking information across different institutions. However, respondents also noted that the primary barrier when consuming linked data was *matching, disambiguating, and aligning source data and the linked data resources*. Moreover, it is important to note that among the remaining barriers for consuming linked data, respondents mentioned *mapping of vocabulary* and *automation processes to link the data are undeveloped*. Therefore, it appears that, even though publishers are actively trying to improve their integration efforts, users find that when consuming the data, it is hard to integrate different sources to efficiently consume them.

The disconnect between publishers and consumers stems from two main challenges when publishing linked data:

1. Finding ontologies: it is difficult to find ontologies on the Web [21], but when they are chosen, it is still not a trivial task to manually map real-world entities with concepts defined in ontologies [142].

2. Integrating data models: when existing data models have been published that model the same concepts, the problem shifts to choosing the data model that best suits the data and use-case.

Due to these challenges, data publishers tend to create their own data model or select a single existing data model, potentially hindering the interoperability of the dataset with other data models. Therefore, it is common to find several valid, existing data models to represent the same domain and no clear agreement exists between publishers over a best approach. For example, in the library data domain, several data models have been implemented over the years [26]: different libraries use the Dublin Core standards, BIBO [143], FOAF [62], or SKOS [144]. These are all well-established standards that are valid and fit well in the library data domain. Therefore, this practice leads to several data models existing to model the same domain and it is not always obvious which one fits a specific use-case. In the case of a data publisher trying to expose their data in RDF, while still integrating it with existing datasets in the domain, it is not an easy decision to choose a data model for their data.

In this chapter, we describe methods for generating data model candidates from a knowledge graph of published RDF data sources. Figure 4.1 shows the steps of the framework described in this chapter, which include candidate generation and ranking. The candidate generation stage parses the input dataset provided in a structured or semi-structured format and extracts unranked entity type and property candidates. The entity type, datatype property, and object property candidates are independently processed, and we propose methods to rank the extensive list of candidates obtained in the candidate generation stage. These independent scores are based on content- and graph-based measures obtained from the document store and ontology graph built based on the methods described in Chapter 3. These scores favour candidates that follow the vocabulary practice consensus extracted from the background knowledge graph and emphasise interoperability between datasets. These individual scores are aggregated into a single score per triple pattern in the input dataset that considers the consistency with the models in the knowledge graph and the proposed data model. It should be noted that when we mention consistency with the data model, we are not referring to logical consistency within the ontologies, e.g., verifying if disjoint axioms are not being ignored. The data model consistency is verified in terms of the frequency of a triple in the background knowledge graph and consistently suggesting the same candidate for the same input. Therefore, this consistency score takes into account the co-occurrence of the triple and pairwise combinations of ⟨domain – property ⇾ range⟩ in the knowledge graph to score the consistency of the candidate triples. Then it maximises consistency across triples to make sure that the same resource has the same or similar candidates proposed throughout the data model. We propose experiments to test these methods and show the usefulness of their application. These experiments evaluate methods for which a ground truth is available and show an overview of the results when one is not available. Therefore, we discuss the strengths and limitations of the framework to delineate the context of useful applications for the methods. We also present a demonstration of a prototype for a potential interface for the application of the methods.

In the following sections, we will first discuss the related work and then describe in detail the generation and ranking stages of the framework, i.e., entity type, datatype property, and object property generation and ranking. For these descriptions, we will make use of the notations described in Appendix B. We present and discuss the experiments with the proposed methodology and describe a demonstration of the output of the framework. Finally, we present the conclusions, including limitations and future work.

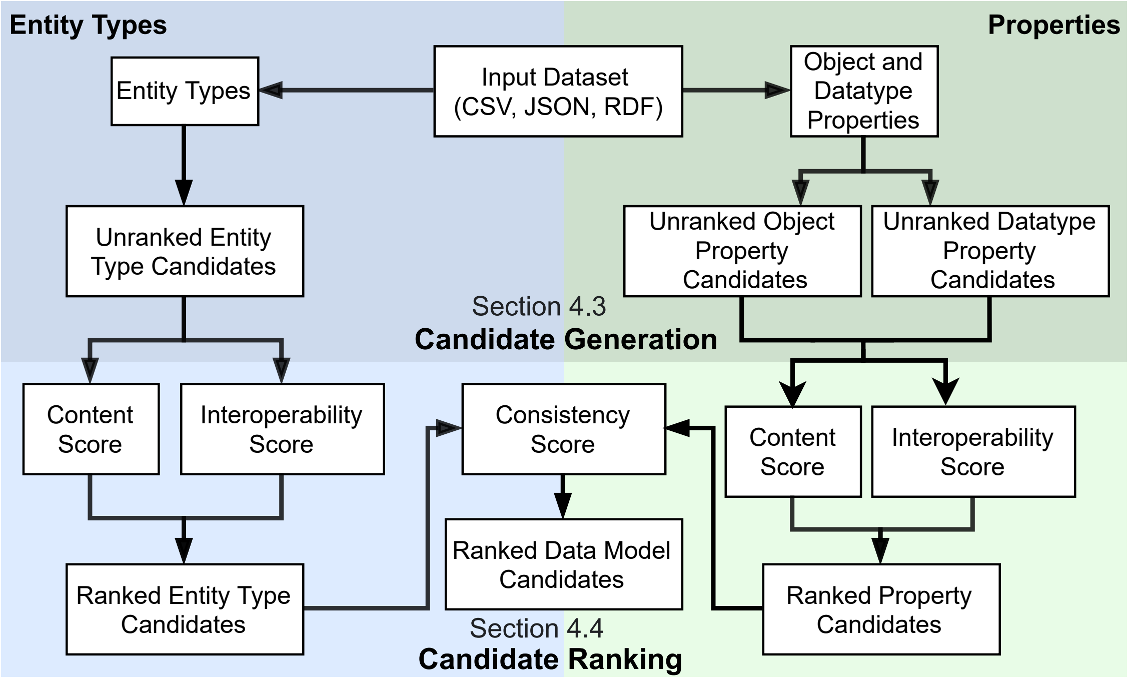

Entity Types
Properties
Entity Types
Input Dataset (CSV, JSON, RDF)
Object and Datatype Properties
Unranked Entity Type Candidates
Unranked Object Property Candidates
Unranked Datatype Property Candidates
Section 4.3
Candidate Generation
Content Score
Interoperability Score
Consistency Score
Content Score
Interoperability Score
Ranked Data Model Candidates
Ranked Entity Type Candidates
Ranked Property Candidates
Section 4.4
Candidate Ranking

Figure 4.1: Overview of the framework components described in this chapter.

4.2 RELATED WORK

The problem of generating and ranking entity type and property candidates is part of the schema matching research topic. Schema matching finds correspondences between schemas by relating their relationships and concepts and enabling the exchange of data between aligned sources. Approaches have been proposed that use metadata, data instances, or a combination of both to generate mappings between schemas. In RDB, it is common to find approaches that use metadata (e.g., schema constraints or query logs), but there are also hybrid metadata/data solutions proposed [145–148]. The surveys Rahm and Bernstein [149] and **a_survey_2017** include more background information and solutions for schema matching in RDB. Schema matching solutions have also been proposed for structured to semi-structured data not included in databases [150, 151]. Aligning CSV tables is also commonly associated with the schema matching problem [152–155].

When matching datasets that follow the RDF data model, schema matching becomes an ontology matching problem (more details in Section 2.2.2). RDF data is usually modelled by ontologies that facilitate the integration of data. Therefore, aligning different RDF datasets fits the ontology matching field. Approaches have been proposed that exploit instance-level links to find matches between RDF datasets using entity overlap [121] or string semantic similarity [156].

It is also common to find works on schema matching between heterogeneous sources and RDF data. The goal of these tasks is usually to integrate or find overlaps between unstructured or semi-structured data sources with RDF data to produce an interoperable knowledge graph. In Bikakis et al. [157], the authors describe a survey of approaches to integrate an XML Schema and an RDF Schema, and, in Hertling and Paulheim [158], authors match a shallow schema of classes and properties extracted from wiki infoboxes to the DBpedia ontology using string similarity matching approaches. For more information, a survey has been conducted over the data integration problem, including schema matching, covering various perspectives and data models [85].

Similarly to these schema matching approaches, in this chapter, we match the schema between a source and a target, where the source can follow an RDF, JSON, or CSV data model and we match it against a knowledge graph that features multiple schemas. Contrary to the schema matching approaches discussed, we do not aim to find the best schema mappings, but, instead, we produce a set of candidate correspondences to facilitate the task of selecting the mapping that makes more sense in the context of the input dataset and the use-case. Our methods also propose a novel approach to candidate ranking by considering not only accuracy, also interoperability and consistency with several data sources.

When looking specifically at entity types matching, early approaches focused on finding and ranking entity types in search queries [159, 160] used a single ontology and exploited its hierarchy to produce results. A more recent approach [161] uses a fixed group of ontologies (DBpedia, YAGO, and schema.org), chosen due to pre-existing mappings between them, as background knowledge to choose the best entity type to match an input. This approach also uses a search-based approach to generate candidates and applies a ranking methodology to an input unstructured text. The authors of this paper specifically use the frequency of an entity type in the knowledge graph, the frequency of neighbouring entities sharing the same entity type, and `owl:sameAs` links. Then they use Term Frequency-Inverse Document Frequency (TF-IDF) in combination with these frequencies to find and rank the best matching labels and extract their entity types. The authors also use the ontology hierarchy to further rank the entity types, using the depth of the entity type, the number of ancestors, and the depth of the ancestors. Finally, the authors consider the context of the entity, using frequency-, graph-, and text-based methods to rank the entity types based on contexts extracted from the source document. This approach has several similarities to the approach proposed in this thesis since we also use search-based methodology to generate candidates and apply content- and graph-scores to rank the candidates. However, our approach does not take into account context since currently our framework does not accept unstructured text as an input and, therefore, there is no context to be extracted. Furthermore, while Tonon et al. [161] focus on entity types only, we propose a methodology to match the whole data model of an input dataset that not only ranks candidates based on content but also focuses on maximising interoperability with existing RDF datasets.

Other approaches to entity type matching focus on exploiting relationships between entities to match entity types [162, 163]. Both of these approaches again focus only on predicting entity type candidates, while our proposed framework matches entity types and considers the whole data model. Similarly to these works, our proposed framework also uses the co-occurrence of a predicate with a subject/object when ranking candidates. However, this score, called "consistency score", is applied in the context of scoring the whole data model and not only the entity type.

When focusing on the schema matching problem in the alignment of properties, several advances have been made using ontology matching techniques. ILLIADS [164] aligns ontologies, including their properties, by using a combination of lexical and structural similarities, and a clustering algorithm. In PARIS [165], the authors propose a framework to perform ontology matching of classes, instances, and properties using a probabilistic framework. In the case of object properties, it is common to find approaches that refocus the problem as a task to find relationships between individuals. RelFinder [166] is an example of this approach since they perform a search over entities in an RDF dataset to find their relationships, while other approaches use schema paths to find relationships between entity individuals through their entity types [167, 168]. Pereira Nunes et al. [169] also finds relationships between entities, but, besides considering the path between entities, also takes into account external resources by measuring the co-occurrence of entities in Web documents. ROSA [170] focuses on finding relationship matches by mining rules in the alignment of instances, while Koutraki, Preda and Vodislav [171] uses a supervised learning approach to align relationships. Similarly, Gunaratna et al. [172] uses a statistical approach to generate object property matches from aligned entity instances. Contrary to these last approaches, our proposed framework does not depend on prior mappings between the entities of the datasets it is trying to match. Due to the holistic nature of our framework, we extract the object properties from the entity type candidates generated and ranked. Therefore, we do not need the datasets to be aligned *a priori.* Overall, in the work proposed in this thesis, we consider a broader view of relationships by taking into account schemas from different datasets and providing candidates that match them. Moreover, our relationship matching strategy works not only for RDF ontologies but also for semi-structured datasets.

Looking at matching datatype properties, the focus is usually in analysing the values of the datatype properties to find matches. Pereira Nunes et al. [173] proposed a method to match datatype property candidates using mutual information at the instance level to generate candidates and applying a genetic algorithm to create mappings between datatype properties. The authors of this work focus on finding the best matches for simple and complex datatype properties. In our work, we aim to find a broad list of candidates from several existing data models and rank them according to different measures. Therefore, again our approach differs from existing approaches to datatype property matching since our goal is fundamentally different since we are not trying to find 1-to-1 matches between two properties but instead focus on finding the best candidates from a wide pool of possibilities.

The problem of generating datatype property candidates is akin to the task of tabular data interpretation, where table headers represent entity types or properties for each row, which represents an entity. The first task in tabular data annotation is identifying the type of data in each column and the relations between columns. Syed et al. [174] uses Wikipedia as background knowledge, while other approaches use HTML (Hypertext Markup Language) tables obtained from the Web [152, 175, 176] to interpret tabular data.

4.3 GENERATING CANDIDATES

In this stage of the framework, we generate an exhaustive list of candidates to match entity types, datatype, and object properties between the input data and the knowledge graph. As a preliminary pre-processing step, each input dataset is parsed and converted to the same JSON structure of the knowledge graph and is stored and indexed in the document store. For ease of understanding, we assume that the input data D takes the shape of a set of $\langle \mathsf{s} - \mathsf{p} \rightharpoonup \mathsf{o} \rangle$ triples, which directly translates to RDF data. For JSON and CSV, we loosely adapt the data to fit this model. For JSON data, we consider the identifier of the document as the subject s (if no identifier is given, one is created), the attributes as predicates p, and values as objects o. For CSV data, we assign a row identifier if none is given and use it as subject s, the header or column number as the predicate p, and the value of each row/column as object o.

Figure 4.2 shows the framework components with added red arrows indicating the structures involved in generating each one of the candidates. Therefore, the entity type candidates are extracted by searching matching entities in the document store. Datatype properties are obtained by using a classification model to predict candidates from the values of the properties in the input data. Finally, object properties are obtained after generating and ranking the entity type candidates. We obtain these properties by extracting the domain and range of the object properties in the input data and then extracting the object property candidates through the pairwise combination of all entity type candidates of the domain and range. Then we extract the object property candidates from the relationships between all pairwise comparisons of the entity type candidates in the domain and range. The following sections describe each one of these processes in more detail.

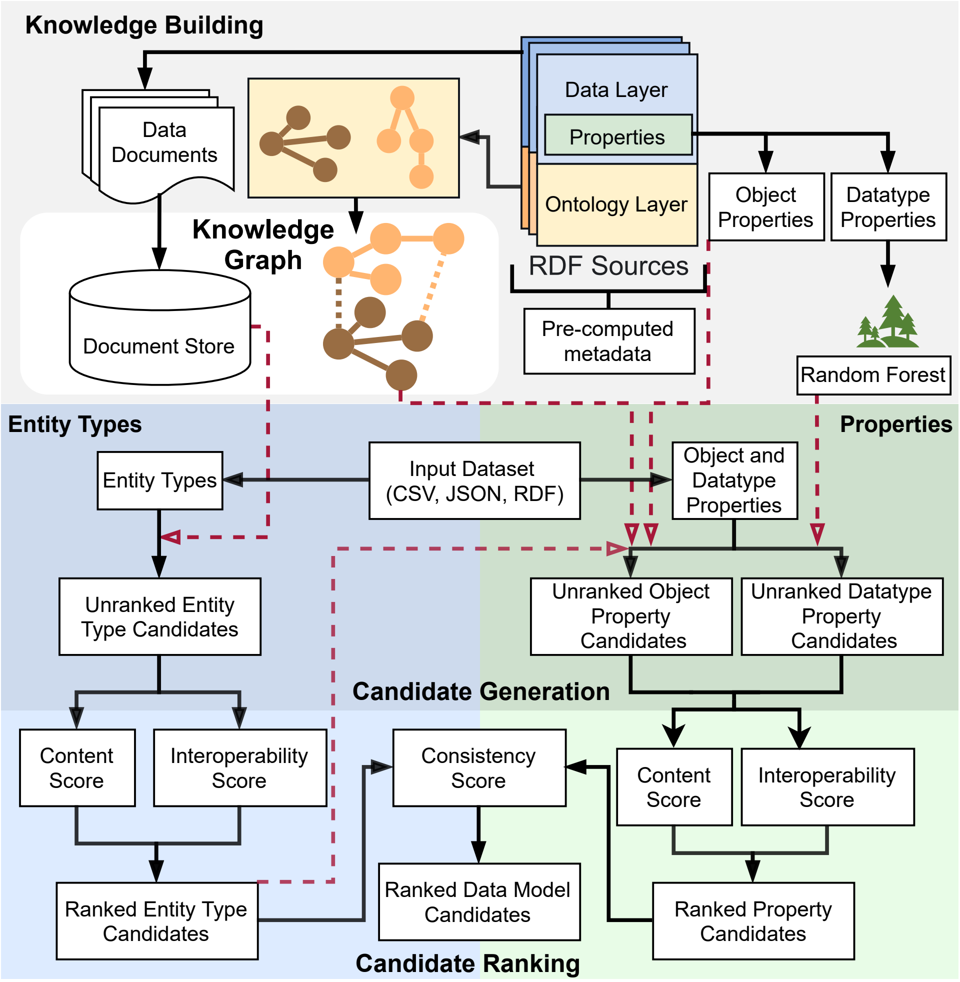


Figure 4.2: Overview of the framework with red arrows indicating the actors involved in the generation of each type of candidate.

4.3.1 Entity Types

This stage generates an exhaustive list of entity type candidates for entities in the input dataset D. As a first optional step, the user can provide descriptors dsc, i.e, column names, attributes, or predicates that describe entities. These descriptors should contain values that identify the entities, even if with some ambiguity, to facilitate the search in other datasets that might not follow the same standards. For example, considering a dataset with book entities, the user might provide the columns, attributes, or predicates that include the titles of books. If not provided, the search is performed over every column, attribute, or predicate related to an entity in the dataset. Optionally, entity descriptors can also be provided for the datasets in the document store. We found empirically that when entity descriptors for both input dataset and document store are provided, they improve the accuracy and efficiency of the candidate generation and ranking.

Algorithm 4.1: Entity Type Generation

```
Input: $S$: document store
Input: $D$: input dataset with triples $\langle \mathsf{s} - \mathsf{p} \rightarrow \mathsf{o} \rangle$
Input: $n$: size of random sample
Input: $w$: size of the search result window
Input: $dsc$: list entity descriptors (optional)
Output: $C_{et}$: mapping of key type $t$ and entity $e$ to value, which is a list of tuples
        with entity type candidates $o$, entity labels $L_e$, and candidate labels $L_e$
// extracts unique types or type equivalent elements for entities in $D$
$T \leftarrow unique\_types(D)$
$C_{et} \leftarrow \{\,\}$
for $t$ in $T$ do
    // returns a random sample of $n$ documents in $D$ of type $t$
    $E_t \leftarrow random\_sample(D, t, n)$
    for $e$ in $E_t$ do
        $C_{et}[\langle t, e \rangle] \leftarrow [\,]$ if key $t \notin C_{et}$
        // input label extraction
        $L_e \leftarrow \{o \mid s, p, o \in e \wedge literal(o)\}$
        for $l$ in $L_e$ do
            // searches $S$ for label $l$ and returns a maximum of $w$ search results
            $M \leftarrow search(S, l, w)$
            for $m$ in $M$ do
                // candidate label extraction
                $L_c \leftarrow \{o \mid s, p, o \in m \wedge literal(o)\}$
                // entity type extraction
                for $s, p, o$ in $m$ do
                    if $p = \mathit{rdf{:}type}$ then
                        $C_{et}[\langle t, e \rangle]$.append($\langle o, L_e, L_c \rangle$)
                    end
                end
            end
        end
    end
end
```

Algorithm 4.1 shows the process of obtaining entity type candidates for the entity types found in input dataset D. See Appendix C.1 for an illustrative example of the algorithm workflow.

For RDF input data, we assume the entity type is the object of the `rdf:type` predicate. In JSON data, we ask the user to supply the attribute which denotes the type of the document. If no type is supplied, then the JSON data is treated like CSV data, where each column is assumed to be a potential entity to have an entity type assigned. For both of these cases, if descriptors are supplied, we find only entity type candidates for the columns or attributes indicated. The first step of the algorithm is, therefore, to extract the set of potential unique types from the input dataset. Following the logic described, the function $unique\_types(D)$ extracts the unique types or type equivalent elements T for entities in D.

Then, the function $random\_sample(D, t, n)$ returns a random sample of n documents in D of type t. These documents represent all triples associated with a subject of type t in RDF input data, an individual JSON document of type t, or a row in a CSV file that contains a value for column t. In our implementation, we use ElasticSearch's random score function[1] to randomly sample the documents.

Next, for each input triple, we extract all objects that are literals (i.e., $literal(s) = True$) from all properties in entity document e or for all properties in the descriptors dsc if these were provided. The function $search(S, l, w)$ searches the document store S for label l and returns a maximum of w search results. In our implementation, we use ElasticSearch's multi-match query[2] to search for full-text matches of label l in the datatype property values indexed in the document store. These fields are replaced by entity descriptors if these are provided for the knowledge graph data by the user. This algorithm then returns a mapping between entity type t for each entity e and an extensive list of randomly sampled entity type candidates, including the labels that matched between input and document store entities.

4.3.2 Datatype Properties

The datatype property candidates are generated from the Random Forest models (RF) trained using the method described in Chapter 3. We train a model for each RDF data source individually and for the knowledge graph as a whole. Algorithm 4.2 shows the process of obtaining datatype property candidates for datatype properties in input data D. See Appendix C.3 for an illustrative example of the algorithm workflow.

First, we generate the set DP from the data in input D with unique datatype properties in the input data D. Then we obtain a random sample of entities E_t in the input data that contain each of the datatype properties in DP. After extracting the literal values L of the properties, we use the trained random forest models RF to classify the value with a datatype property from the knowledge graph. The $predict(rf, l)$ function returns a list ordered by the probability estimates of each datatype property in the classification models matching the literal value. Therefore, we obtain the datatype property candidates C_{dp} by saving the sorted prediction results for each value in each model.

[1] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-function-score-query.html#function-random (Accessed in September 2020)

[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html (Accessed in September 2020)

Algorithm 4.2: Datatype Property Generation

```
Input: S: document store
Input: RF: random forest models
Input: D: input dataset with triples ⟨s – p –▹ o⟩
Input: $n$: size of random sample
Output: $C_{dp}$: mapping of key datatype property $dp$, model $rf$, and entity $e$ to
        value, which is a list of ordered lists $p$ of predicted datatype property
        candidates $c$.
// extracts datatype properties p and their domain type $d_t$
$DP \leftarrow \{p \mid s, p, o \in D \wedge literal(o)\}$
$C_{dp} \leftarrow \{\,\}$
for $dp$ in $DP$ do
    // returns a random sample of n documents in DP where property dp exists
    $E_t \leftarrow random\_sample(DP, dp, n)$
    for $e$ in $E_t$ do
        // property value extraction
        $L \leftarrow \{o \mid s, p, o \in e \wedge p = dp\}$
        for $l$ in $L$ do
            for $rf$ in $RF$ do
                $p \leftarrow predict(rf, l)$
                $C_{dp}[\langle dp, rf, e\rangle] \leftarrow [\,]$ if key $\langle dp, rf\rangle \notin C_{dp}$
                $C_{dp}[\langle dp, rf, e\rangle].append(p)$
            end
        end
    end
end
```

4.3.3 Object Properties

The object property candidate generation step depends on the generation and ranking of entity type candidates (see Section 4.4 for more details on the ranking of entity types). Algorithm 4.3 shows the process of obtaining object property candidates for object properties in input data D. See Appendix C.5 for an illustrative example of the algorithm workflow.

First, we obtain all triples which do not have literals as their object. For each property in these triples, we extract their domain and range entity types with the function $type(e)$. The object property candidate map C_{op} is obtained by retrieving the edges in the ontology graph that exist between the pair of subject and object candidates. The function $get\_edges(G, s, t)$ returns a tuple with the object property edges op in graph G between source vertex s and target vertex t and their data provenance $prov$, e.g., $get\_edges(G, s, t) = [\langle op_1, prov1\rangle, \langle op_2, prov2\rangle]$. The ontology provenance represents the RDF data sources use the edge suggested as an object property. This information is stored as a property of the edge when the edge is introduced in the ontology graph.

Algorithm 4.3: Object Property Generation

```
Input: G: ontology graph
Input: D: input dataset with triples $\langle s - p \rightarrow o \rangle$
Input: $CS_{et}$: output of Algorithm 4.4
Output: $C_{op}$: mapping of key domain entity type $d_t$, object property $p$, and
        range entity type $r_t$ to value, which is a list of tuples with object
        property op and its provenance o.
$OP \leftarrow \{(type(s), p, type(o)) \mid s, p, o \in D \wedge \neg literal(o)\}$
for $d, p, r$ in OP do
    for $c_d, c_r$ in $CS_{et}[d_t] \times CS_{et}[r_t]$ do
        // Returns the edges between $c_d$ and $c_r$ and their ontology provenance
        $op \leftarrow get\_edges(G, c_d, c_r)$
        $C_{op}[p].append(op)$
    end
end
```

4.4 RANKING CANDIDATES

To rank the entity type, datatype, and object property candidates generated, we use three scores: content, interoperability, and consistency score. This step ranks individually the lists C_{et}, C_{dp}, and C_{op} with the content score, returning CS_{et}, CS_{dp}, and CS_{op}, and the interoperability score, returning IS_{et}, IS_{dp}, and IS_{op}. The content score focuses on measuring individual characteristics of the entity type or property candidates, such as frequency in the search results and distance in the ontology graph. The interoperability score focuses on boosting the score of candidates with higher connectivity in the ontology graph. Finally, the consistency score transforms the individual lists of ranked candidates into cohesive data model candidates DM, and combines the individual scores with a score of the frequency of triple patterns in the input data.

The next sections detail all the metrics used in each type of score. We then describe the specific scoring methodology used for each type of candidate in the data model.

4.4.1 Content Score

The content score combines metrics based on string similarity, different types of frequencies, and graph distances into a single score $CS_c \; \forall \, c \in \{C_t, C_{dp}, C_{op}\}$. The final candidate ranking produces a set of candidates that are sorted according to their appropriateness in matching entity types of the input data, considering the metrics used. We obtain this result by using different combinations of the measures described in the next sections.

Content Score Metrics

Here we present a set of generic metrics that are used to measure the appropriateness of a candidate to fit an input entity type or property. These metrics were empirically added to framework by studying the needs and characteristics of the use-case datasets. The intuition behind each measure is discussed individually in their respective sections.

In each measure, we refer to a generic list of candidates C that can represent any of the resources (i.e., entity types or properties) in the data model of the input data.

STRING SIMILARITY When comparing labels of entities we consider that a label is more likely to be correct if it is closer to the input label. Therefore, we applied string similarity measures to find the degree of similarity between input and candidate labels.

Nowadays, complex methodologies exist that allow the computation of string similarity. Such approaches, for example, make use of word embeddings using techniques such as word2vec [177] or GloVe [178], to train machine learning models to predict the similarity between words. For our framework, however, we were looking for computational inexpensive methods that would give a fast and easily explainable result. Therefore, we tested classical string similarity methods (see Section 4.5.3) and devised a modified string similarity score sim based on the concept of n-grams. This metric was created through observation of the use-case datasets, which have labels with high variability in terms of string size.

An n-gram is a sequence of n items belonging to a given string. In this work, we use a bi-gram ($n = 2$) function $n\text{-}gram(s)$ to decompose a string s into a set of pairs at the letter and word level, e.g. *Harry Potter* is decomposed into {("_h"), ("ha"), ("ar"), ("rr"), ("ry"), ("y "), (" p"), ("po"), ("ot"), ("tt"), ("te"), ("er")} at the letter level and {("_", "harry"), ("harry", "potter")} at the word level. Left padding (_) is added to emphasise prefixes. The intuition behind this method is that shorter strings should be weighed more heavily in terms of their letter bi-gram overlap, while longer strings should favour word bi-gram overlap.

We first pre-process the strings by removing punctuation and stop words, then we define the n-gram similarity with a Jaccard index $g\text{-}sim$ between sets A and B as follows:

$$\begin{aligned} union(A, B, n) &= |n\text{-}gram(A) \cup n\text{-}gram(B)| \\ g\text{-}sim(A, B, n) &= \frac{|n\text{-}gram(A) \cap n\text{-}gram(B)|}{union(A, B, n)} \end{aligned} \tag{4.1}$$

We then define the letter and word level n-gram similarities $l\text{-}sim$ and $w\text{-}sim$ using the letter and word n-grams A_l, B_l and A_w, B_w as follows:

$$\begin{aligned} l\text{-}sim(A_l, B_l, n) &= g\text{-}sim(A_l, B_l, n) \\ w\text{-}sim(A_w, B_w, n) &= g\text{-}sim(A_w, B_w, n) \end{aligned} \tag{4.2}$$

Lastly, we compute the string similarity metric $s\text{-}sim$ as follows:

$$\begin{aligned} \mathrm{inv}(A, B, n) &= \begin{cases} \mathrm{union}(A, B, n), & \text{if } \mathrm{union}(A, B, n) \leqslant 1 \\ \frac{1}{\mathrm{union}(A,B,n)-1}, & \text{if } \mathrm{union}(A, B, n) > 1 \end{cases} \\ \text{s-sim}(A, B, n) &= \text{w-sim}(A_w, B_w, n) \cdot (1 - \mathrm{inv}(A_w, B_w, n)) + \\ &\quad \text{l-sim}(A_l, B_l, n) \cdot \mathrm{inv}(A_w, B_w, n) \end{aligned} \tag{4.3}$$

The word-level similarity $w\text{-sim}$ gives a better understanding of the equivalence of two entities based on their labels. However, when the labels have few words, this metric loses its meaning. Therefore, the $\mathrm{inv}(A, B, n)$ function balances the weight given to $w\text{-sim}$ versus $l\text{-sim}$. For example, when comparing *Harry Potter* with *Harry's Pottery*, $l\text{-sim} = 0.73$, $w\text{-sim} = 0.0$, and $s\text{-sim} = 0.24$, therefore, representing the level of similarity between the strings but emphasising that, at the word level, the strings do not overlap.

Finally, considering two sets of strings L_e and L_c, we obtain the string similarity score sim by computing the maximum pairwise string similarity between the two sets as:

$$\mathrm{sim}(L_{e,t}, L_{c,t}) = \max_{\substack{l_e \in L_{e,t} \\ l_c \in L_{c,t}}} \text{s-sim}(l_e, l_c, n) \tag{4.4}$$

We select candidates that have $\mathrm{sim} >= t$, where t is a user-provided threshold.

SEARCH RESULTS FREQUENCY The search results frequency score $\mathrm{freq}_s(c)$ follows the intuition that the most common candidate in the list of search results has a higher likelihood of being a good candidate. Considering that $\mathrm{count}(c)$ is a function that represents the raw count of the candidate c in a list of candidates C, we apply the following transformation to obtain a final $\mathrm{freq}_s(c)$:

$$\begin{aligned} \mathrm{freq\_norm}(c) &= \log(1 + \mathrm{count}(c)) \\ \mathrm{freq}_s(c) &= \frac{\mathrm{freq\_norm}(c)}{\max\limits_{c \in C} \mathrm{freq\_norm}(c)} \end{aligned} \tag{4.5}$$

KNOWLEDGE GRAPH FREQUENCY This metric measures how frequent a resource is in the knowledge graph. We extract the count of a candidate c from the ETF map (see Section 3.5) and compute the normalised frequency per RDF data source ds as:

$$\mathrm{freq}_{kg}(ds, c) = \frac{\mathrm{EFT}[ds][c]}{\max\limits_{c \in \mathrm{EFT}[ds]} \mathrm{freq}_{kg}(ds, c)} \tag{4.6}$$

BORDA SCORE This score is based on the Borda count election method [179], where voters rank candidates in order of preference. In opposition to majority election, the Borda count elects the candidate that is more broadly accepted by the voters. It is therefore a consensus-based voting system where the voters, in this case, correspond to the independent candidate results per sampled entity type or property, i.e., the documents returned in the search results per sampled entity in the input data sorted by ElasticSearch score or the ranking of predictions obtained by the classification model per sampled datatype property.

The intuition behind this score is that when the initial candidate generation already has a preliminary ranking, as is the case with the entity type and datatype property candidates, instead of just applying flat frequency we consider the ranking in the list of candidates.

Borda count works by distributing a number of points determined by the number of candidates n. Each candidate c receives $n - r_v$ points, where r_v represents the ranking of the candidate in a ballot of voter v, starting from rank 0 for the candidate ranked first. For example, with $n = 3$ candidates per ballot, the candidate ranked first will get 3 points, the 2nd will get 2, and the 3rd will get one point.

With $N = |C|$ and a pool of voters $V = \{v_1, v_2, \dots, v_k\}$, we compute the Borda Score $\mathrm{borda\_score}(c)$ for candidate c with:

$$\begin{aligned} \mathrm{borda\_count}(c, V) &= \sum_{k=1}^{|V|} n - r_{v_k} \\ \mathrm{borda\_score}(c, V) &= \frac{\mathrm{borda\_count}(c, V)}{\max\limits_{c \in C} \mathrm{borda\_count}(c, V)} \end{aligned} \tag{4.7}$$

RESOURCE FREQUENCY PROPORTION We use the NP map (see Section 3.5) to extract the neighbourhood proportion of each candidate and the input resource they are matching. This score captures the intuition that the same types of resources should represent a similar proportion of entities in the datasets of the same domain. For example, entity types that define a book in a library should exist in a similar proportion in different libraries.

The resource proportion score $\mathrm{freq\_p}(c, r)$ for candidate c is obtained from the proportion p_r of the resource r extracted from NP and the proportion of the candidate p_c as:

$$\mathrm{freq\_p}(c, r) = 1 - |p_r - p_c| \tag{4.8}$$

DISTANCE TO SOURCE RESOURCE Using the edge-enriched ontology graph, we compute the distance between the source resource r and each candidate c in the candidate list C. This score follows the intuition that candidates closer to the source resource in the ontology graph are more likely to be good candidates. We calculate the score $\mathrm{dist}_r(r, c)$ between a source resource r and a candidate c as:

$$\mathrm{dist}_r(r, c, G) = 1 - \frac{\log(\mathrm{shortestDistance}(r, c) + 1)}{\log(\max\limits_{c \in C} \mathrm{dist}_r(r, c, G) + 1)} \tag{4.9}$$

CANDIDATE DISTANCE This metric measures the pairwise distance in the ontology graph between candidates and is based on the assumption that candidates that are distant from the other candidates are less likely to be desirable. For example, if C_t includes `bibo:Book`, `bibo:Document`, `schema:Book`, `frbr:Work`, and `dcterms:Agent`, by calculating the pairwise distances between those five candidates we would find that the first four are closer in the ontology graph than the last and, therefore, this last candidate is considered less likely to be appropriate.

We calculate the Candidate Distance Score dist_c by averaging the inverse of the distance between c and all other candidates as follows:

$$\begin{aligned} \forall c, c_n \in C \times C \setminus \{c\} : \mathit{distance}(c, c_n) &= \frac{1}{\mathit{shortestDistance}(c, c_n)} \\ \mathit{dist}_c(C) &= \frac{\mathit{distance}(C)}{\max\ \mathit{distance}(C)} \end{aligned} \tag{4.10}$$

Content Scoring Methodologies

We consider three types of elements of the data models in the knowledge graph and input datasets: entity types, datatype properties, and object properties. Through experimentation, we tested different combinations of metrics for each element and chose the ones that not only obtained good results in the evaluation but also yield better fined-tuned results for the expected outcomes. Therefore, the content scoring methodologies differ between these elements to accommodate their different characteristics. In the next sections, we present the individual content scoring methodologies for each element type in the data model.

ENTITY TYPE CONTENT SCORING Considering a list of candidates C_{et} with all the generated unranked entity type candidates with source entity labels L_e and target candidate labels L_c, Algorithm 4.4 shows the processes involved in computing the content score for entity type candidates, obtaining the ranked candidates CS_{et}. See Appendix C.2 for a hypothetical example of the application of the algorithm over the candidates generated in example Figure C.1.

Algorithm 4.4 starts by calculating the string similarity of each candidate and filtering out the candidates with similarities below threshold α. In the last step of the string similarity calculation, the algorithm selects the maximum similarity found per type and per entity for each candidate to represent the similarity sim of the candidate stored in the mapping S.

Then the algorithm creates a mapping C of each source entity type to all the unique candidates in mapping S, which is used to compute dist_c during the next step.

Finally, we compute the content score metrics by iterating over mapping S. First, we obtain the frequency of each candidate c, considering the raw count equivalent to the size of the scores list. Additionally, for this content score methodology, we consider the resource frequency proportion freq_p, the distance to source dist_r, and the candidate distance dist_c. We combine these scores into a final cs score per candidate per type and select the k elements with the highest values of cs per type t.

Algorithm 4.4: Entity type content scoring algorithm

```
Input: G: ontology graph
Input: $C_{et}$: output of Algorithm 4.1
Input: $\alpha$: filtering threshold for string similarity score
Input: k: cut-off value for the selection of highest ranked candidates in a list
Output: $CS_{et}$: mapping of key types t to value, which is a list of tuples with
        entity type candidates c and content score cs
$S \leftarrow \{\}$
// Calculate candidate string similarity per type and candidate
for key-value pairs $(\langle t, e\rangle, C)$ in $C_{et}$ do
    $G \leftarrow \{\}$
    for $\langle c, L_{e,t}, L_{c,t}\rangle$ in C do
        $G[c] \leftarrow [\,]$ if key $c \notin G$
        $s \leftarrow \mathrm{sim}(L_{e,t}, L_{c,t})$
        if $s >= \alpha$ then
            $G[c].\mathrm{append}(s)$
        end
    end
    for key-value pairs $(c, \mathrm{sims})$ in G do
        $S[\langle t,c\rangle] \leftarrow [\,]$ if key $\langle t,c\rangle \notin S$
        $S[\langle t,c\rangle].\mathrm{append}(\max(\mathrm{sims}))$
    end
end
// Compute set of unique candidates
$C \leftarrow \{\}$
for key $(t,c)$ in S do
    $C[t] \leftarrow C[t] \cup \{c\}$
end
// Compute content score
$CS_{et} \leftarrow [\,]$
for key-value pairs $(\langle t,c\rangle), \mathrm{scores})$ in S do
    $\mathrm{freq} \leftarrow \mathrm{freq}_s(c)$ where $\mathrm{count}(c) = |\mathrm{scores}|$
    $cs \leftarrow \mathrm{freq} \cdot \mathrm{mean}(\mathrm{mean}(\mathrm{scores}), \mathrm{freq\_p}(c,t), \mathrm{dist}_r(t,c,G), \mathrm{dist}_c(C[t]))$
    $CS_{et}[t].\mathrm{append}(\langle c, cs\rangle)$
end
$CS_{et} \leftarrow \mathrm{top}_k(CS_{et}, \mathrm{sort\_key} \leftarrow cs)$
```

DATATYPE PROPERTY CONTENT SCORING Considering the datatype property candidate mapping C_{dp} generated through the processes described in Algorithm 4.2, Algorithm 4.5 describes the process of ordering these candidates according to the combination of content score metrics. See Appendix C.4 for a hypothetical example of the application of the algorithm over the candidates generated in example Figure C.3.

First, the algorithm computes the Borda score per datatype property dp, per entity e, and per model rf for each datatype property candidate in the ranked prediction p. In step 2, we aggregate the Borda score results per input datatype property and candidate. In step 3, we average the Borda scores that each candidate got in the different entities and models and average this score with the distance to source dist_t. Finally, the mapping $\mathrm{CS}_{\mathrm{dp}}$ returns the top_k candidates per datatype property dp considering the highest content score cs computed.

Algorithm 4.5: Datatype property content scoring algorithm

```
Input: G: ontology graph
Input: $C_{dp}$: output of Algorithm 4.2
Input: $CS_{et}$: output of Algorithm 4.4
Input: k: cut-off value for the selection of highest ranked candidates in a list
Output: $CS_{dp}$: mapping of key datatype property dp to value, which is a ranked
        set of tuples of datatype property candidate c and its content score cs
// Computes borda score of each candidate per property dp and per model rf
$S_1 \leftarrow \{\}$
for key-value pairs $(\langle dp, rf, e\rangle, p)$ in $C_{dp}$ do
    $S_1[\langle dp, rf\rangle] \leftarrow [\,]$ if key $\langle dp, rf\rangle \notin S_1$
    for c in p do
        $S_1[\langle dp, rf\rangle].append(\langle c, borda\_score(c, p)\rangle)$
    end
end
// Aggregates Borda results per property dp per candidate c
$S_2 \leftarrow \{\}$
for key-value pairs $(\langle dp, rf\rangle, \langle c, borda\_score\rangle)$ in $S_1$ do
    $S_2[\langle dp, c\rangle] \leftarrow [\,]$ if key $\langle dp, c\rangle \notin S_2$
    $S_2[\langle dp, c\rangle].append(borda\_score)$
end
// Computes cs score per property and candidate
$S_3 \leftarrow \{\}$
for key-value pairs $(\langle dp, c\rangle, borda\_scores)$ in $S_2$ do
    $cs \leftarrow mean(mean(borda\_scores), dist_r(dp, c, G))$
    if $cs > S_3[\langle dp, c\rangle]$ then
        $S_4[(dp, c)] \leftarrow cs$
    end
end
$CS_{dp} \leftarrow [\,]$
for key-value pairs $(\langle dp, c\rangle, cs)$ in $S_3$ do
    $CS_{dp}[dp].append(\langle c, cs\rangle)$
end
$CS_{dp} \leftarrow top_k(CS_{dp}, sort\_key \leftarrow cs)$
```

object property content scoring Considering the object property candidates mapping C_{op} generated through the processes described in Algorithm 4.3, Algorithm 4.6 describes the process of ordering the object property candidates according to a combination of content score metrics. See Appendix C.6 for a hypothetical example of the application of the algorithm over the candidates generated in example Figure C.5.

Algorithm 4.6 first iterates over each candidate c and its provenance ontology o to extract $\mathtt{freq}_s(c)$ where the raw count is represented by the number of times a tuple in $\mathtt{edges}$ contains candidate c. The frequency $\mathtt{freq}_{kg}$ represents the number of documents in the knowledge graph that use object property candidate c. Finally, the algorithm computes $\mathtt{dist}_r$ and combines all scores into the final content score cs of candidate c. The algorithm returns the $\mathtt{top}_k$ object property candidates sorted by content score cs.

Algorithm 4.6: Object property content scoring algorithm

```
Input: G: ontology graph
Input: $C_{op}$: output of Algorithm 4.3
Input: k: cut-off value for the selection of highest ranked candidates in a list
Input: $\beta$: weight given to compute the final score
Output: $CS_{op}$: mapping of *key* datatype property op to *value*, which is a ranked
        set of tuples of object property candidate c and its content score cs
$CS_{op} \leftarrow \{\}$
for key-value pairs $(op, edges)$ in $C_{op}$ do
    for $c, prov$ in $edges$ do
        $freq_{kg} \leftarrow freq_{kg}(prov, c)$
        $dist_r \leftarrow dist_r(op, c, G)$
        $cs \leftarrow (dist_r \cdot \beta) + (freq_{kg} \cdot 1 - \beta)$
        $CS_{op}[op].append(\langle c, cs \rangle)$
    end
end
$CS_{op} \leftarrow top_k(CS_{op}, sort\_key \leftarrow cs)$
```

4.4.2 Interoperability Score

The interoperability score follows the intuition that candidates that are more interoperable within the background knowledge graph will not only be well-connected in the ontology graph but will also be closer to frequently used classes and properties within the knowledge graph. Therefore, the interoperability score focuses on re-ranking the candidate list obtained via the content score by boosting candidates that are well-connected in the ontology graph and are connected to frequent entity types in the knowledge graph. This score ensures that the candidates ranked first are not only good matches but also maximises the integration with frequently used entity types in the knowledge graph. Considering that the knowledge graph is composed of several potential candidates for each resource, this score improves the integration with related resources by ranking less specific resources higher since they are more likely to contain a broad network of equivalences and subsumptions.

Therefore, the interoperability score (IS) contains information from the Knowledge Graph (KG) and from a sub-graph of the ontology graph $G_e = (V, E_s)$ with $E_s \subseteq E$ representing ontology mappings, extended mappings, `owl:equivalentClass`, and `superClassOf` edges only. This subset of edges restricts the graph to the set of vertices that are equivalent, subsumed, or directly related.

This score is based on two metrics: *neighbourhood size* and *interoperability*. Both scores use the concept of i^{th} neighbourhood $N_G(v, i)$ of vertex v, which is the set of all vertices that lie at the distance i from v in graph G. Neighbourhood size represents the total number of vertices in $N_{G_e}(v, i)$, while the interoperability metric considers the frequency of these vertices in the knowledge graph. Higher scores in these metrics translate into a more connected and relevant neighbourhood. In a fully connected graph, the values of these metrics keep increasing until all vertices are included in this score. In contrast to the interoperability metric, the neighbourhood size rewards candidates with neighbours that do not appear in the knowledge graph but are still in the path of the candidate vertex in G_e. Entity types with high neighbourhood size scores but low interoperability metric scores are valuable for integration with datasets not included in the background knowledge graph.

Neighbourhood Size Metric

Considering V as the set of vertices in G_e corresponding to the candidates in CS obtained from the content score ranking of any data model resource, we calculate the neighbourhood size metric $NS(v, d_{max})$ as the sum of the number of vertices in the i^{th} neighbourhood of vertex $v \in V$ up to a maximum weighted distance d_{max}, i.e., the maximum distance to consider i^{th} neighbourhoods. Therefore, the neighbourhood size $NS(v, d_{max})$ is computed with:

$$NS(v, d_{max}) = \sum_{i=0}^{d_{max}} |N_{G_e}(v, i)| \qquad (4.11)$$

Interoperability Metric

The interoperability metric is defined as the sum of the frequencies $\mathtt{freq}(n)$ that each neighbourhood vertex n has in the knowledge graph KG over the i^{th} neighbourhood $\forall v \in V : N_{G_e}(v, i)$, where i is the weighted distance from v to a maximum of d_{max}:

$$\mathtt{interop}(v, d_{max}) = \sum_{n \in N_{G_e}(v, d_{max})} \mathtt{freq}(n) \qquad (4.12)$$

Interoperability Score Methodology

Considering r as a data model resource in $\{et, dp, op\}$, Algorithm 4.7 takes as input the CS_r mapping resulting from the content score algorithms respective to each resource and computes the interoperability score, IS_r. See Appendix C.7 for a hypothetical example of the application of the algorithm over the candidates generated in example Figure C.1.

Algorithm 4.7 first finds the vertex in graph G_e that corresponds to each candidate and then computes the neighbourhoods size and interoperability metrics. The mean of these two values is the final interoperability score is.

Algorithm 4.7: Interoperability score algorithm

```
Input: $G_e$: subset of the ontology graph with relevant edges only
Input: $CS_r$: mapping of key resource $r$ to value, which is a ranked set of tuples of
       resources candidate $c$ and its content score $cs$
Input: $d_{max}$: maximum neighbourhood distance to consider
Output: $IS_r$: mapping of key resource $r$ to value ranked set of tuples of resources
        candidate $c$ and its interoperability score $is$
$IS_r \leftarrow \{\}$
for key-value pairs $(r, \langle c, cs \rangle)$ in $CS_r$ do
    $v_c \leftarrow \mathrm{get\_vertex}(G_e, c)$
    $is \leftarrow \mathrm{mean}(\mathrm{NS}(v_c, d_{max}), \mathrm{interop}(v_c, d_{max}))$
    $IS_r[r].\mathrm{append}(\langle c, is \rangle)$
end
```

4.4.3 Consistency Score

The consistency score follows the intuition that a candidate data model is more desirable if the suggested triples frequently appear together in the knowledge graph and, at the same time, the same candidate is consistently suggested at the same rank for each input. Therefore, the consistency score takes as input the results from the individual scoring methodologies of entity types and properties and re-ranks them considering their combined frequency in the knowledge graph but also the homogeneity of the candidate data models being proposed. Therefore, the consistency score has two main phases: (1) score aggregation, which combines the individual scores and knowledge graph frequencies into a single triple candidate score, and (2) score refinement, which iterates over the candidate triples and boosts triples that improve the consistency of the entity types and properties being suggested in relation to the whole candidate data model.

Score Aggregation

In the score aggregation phase, for each triple pattern in the input data, we obtain the individual content and interoperability scores of each element and combine them with their co-occurrence scores extracted from the KGP map constructed (see Section 3.5). The co-occurrence score corresponds to the frequency in which elements of the triple co-occur in a triple pattern of the KGP. We compute the co-occurrence frequencies of the candidate triples ⟨domain – property ⇾ range⟩ dpr and the pairs domain-property dr, domain-range dr, and property-range pr. Therefore, the score aggregation is computed for each input triple pattern ⟨d – p ⇾ r⟩ in the input dataset by considering the individual scores CS_e and IS_e of the element e, where e is an element of the triple pattern candidate $\{c_d, c_p, c_r\} \Leftrightarrow c \in C$. The individual scores CS_e and IS_e are obtained with the specific methodologies per resource described in sections 4.4.1 and 4.4.2.

Overall, we normalise all scores using the following:

$$\widetilde{s}(e) = \frac{s(e)}{\max\limits_{e \in E} s(e)} \tag{4.13}$$

Where E is the set of all elements for which $s(e)$ can be computed. Whenever the tilde character ($\tilde{}$) is used, it represents a value normalised by the maximum of the set.

We then compute the mean of the co-occurrences of the triple patterns co_r with:

$$co\_mean(d, p, r) = mean(\widetilde{co}_{dpr}, \widetilde{co}_{dp}, \widetilde{co}_{pr}, \widetilde{co}_{dr}) \tag{4.14}$$

Algorithm 4.8 shows the complete methodology to compute the score aggregation phase of the consistency score. See Appendix C.8 for a hypothetical example of the application of the algorithm over the candidates generated and ranked with the previous algorithms.

In Algorithm 4.8, first we define the general scoring function that is used to initialise the aggregated score and is also used in the score refinement phase to recompute the aggregated score after refinement.

From line 8 starts the description of the aggregation phase computation, which begins with extracting the individual scores of each of the candidates that match the triple pattern. The algorithm then computes the Cartesian product between the three sets of candidates (line 19), and obtains the co-occurrence score co, which is used, in addition to the individual score, to compute the final aggregated score agg. The final step uses the function $sortby(C, agg)$ to sort the candidates in descending order of agg score per d, p, r key in the map.

Score Refinement

In the score refinement phase, we iterate over all candidate triples that share elements, i.e., the same input subject, predicate, or object, and adjust the individual score of the overlapping resource by boosting or penalising it depending on its ranking in the different overlapping triples. We use the Borda score (Section 4.4.1) to consider the rankings of the candidates when refining the score. For example, considering two triple patterns in

Algorithm 4.8: Consistency Score - Aggregation algorithm

```
Input: KGP: mapping of triple patterns $\langle d, p, r \rangle$ to its frequency in the
       knowledge graph
Input: S: $\langle CS_{et}, CS_{dp}, CS_{op}, IS_{et}, IS_{dp}, IS_{op} \rangle$, where $CS_r$ are outputs of
       Algorithm 4.4, 4.5, and 4.6, respectively, and $IS_r$ are outputs of 4.7
Input: W: scoring weights $\langle w_{cs}, w_{is}, w_{cns} \rangle$
Output: DM: mapping of triple pattern $\langle d, p, r \rangle$ to a tuple that includes $d, p, r$
        candidates, their individual scores $cs_r$ and $is_r$, the triple co-occurrence
        score $co$, and the final aggregated score $agg$
Function scoring($cs_d, is_d, cs_p, is_p, cs_r, is_r, co, W$):
    $ds \leftarrow pow(cs_d, w_{cs}) \cdot pow(is_d, w_{is})$
    $ps \leftarrow pow(cs_p, w_{cs}) \cdot pow(is_p, w_{is})$
    $rs \leftarrow pow(cs_r, w_{cs}) \cdot pow(is_r, w_{is})$
    return $co \cdot w_{cns} + mean(\widetilde{ds}, \widetilde{ps}, \widetilde{rs}) \cdot (1 - w_{cns})$
end

$DM \leftarrow \{\}$
for key $(d, p, r)$ in KGP do
    // extract individual scores
    $C_d \leftarrow \{\langle c_d, cs_d, is_d \rangle \mid c_d, cs_d \in S.CS_{et}[d] \wedge c_d, is_d \in S.IS_{et}[d]\}$
    if $literal(o)$ then
        $C_p \leftarrow \{\langle c_p, cs_p, is_p \rangle \mid c_p, cs_p \in S.CS_{dp}[p] \wedge c_p, is_p \in S.IS_{dp}[p]\}$
        $C_r \leftarrow \emptyset$
    else
        $C_p \leftarrow \{\langle c_p, cs_p, is_p \rangle \mid c_p, cs_p \in S.CS_{op}[p] \wedge c_p, is_p \in S.IS_{op}[p]\}$
        $C_r \leftarrow \{\langle c_r, cs_r, is_r \rangle \mid c_r, cs_r \in S.CS_{et}[r] \wedge c_r, is_r \in S.IS_{et}[r]\}$
    end
    // compute aggregated score for the Cartesian product of the sets of candidates
    $DM[\langle d, p, r \rangle] \leftarrow [\,]$ if key $\langle d, p, r \rangle \notin DM$
    for $(\langle c_d, cs_d, is_d \rangle, \langle c_p, cs_p, is_p \rangle, \langle c_r, cs_r, is_r \rangle)$ in $C_d \times C_p \times C_r$ do
        $co \leftarrow co\_mean(c_d, c_p, c_r)$
        // normalise each score by its maximum in $C_d \times C_p \times C_r$
        $agg \leftarrow scoring(cs_d, is_d, cs_p, is_p, cs_r, is_r, co, W)$
        $DM[\langle d, p, r \rangle].append(\langle c_d, cs_d, is_d, c_p, cs_p, is_p, c_r, cs_r, is_r, co, agg \rangle)$
    end
end
// sort DM in descending order of agg score
for key-value pairs $(\langle d, p, r \rangle, C)$ in DM do
    $DM[\langle d, p, r \rangle] \leftarrow sortby(C, agg)$
end
```

the input data: [`pgterms:ebook`, `dcterms:creator`, `pgterms:agent`] and [`pgterms:ebook`, `dcterms:publisher`, `literal`], the refinement phase would look at the rankings of the candidates for `pgterms:ebook` in both triples and adjust the score according to the rankings of the entity types in both triples. This is an iterative process that runs until the data model converges or up to a maximum of n iterations.

Algorithm 4.9: Consistency Score - Refinement algorithm

```
Input: DM: output of Algorithm 4.8
Input: $W$: scoring weights $\langle w_{cs}, w_{is}, w_{cns} \rangle$
Input: $m$: maximum number of iterations
Output: DM: re-ordered input mapping DM
$n = 0$
$DM' \leftarrow \{ \}$
while $n < m$ or $DM' \neq DM$ do
    $DM' \leftarrow DM$
    for key-value pairs $(\langle d, p, r \rangle, C)$ in DM do
        // compute the Borda score for each candidate of each triple element
        $B \leftarrow \{\}$
        for $e$ in $\{d, p, r\}$ do
            $C_e \leftarrow \mathrm{groupby}(DM, e)$
            $C_e \leftarrow [c.c_e \mid c \in C_e]$
            for $c_e$ in $C_e$ do
                $B[\langle e, c_e \rangle] \leftarrow \mathrm{borda\_score}(c_e, C_e)$
            end
        end
        // refine the individual scores by multiplying them by their Borda score
        $DM[\langle d, p, r \rangle] \leftarrow [\,]$
        for $c$ in $C$ do
            for $e$ in $\{d, p, r\}$ do
                $c.cs_e \leftarrow c.cs_e \cdot B[\langle e, c_e \rangle]$
                $c.is_e \leftarrow c.is_e \cdot B[\langle e, c_e \rangle]$
            end
            // normalise each score by its maximum in C
            $c.agg \leftarrow \mathrm{scoring}(c.cs_d, c.is_d, c.cs_p, c.is_p, c.cs_r, c.is_r, c.co, W)$
            $DM[\langle d, p, r \rangle].\mathrm{append}(c)$
        end
    end
    // sort DM in descending order of agg score
    for key-value pairs $(\langle d, p, r \rangle, C)$ in DM do
        $DM[\langle d, p, r \rangle] \leftarrow \mathrm{sortby}(C, agg)$
    end
    $n \leftarrow n + 1$
end
```

Algorithm 4.9 shows the complete methodology to compute the score refinement phase of the consistency score. See Appendix C.9 for a hypothetical example of the application of the refinement algorithm over the candidate data models DM ranked in Algorithm 4.8.

First, the algorithm computes the Borda score for each candidate of each element. For that, it uses the function $\mathrm{groupby}(DM, e)$ to group key-value pairs in DM, where the element e is a specific component of the $\langle d, p, r \rangle$ key. For example, $\mathrm{groupby}($DM, `pgterms:ebook`$)$ gets all the triple patterns and their candidate lists that have the subject `pgterms:ebook`. The Borda score is then used to refine the individual scores of each candidate and the function $scoring()$ computes an updated aggregated score agg. Finally, we re-order the candidates in descending order of the new agg score. This process is repeated until the refinement does not change the order of the DM mapping anymore or until m iterations are reached.

4.5 EXPERIMENTS

The purpose of the following experiments is not only to evaluate the components that can be evaluated but also to demonstrate the potential and applicability of the framework. Even though knowledge graph annotation and matching is not an uncommon challenge in the Semantic Web field, to the best of our knowledge, no ground truth exists that would allow us to fully evaluate the potential of our proposed scoring methodologies. As previously mentioned in the Related Work sections, our framework differs from common approaches that aim to find the most precise matches for entities and schema of the knowledge graph. These approaches are evaluated with manually constructed ground truths (e.g., [67, 161, 180]) using precision, recall, and F-measure of the proposed entity and schema mappings against a manually curated set of mappings. However, our framework does not aim to find the most accurate candidate, as it instead focuses on finding a suitable data model to fit a whole dataset while maximising integration with multiple data sources.

Following a closed-world assumption for the knowledge graph, i.e., the data models in the knowledge graph as correct and no other data model is known, our framework aims to rank full candidate data models according to a set of parameters that modify the weights of the different framework components for the ranking of the data model recommendations. Nonetheless, the first components of the framework (generation and content scoring) include generation and ranking of candidates based on their accuracy. Therefore, we evaluate these components using available ground truths with methods that mimic the application of the components in the context of the framework. The interoperability and consistency scores, however, re-rank these candidates according to parameters that are currently not easy to evaluate. Therefore, for these components we present experiments that showcase their overall impact in the recommendations of the framework.

In the next sections, we first present the ground truths used to evaluate the generation and content ranking components. Then we describe the evaluations performed and their results. Finally, we define and discuss the different experiments implemented to test the interoperability and consistency components.

4.5.1 Ground Truths

For the generation and content score components, we use the following ground truths:

1. DBpedia as knowledge graph using the ground truths from the 2019 SemTab Challenge [180]

2. DBkWik [158] datasets as knowledge graph using the ground truth from the Knowledge Graph track of OAEI [67]

3. Library linked data as knowledge graph and ground truth extracted from `owl:sameAs` links

SemTab

Recently, the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) was started to benchmark systems dealing with the problem of annotating entities, entity types, and object properties[3]. The 2019 and 2020 editions of this challenge included three tasks: (1) assigning an entity type to a column, named Column Type Annotation (CTA), (2) matching an entity in the tabular data with an entity in the knowledge graph, called Column Entity Annotation (CEA), and (3) assigning a knowledge graph property to the relationship between two columns, which is equivalent to the object property generation and ranking task, called Column Property Annotation (CPA).

In our experiments, we use the datasets of the Round 4 of the challenge and the ground truths made available after the 2019 edition of SemTab [181]. These ground truths include mappings of tabular data to entities of the English DBpedia, and DBpedia ontology classes and properties. Therefore, we retrieved the labels, instance types, literal, and object mappings from the DBpedia Databus version of 2020.07.01[4] and loaded the data using the processes described in Section 3.6.

DBkWik

For this dataset, we use the data extracted for the DBkWik knowledge graph [158] and made available for the OAEI 2019 Knowledge Graph track[5]. The data was extracted from pages in the Fandom Wikifarm[6] for three Star Wars-related wikis and three Star Trek wikis. The Knowledge Graph track also includes a Marvel-related dataset. However, we are looking for cases similar to ours which include a knowledge graph with several data sources. Since the Marvel ground truth only includes 1-to-1 mappings, we exclude it from these experiments. Therefore, we use the Star Wars (SW), Star Wars: The Old

[3] `http://www.cs.ox.ac.uk/isg/challenges/sem-tab/` (Accessed in September 2020)
[4] `https://databus.dbpedia.org/dbpedia/`
[5] `http://oaei.ontologymatching.org/2019/knowledgegraph`
[6] `https://www.fandom.com`

Republic (SWTOR), and Star Wars: Galaxies (SWG) Wikis, and the Memory Alpha (MA), Memory Beta (MB), and Star Trek Expanded (STE) Universe Wikis.

We use the SWTOR and SWG as the Star Wars knowledge graph, and the MB and STE as the Star Trek knowledge graph. Both SW and MA are used as the input data since they are the most likely to overlap with the resources in the knowledge graph data sources and, therefore, better emulate the application of our framework.

`owl:sameAs` Dataset

As an alternative to a curated ground truth, we created an evaluation dataset extracted from `owl:sameAs` links between the RDF library datasets. Four of the chosen libraries (British, French, German, and Spanish) include `owl:sameAs` links to external resources. Therefore, we created an automatic ground truth by selecting documents that linked to the same third party resource in all four libraries and extracted the types of each document. In this case, these four libraries overlap when creating an external link to VIAF (Virtual International Authority File), which is a service that combines authority names from multiple libraries under a single domain. For example, `bnb:person/CaroDugoCarmen1963-`, `bnf:12148/cb150440479#about`, `gnd:173023355`, and `bne:XX1379294` all contain a `owl:sameAs` link to `http://viaf.org/viaf/47047050`. Therefore, we store in the `owl:sameAs` ground truth the entities and their entity types as equivalent to each other.

Using this methodology, we obtain a ground truth with 26 009 overlapping person entities between the four libraries.

Descriptive Statistics

Table 4.1 contains descriptive statistics of the knowledge graphs used to evaluate the initial components of the framework, including number of entities and distinct entity types, datatype, and object properties. The number of triples represents the number of triple patterns in the form of ⟨`domain` – `property` ⊸ `range`⟩.

Table 4.1: Ground truth knowledge graphs statistics

Knowledge Graph	# Entities	# Entity Types	# Dt Properties	# Obj Properties	# Triples
DBpedia	17 721 495	397	482	412	94
DBkWik: Star Wars (SWTOR+SWG)	37 590	172	393	154	547
DBkWik: Star Trek (MB+STE)	127 149	527	268	388	656
Library data (British/French/German)	105 729 880	72	363	806	1169

Table 4.2 shows descriptive statistics for the input data used to match each of the knowledge graphs of Table 4.1. The SemTab DBpedia-based dataset is in CSV format, with no schema, therefore, no *a priori* extraction of entity types and properties is possible.

The dataset has 817 CSV tables with an aggregated total of 52 066 rows. The remaining input datasets are in RDF format and, therefore, the complete statistics are presented in the table.

Table 4.2: Ground truth input data statistics

Knowledge Graph	# Entities	# Entity Types	# Dt Properties	# Obj Properties	# Triples
DBpedia: SemTab (rows)	52 066	-	-	-	-
DBkWik: Star Wars (SW)	335 666	273	231	483	714
DBkWik: Star Trek (MA)	143 838	185	181	160	341
Library data (Spanish)	20 752 087	7	163	38	201

Finally, Table 4.3 presents descriptive statistics of the ground truth provided for each dataset. The SemTab ground truths are separated into their tasks. The CEA task contains correspondences for 107 352 entities. The CTA ground truth contains mappings for 823 columns over 817 CSV files. These mappings contain 158 unique entity types represented. The CPA ground contains 216 unique object properties represented between 825 columns over the same 817 CSV files. The DBkWik ground truths include 1-to-1 equivalence mappings of entities, entity types, and properties for each pair of files. Finally, the `owl:sameAs` ground truth includes a 1-to-many mapping between entities in the Spanish library and the three libraries in the `owl:sameAs` knowledge graph.

Table 4.3: Ground truth statistics

Ground Truth	# Entities	# Entity Types	# Properties
DBpedia: SemTab CEA	107 352	-	-
DBpedia: SemTab CTA	-	158	-
DBpedia: SemTab CPA	-	-	216
DBkWik: Star Wars (SW-SWG and SW-SWTOR)	2454	19	76
DBkWik: Star Trek (MA-MB and MA-STE)	11 021	27	55
Library data (Spanish - British/French/German)	26 009	7	201

4.5.2 Candidate Generation

As previously mentioned, the goal of the candidate generation component of the framework is to generate an exhaustive list of candidates to match entity types, datatype, and object properties between the input data and the knowledge graph. Therefore, we evaluate this component in terms of recall of relevant entities, entity types, and properties.

Entity Types

Entity type candidate generation is evaluated in terms of entity and entity type recall. For entity recall, we use the SemTab (CEA ground truth), DBkWik, and `owl:sameAs` ground truths and compute the recall for entities in the ground truth. Simultaneously, we examine how the number of results per search query, i.e., the search window size, affects the recall to analyse its effect on the candidate generation results.

In terms of entity type recall, we use the same datasets as for entity recall (using the CTA ground truth for SemTab). Here we analyse the recall of entity types, which can be different from the entity recall since even when the wrong entity is retrieved, due to similarities in the name, its entity type can still be correct. Since the goal of the candidate generation is to obtain an exhaustive list of candidates, entity retrieval accuracy is not essential to obtain reasonable results. Simultaneously, we analyse the impact of the input data sample size in the recall of entity type candidates.

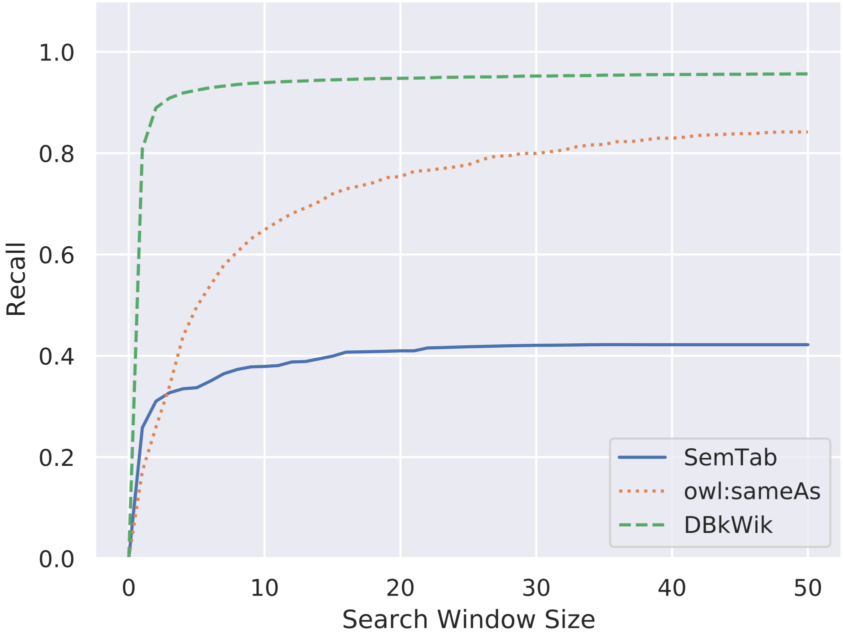


Figure 4.3: Entity Recall

Figure 4.3 shows the result of the recall and search window evaluation for the three datasets under consideration. For each dataset, the maximum number of mappings per entity is three for the `owl:sameAs` dataset, two for the DBkWik dataset, and for the SemTab dataset it varies between 1 and 209. However, these mappings include several variations or deprecations of the same entity URI. Therefore, the lower performance results are negatively influenced by the extensive list of results that is present for some entity candidates, and is not a trivial to distinguish true low recall from deflated results due to the ground truth. The remaining datasets obtain high recall, maximising between 80-90% for the largest search window tested.

The x-axis of Figure 4.3 shows how the size of the search window affects the generation process. For the SemTab dataset, around window size $w = 10$, there are no significant changes. Considering that the mean number of mappings per entity is 4.93 with a median of 2, on average, doubling the size of mapping obtains close to maximal recall. For the DBkWik, there is a mean of 1.1 mappings per entity. Therefore, the recall is close to the maximum with a search window of 2-3, which also represents a close double of the number of mappings per entity.

The `owl:sameAs` dataset contains the most heterogeneous data since it has data in different languages, therefore, it is the one that benefits the most from a larger search window. In this dataset, all entities contain three correct mappings (one for each library). Despite the different languages, the generation algorithm manages to obtain a high recall, with a search window of 15, already obtaining around 70% recall. Since this is the dataset that more closely resembles the data we are using in the chosen use-cases, for the remainder of the experiments, we set the search window to 15.

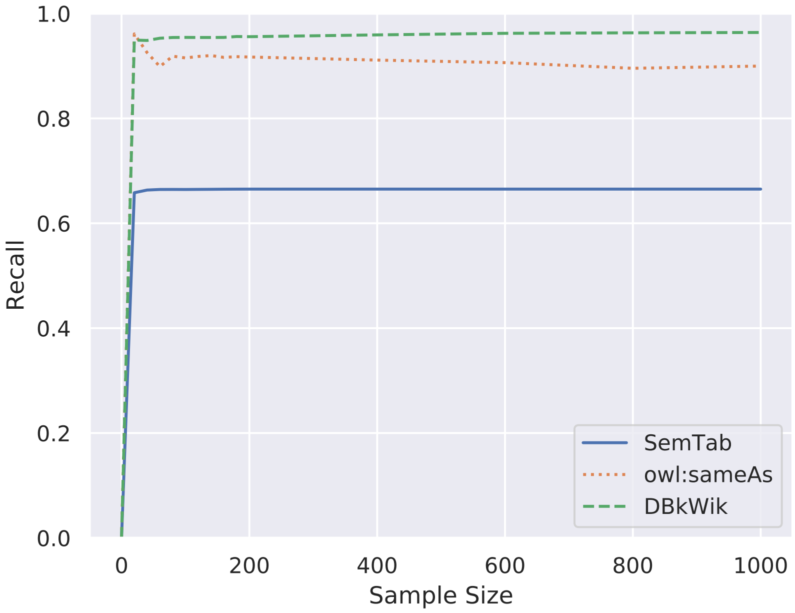


Figure 4.4: Entity type recall

Figure 4.4 shows the result of the entity type recall and the sample size for the three ground truths considered. For the `owl:sameAs` and DbkWik ground truths, the entity type recall was high even with a small sample. For the `owl:sameAs` ground truth, this result is explained by the reduced number of entity types featured in the dataset (see Table 4.3). Therefore, even a small sample of entities is sufficient to obtain all the correct entity types. For the DbkWik ground truth, the entity type results are in line with the entity recall since the correct entities are being retrieved. The small impact of the sample size in this dataset is explained by the small number of entities in the ground truth for each type. On average, each entity type in the ground truth has 55.2 entities, with a median of 6. Only 2 entity types have more than 1000 entities, and 27 have more than 100. Therefore, sampling above hundred is mostly meaningless for this dataset.

The SemTab evaluation against the CTA ground truth obtained a lower performance, which can be explained by the way the ground truth is constructed: It includes the most precise entity type for each entity but also includes the full hierarchy of that entity type. Since this ground truth was not designed to evaluate recall, the results are not optimal. Nonetheless, the ground truth has a mean of 3.46 entity types per entity, with a median of 3.

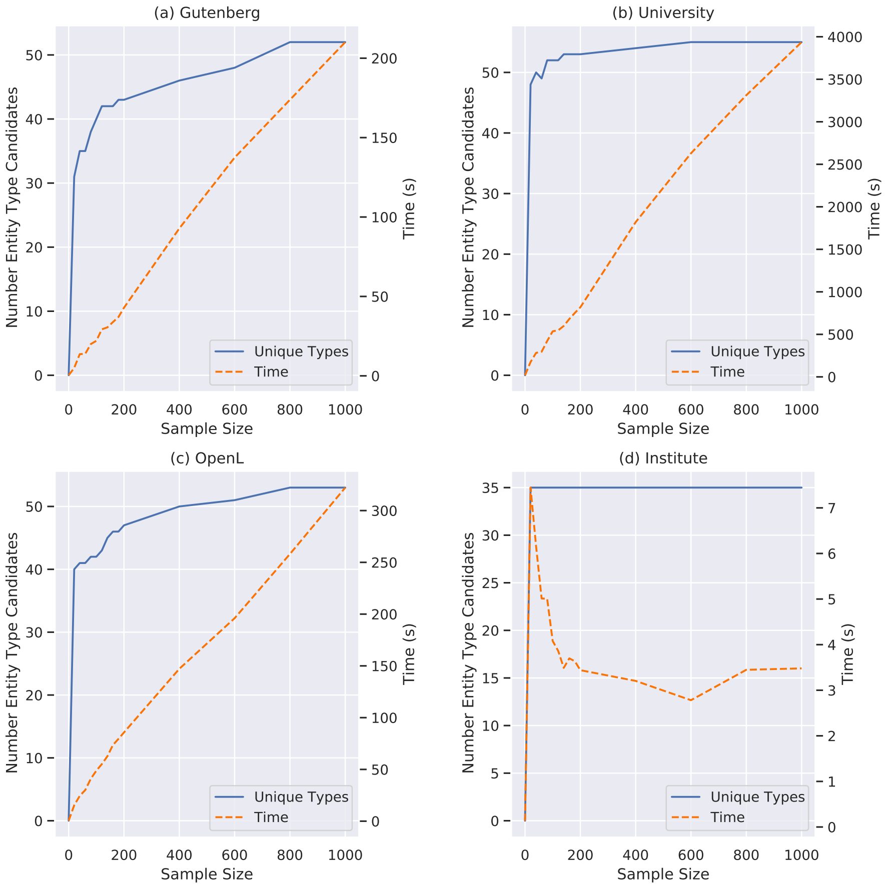


Figure 4.5: Unique types generated

Overall, due to their specific characteristics, none of the ground truths used was able to fully evaluate the impact of the sample size in the entity type generation methods. Therefore, Figure 4.5 shows how the sample size affects the entity type candidate generation methodology over the library use-case. In this experiment, we run the entity type generation for each of the input library datasets and use different sample sizes to generate candidates. The *Unique Types* line represents the total number of unique entity type candidates that are retrieved with sample size s. The *Time* dashed line shows the time the algorithm takes to retrieve these candidates with sample size s[7]. The results seen in the figure show that the computation time increases linearly with the sample size, which is consistent with the for-loop described in line 5 of Algorithm 4.1. After sample size 300, approximately 10 new unique types are discovered. However, there is a difference of $\approx$ 150 seconds between sample size 200 and 1000 for Gutenberg and $\approx$ 40 minutes for the University dataset, with a difference of less than 10 new unique candidates. The Institute dataset only has 34 entities, therefore, the sample size does not affect the number of new candidates found after this value.

Therefore, by analysing the recall results presented in conjunction with the results shown in Figure 4.5, we select a sample size of 300 for the remaining of the experiments.

Datatype Properties

The results of the datatype property generation include all properties of the knowledge graph ordered by their probability of matching according to the classification models. Therefore, if the correct property can be found in the knowledge graph, it should appear in the datatype property candidate generation results.

To confirm, we compute the recall of the generation method by using the DBkWik datasets, which are the only ones that include datatype properties in their ground truths. Therefore, to analyse the datatype property candidate generation, we train classification models for these datasets as explained in Section 3.9. To accelerate the training process, we run a random grid search but skip the exhaustive grid search. We train the models with the parameters presented in Table 4.4.

Table 4.4: Hyperparameter search

Hyperparameter	**Randomised**	**Final**
n_estimators	[100, 1000]	500
criterion	entropy	entropy
min_samples_split	[0.01, 0.40] step = 0.01	0.03
min_samples_leaf	[0.01, 0.40] step = 0.01	0.02
max_features	[2, 7]	4
min_impurity_decrease	[0.0, 0.2] step = 0.01	0.03
bootstrap	False	False

We obtained a recall of $\approx$ 95% using this evaluation method. The recall is below 100% due to the fact that the ground truth features mappings between object properties and datatype properties. Our framework strictly distinguishes datatype and object properties, therefore, a property cannot be

[7] Executed in an 8-core Intel Xeon CPU @ 2.40GHz with 48GB of RAM running Ubuntu 16.04.2 LTS

both. When the framework finds a property which is both a datatype and a object property, it assigns it to the object properties. For example, the property `http://dbkwik.webdatacommons.org/starwars.wikia.com/property/title` in the SW dataset is mapped as equivalent to `http://dbkwik.webdatacommons.org/swg.wikia.com/property/title` in SWG and to `http://dbkwik.webdatacommons.org/swtor.wikia.com/property/title` in SWTOR. However, in SWTOR this property is both a datatype property and an object property, while in SWG it is only a datatype property. This pattern is found for 345 properties over all the DBkWik datasets. Since our framework strictly distinguishes datatype and object properties, the property in the ground truth mapped to SWG is not discovered since it is considered as an object property.

Object Properties

The generation of candidates for object properties depends on the generation of candidates for entity types and ranking using content score (see Algorithm 4.3). Therefore, the performance of this generation step is directly correlated with the performance of the entity type generation and content score ranking. This generation step extracts an exhaustive list of all edges between two vertices in the ontology graph. Since currently, the framework is not able to identify object properties unless they are explicit in the input dataset (e.g., a CSV featuring a column of unique identifiers mentioned in another CSV file), we are not able to use the SemTab ground truth to evaluate this step.

Therefore, to evaluate this generation step, we instead use the algorithm to generate candidates for the object properties in the DBkWik ground truth. We compare the results of the ground truth with the candidates returned in terms of recall. For the object property generation step, we obtain a recall of $\approx$ 50% using the DBkWik: Star Wars ground truth and $\approx$ 80% using the DBkWik: Star Trek ground truth. Similarly to the datatype property generation, the object property generation is degraded by overlapping mappings between datatype and object properties in the ground truth, which the framework does not allow.

4.5.3 Content Score Ranking

Content score ranking consists in ordering the list of candidates generated to fit various measures that focus on finding the most suitable candidates to match an input resource.

We start by comparing the results of our content scoring ranking to the results obtained by the tools that participated in the OAEI 2019 Knowledge Graph track. For this, we used the datasets mentioned in Tables 4.1-4.3. Additionally, to make the results comparable, we used the third dataset (Marvel dataset) excluded from the previous and following experiments.

Table 4.5 presents a summarised version of the results from the OAEI track, including the maximum score obtained with the average (in brackets) of all participating tools that obtained a result greater than 0.0 and excluding the baselines.

System	Resource	Precision	Recall
OAEI Systems	Instance	0.91 (0.70)	0.86 (0.64)
	Properties	1.0 (0.83)	0.98 (0.68)
Ours	Instance	0.67	0.61
	Properties	0.19	0.35

Table 4.5: Summarised results of the OAEI Knowledge Graph track in comparison to our framework

Overall, our framework obtains lower precision and recall when compared with the systems that participated in the OAEI 2019 Knowledge Graph track. Despite this result, in terms of instance mappings precision, our framework is able to obtain results similar to the average system that participated in the Knowledge Graph track. In relation to property mapping, our framework obtains significantly lower performance. However, in contrast to OAEI tools, we propose a framework that considers all parts of a data model to propose a consistent data model that potentially integrates with several other data models. Therefore, our focus when developing the generation and content score methodologies is to enable the recommendation of complete data model candidates, instead of focusing on specific performance measures.

The main focus of the systems that participate in the OAEI is to improve the performance of their task-specific matchers, therefore, excelling at individual matching tasks of instance and schema of a single knowledge graph. These systems use combinations of string similarity measures, background knowledge exploitation, structural matching between instances in knowledge graphs, or reasoning over ontologies to obtain their results. In contrast, our system focuses on integrating several RDF knowledge graphs, instead of knowing *a priori* which target is more suitable to each input resource. The number of knowledge graphs being integrated increases the complexity of the matching task and, therefore, the complex combination of algorithms these systems use is not directly applicable to the task handled by this thesis. In addition, as previously mentioned, our system strictly differentiates between datatype and object properties, which further hinders performance when using the DBkWik dataset.

Therefore, in the next sections, we experiment and evaluate different aspects of the scoring algorithms to evaluate the impact of the different methods in the ranking of candidates. Since we are looking for the best ranking of candidates, we evaluate candidates with precision at k (P@k).

Entity Types

To evaluate and analyse the methodology of content scoring of entity types, we test different string similarity measures and simultaneously analyse the impact of adding a threshold to exclude search results. Then we evaluate each one of the metrics in the content score methodology and discuss their influence in the overall final content score of an entity type candidate.

STRING SIMILARITY AND THRESHOLD We tested different string similarity measures and the ones with the best performances for our use-cases and evaluation test datasets were n-grams, Jaccard similarity coefficient, and Levenshtein similarity. n-grams were explained previously in Section 4.4.1. The Jaccard similarity coefficient measures the similarity between two sets by calculating the quotient between their intersection and their union. The Levenshtein similarity is calculated from the Levenshtein distance which calculates the number of edits required to transform a word into another. The Levenshtein similarity between strings a and b is then computed as:

$$sim(a, b) = 1 - \frac{levenshtein(a, b)}{max(length(a), length(b))} \tag{4.15}$$

To evaluate these measures, we computed the precision@ $\{1, 3, 5\}$ using the `owl:sameAs` and DBkWik ground truths. Simultaneously, we analyse the impact of adding a cut-off to the candidate generation results by selecting a threshold for the string similarity between input and candidate entities. Finally, we also analyse the impact of the threshold in the average number of search results returned per query. From these results and through observation of the weaknesses of the similarity measures for the library use-case, we developed the combined n-grams approach described in Section 4.4.1. We compare these similarity measures against the list of search results ordered by a baseline computed from the ElasticSearch min-max normalised score per search query.

Figures 4.6 and 4.7 show the results of these experiments for the `owl:sameAs` and DBkWik ground truths, respectively. In both figures, the performance is comparable between similarity measures tested. Through empirical analysis, we found that for the library use-case, which features long literals, the combined n-grams strategy managed to positively rank some of the cases and due to the weighted combination of word and letter n-grams, non-corner cases were not negatively affected. Since both ground truths include 2 and 3 correct results, for the remaining of the ground truth evaluations we will use only precision@3. Plot (d) in both figures shows the number of search results left after the threshold is applied. Using a search window of size 15, between threshold 0.5 and 0.6, most similarity measures have reduced the number of results to a half.

We also evaluate precision in terms of the entity types that are correctly ranked higher, even when the entity is incorrect. Figure 4.8 shows the results of this evaluation for the `owl:sameAs` and DBkWik datasets. Again string similarity measures per-

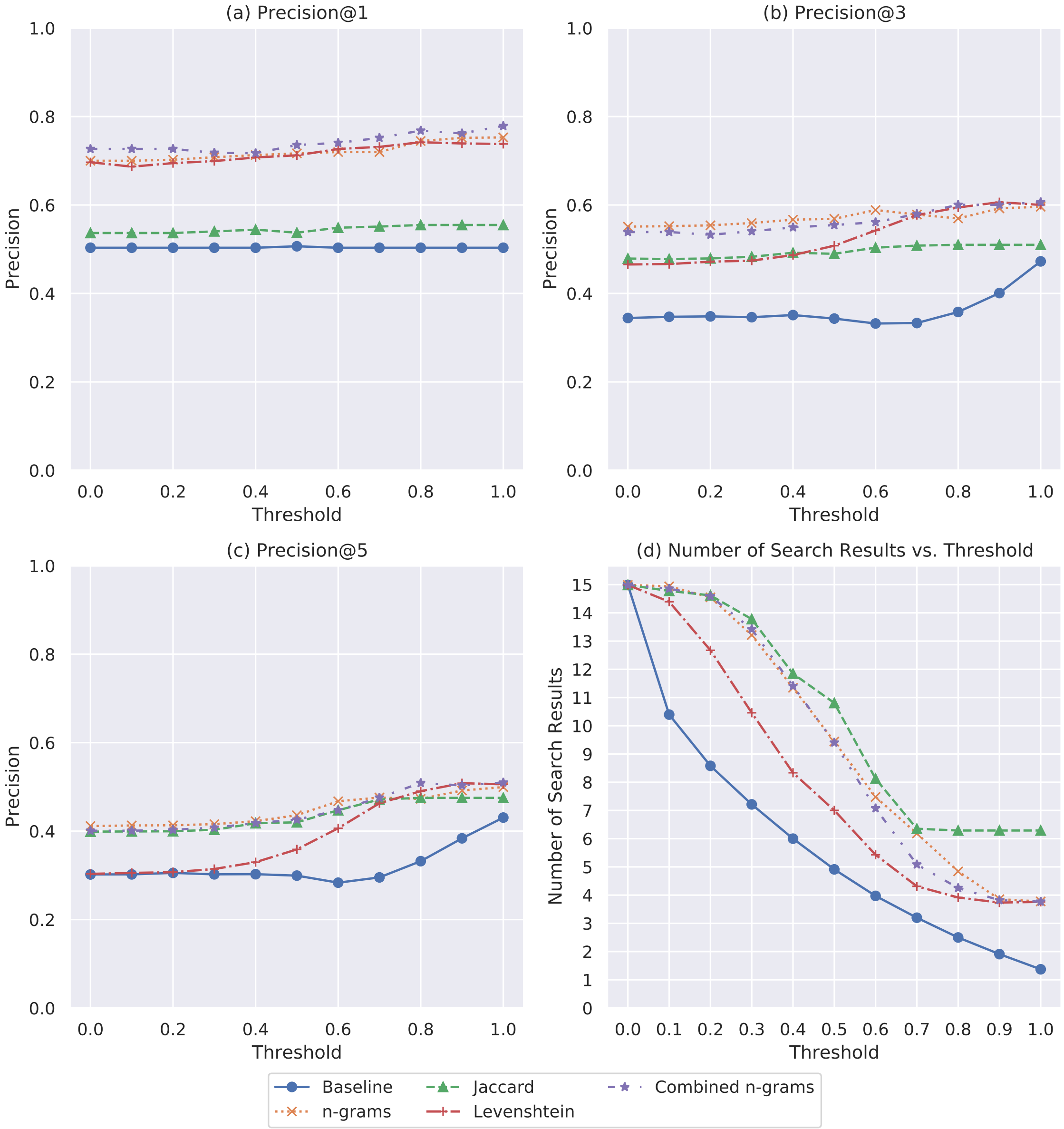


Figure 4.6: Similarity strategy and threshold evaluation for entities in the `owl:sameAs` dataset

form similarly, but it is also clear that they behave differently depending on the dataset chosen. For example, Jaccard similarity is the measure with the worst performance for the `owl:sameAs` dataset but has a comparable performance to the other algorithms in the DBkWik dataset. Therefore, the performance of the string similarity measures highly depends on the characteristics of the literals of the input and knowledge graph datasets.

From this evaluation, and through observation of the results in the use-case datasets, we chose the combined similarity measure for the use-cases of our framework. Due to the similar performance between measures, the best strategy is to analyse the input datasets and knowledge graph, and ponder the strengths and weakness of different string similarity measures. For example, Levenshtein compensates similarities between

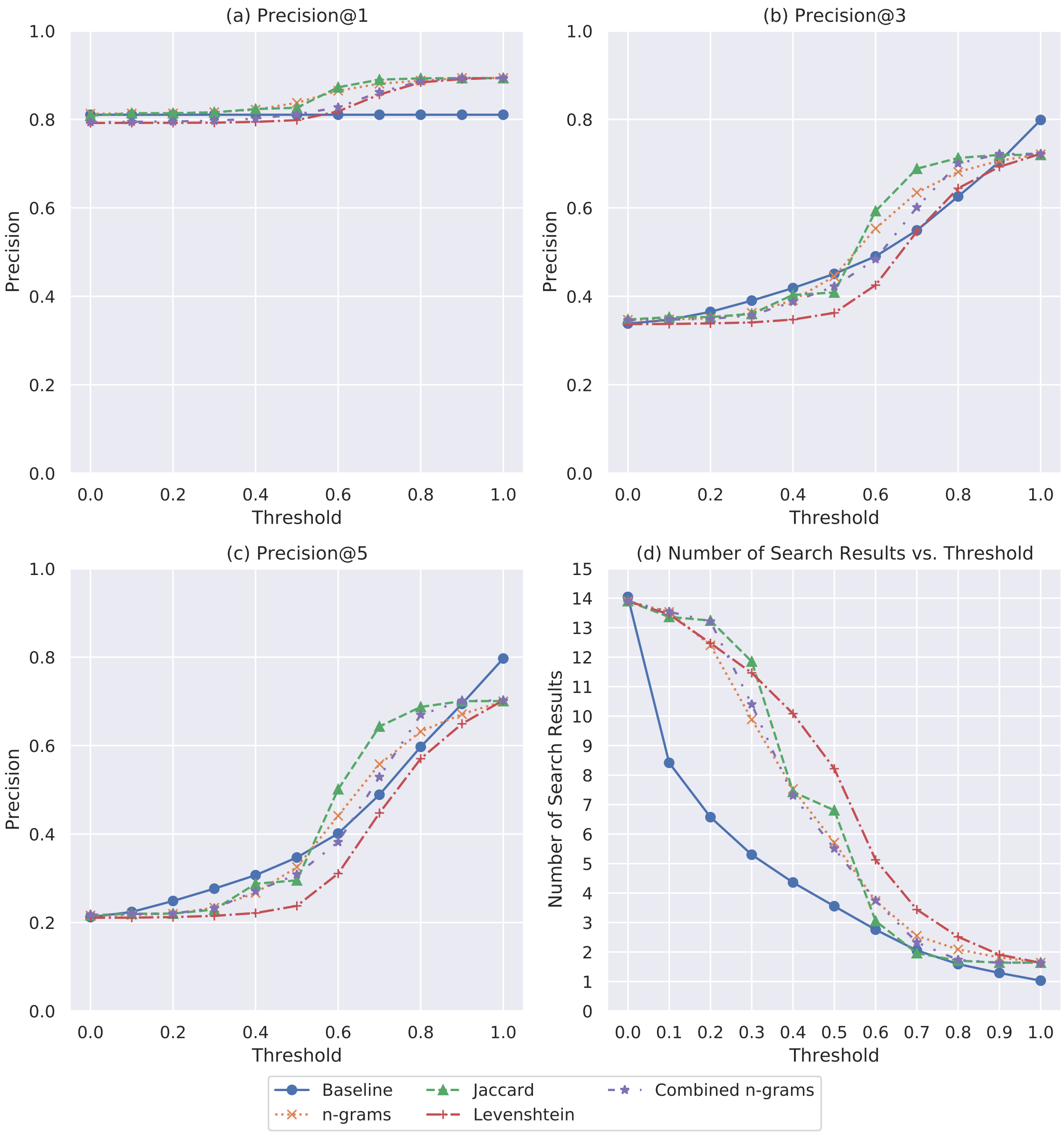
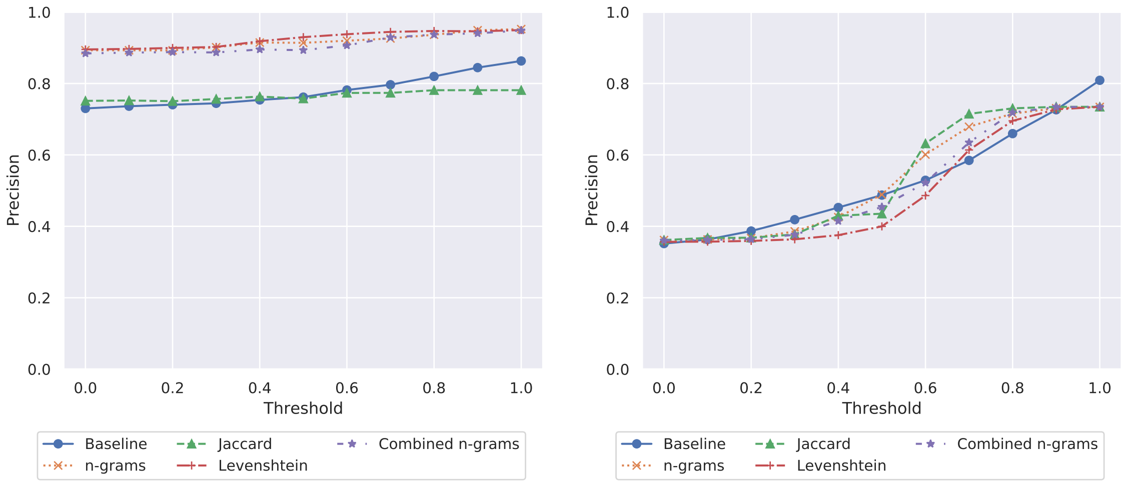


Figure 4.7: Similarity strategy and threshold evaluation for entities in the DBkWik dataset

strings when typos occur because a small edit between two words which leads to high similarity. The chosen combined similarity uses a mixture of different types of n-gram pre-processing with a Jaccard coefficient since the intersection of the n-grams is divided by their union. This approach gives us the similarity between two strings while taking into account the letter or word order in the strings being compared.

From the analyses with entities and entity type precision discussed, we choose a threshold of 0.6 for the remaining experiments since it provides a good balance between high precision and a reasonable number of search results for both entity and entity type candidate rankings.



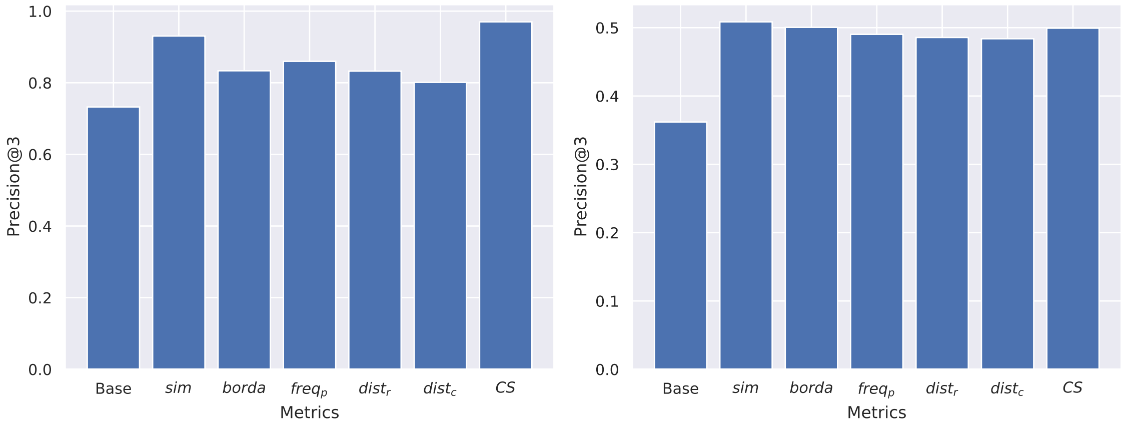
(a) owl:sameAs dataset

(b) DBkWik dataset

Figure 4.8: Similarity strategy and threshold evaluation for entity types

CONTENT SCORE METHODOLOGY The combination of metrics to compute the content score was developed by observing where the re-ordering of candidates was clearly failing and what distinguished an acceptable match from a clearly wrong match (e.g., search for an author and obtaining the type Book). Nonetheless, we evaluated the methodology by calculating the running average of each incremental measure of the content score and computing the precision@3 for the owl:sameAs and DBkWik ground truths over the entity types. We also include the final CS score, which considers the average of all measures and multiplies it by the frequency $\mathtt{freq}_s$ (see Section 4.4.1).

Figure 4.9 shows the results of this evaluation for the owl:sameAs and DBkWik datasets. For these datasets, the measures show no significant improvement to the rankings, with most measures affecting the precision negatively. However, these negative impacts are less than 0.1, which, in large datasets, translate into a small impact in the results.



(a) owl:sameAs dataset

(b) DBkWik dataset

Figure 4.9: Content score methodology evaluation

Due to the characteristics of the ground truths used, which do not perfectly align with the use-cases, we further analyse the measures in the library and biomedical use-case datasets. For that, we apply the same experiment, but instead of traditional precision, we apply *lenient precision*. In this evaluation, instead of binary correct/incorrectness against a ground truth, we measure the shortest distance between input type and candidate entity type as a measure of correctness. We also apply a discount metric DM based on the distance $\mathtt{dist}(s, t)$ between a source vertex s and a target vertex t as follows:

$$\begin{aligned} dist(s,t) &= shortest\_distance(s,t) \\ DM(s,t) &= \frac{1}{\log_2(dist(s,t)+1)} \end{aligned} \tag{4.16}$$

The results of the lenient precision for the library use-case[8] are shown in Figure 4.10. Similarly to the previous results, the lenient precision shows only small changes with the addition of subsequent measures to the average score. Overall, the string similarity $\mathtt{sim}$ and the search result frequency $\mathtt{freq}_s$ are the only ones that have a positive impact in some of the datasets, but, in average, the measures show little impact in top-3 ranked candidates.

However, the aim of the scoring methodologies is not only to allow ranking of the candidates according to appropriateness but also to adjust the difference between the scores obtained by the candidates. Larger differences between the potentially desirable and undesirable candidates are beneficial for the score aggregation phase since it will reduce the likelihood of less desirable candidates being ranked high due to high interoperability or consistency. On the other hand, it is also preferable that desirable candidates obtain similar scores so that they can be better re-ranked later according to their interoperability and consistency. Therefore, in the next experiment, we tested how the content score metrics affect the difference in score between the first and last candidate.

Figures 4.11 and 4.12 show the results of this experiment. The first observation is that the baseline presents the largest difference between top and bottom ranked candidates. This result is due to the min-max normalisation of the ElasticSearch score, which by definition maximises the difference between top and bottom candidates. As expected, the Borda score, the resource frequency proportion $\mathtt{freq}_p$, and the final CS score increase the difference between ranked candidate scores. However, the distance metrics, $\mathtt{dist}_r$ and $\mathtt{dist}_c$, tend to further homogenise the score.

Overall, we conclude that string similarity and frequency-based metrics have the largest impact on increasing the separation of the values of the content score. While distance measures tend to approximate the results more. Through empirical analysis, we verified that the distance metrics had a positive impact in the ranked results. In these cases, the top ranked candidates were found by obtaining a content score significantly higher than the other potentially desirable candidates due to their frequency in the

[8] Similar conclusions are reached in the biomedical use-case

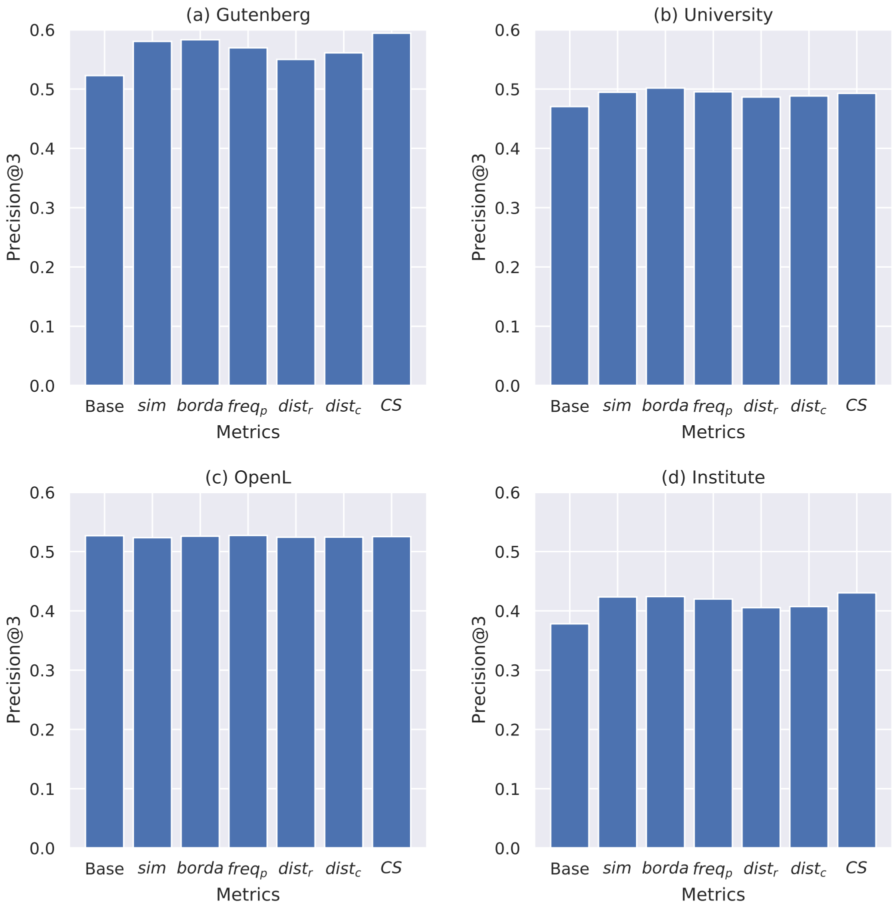
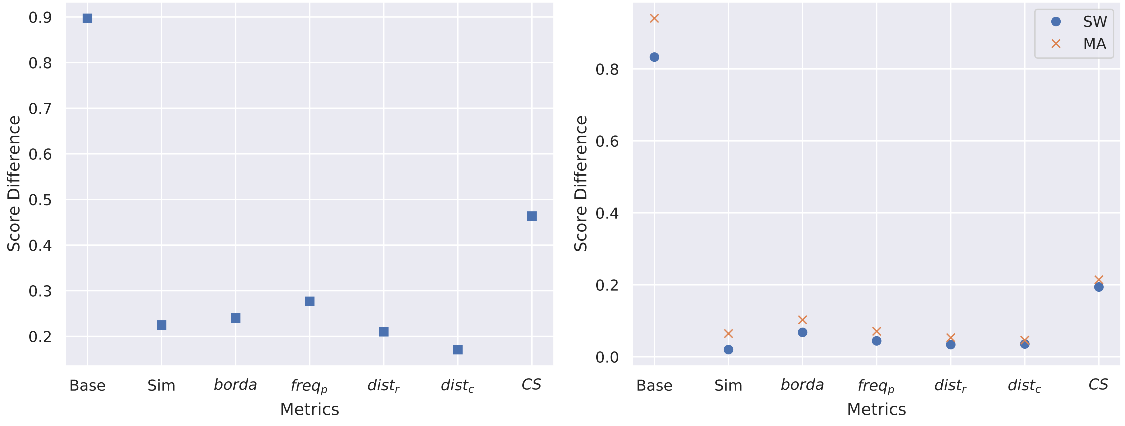


Figure 4.10: Lenient precision results for entity type candidates in the library use-case



(a) `owl:sameAs` dataset

(b) DBkWik dataset

Figure 4.11: Score difference between first and last candidates

search results. Consequently, candidates that were less frequent but still potentially desirable, were boosted by the distance metrics due to their proximity to the input resource in the knowledge graph. Despite not changing the final rankings, these metrics

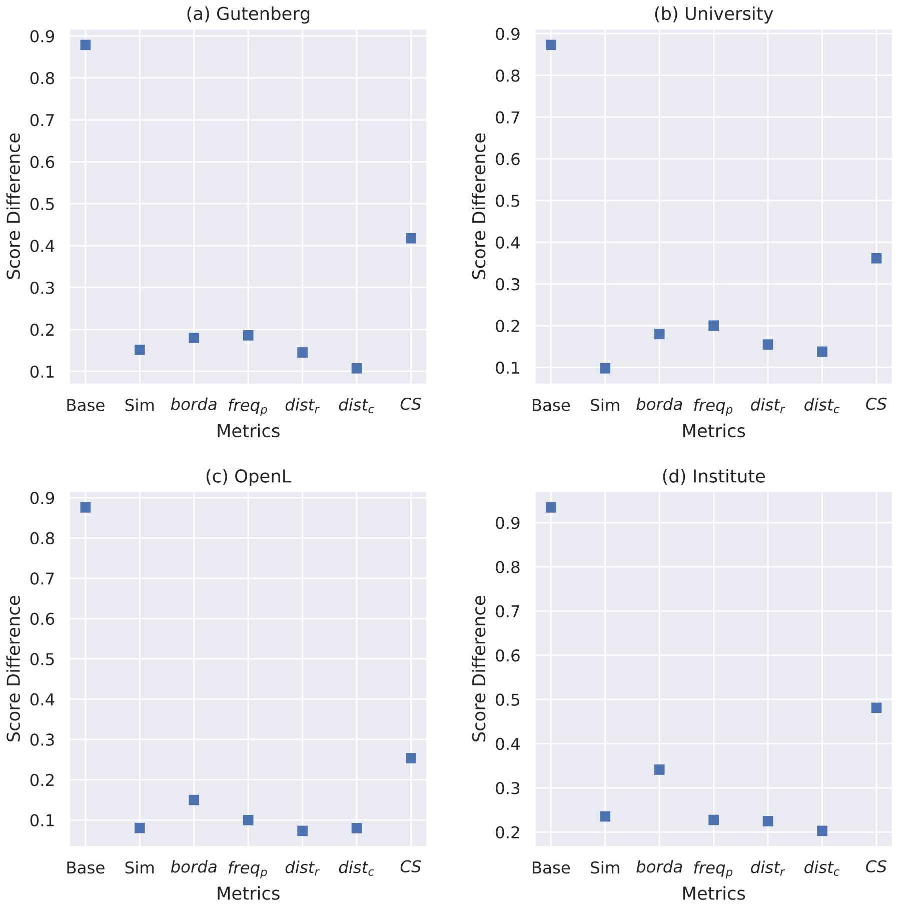


Figure 4.12: Score difference between first and last candidates for the library datasets

are useful to approximate the rankings of candidates that are close to the input and among each other, while further differentiating potentially more distant candidates.

Datatype Properties

We evaluate datatype property content-based ranking with precision@k and recall@k, where $k \in [1 - 10]$. Since precision is a ratio of the total number of retrieved results, the true positive numbers are diluted in the case of datasets like DBkWik which have at maximum two correct answers per query. This recall metric evaluatesif the correct result is in the top-k results presented.

Figure 4.13 shows the precision@k for the content scoring re-ranking of the datatype properties. The figure shows that, using the DBkWik ground truth, the precision remains relatively low (less than 0.4), however, for approximately 40% of the properties in the input, the correct property candidate is in the first 10 candidates after ranking.

Therefore, considering the heterogeneity of the properties found in the DBkWik dataset, with several string-based properties not easily distinguishable, the combination of the random forest classifier generation with the re-order based on the graph metric manages to generate candidates for a reasonable number of input properties. Furthermore,

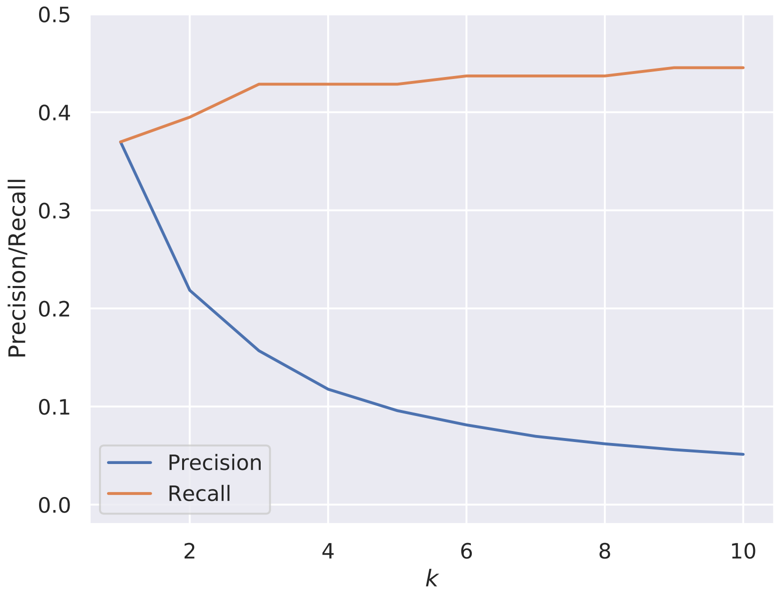


Figure 4.13: Precision and recall for content scoring ranking of datatype properties using the DBkWik ground truth

the DBkWik dataset contains only 41 properties with overlap between the three datasets. Therefore, most properties have only one correct candidate property.

To further analyse the datatype ranking methodology, we again computed the lenient precision for the datatype property ranking. Figure 4.14 shows the results of this evaluation. In these figures, we observe that the impact of the measures on the precision is more noticeable since there is no strong initial measure to compute that is equivalent to the string similarity. Therefore, for datatype properties, the distance $\mathtt{dist}_r$ has the largest positive impact of the precision@3 results.

Nonetheless, even considering lenient precision, the datatype ranking still achieves relatively low results, with all datasets in the library use-case achieving less than 0.4 precision@3.

Object Properties

Similarly to the datatype properties evaluation, we compute precision and recall at different cut-offs.

Figure 4.15 shows similar results to the datatype properties, where approximately 40% of all input properties have a correct match in the top-10 ranked candidates. However, contrary to the datatype properties, the major limitation to performance in object property generation is correct entity type generation and ranking. Increasing the performance of the entity type related methodologies will impact recall of the object type generation, and, consequently, lead to improved precision results.

Therefore, overall, the object property ranking methodology can be strengthened by not only developing the object property ranking methodology but also by improving

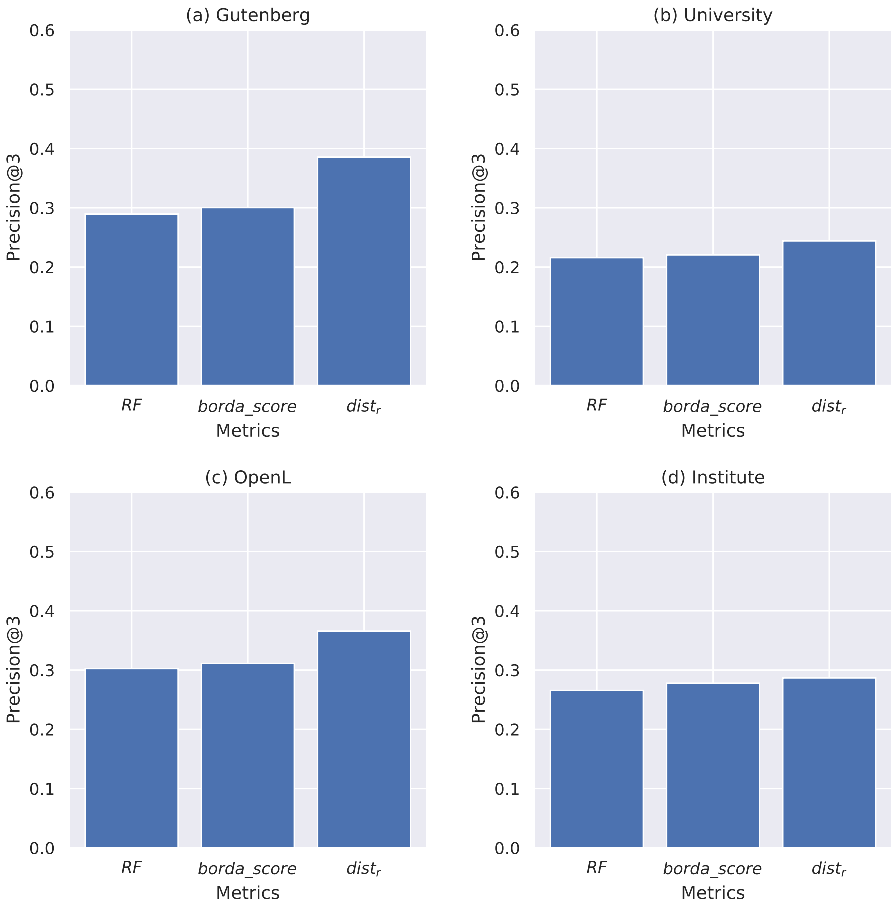


Figure 4.14: Lenient precision results for datatype properties candidates in the library use-case

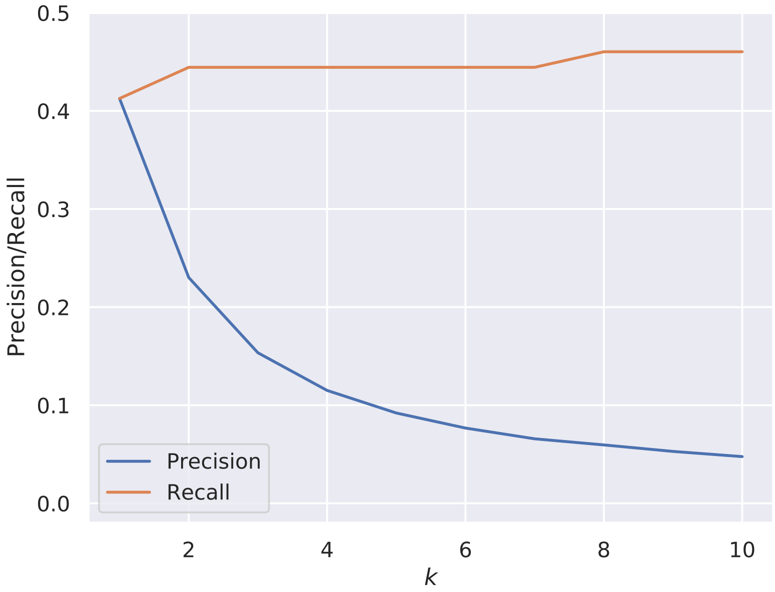


Figure 4.15: Precision and recall for content scoring ranking of object properties using the DBkWik ground truth

the entity type generation and ranking. Nonetheless, for a diverse set of input object properties, with a straightforward methodology, the generation and ranking manages to generate a reasonable set of candidates for the DBkWik ground truth.

4.5.4 Interoperability

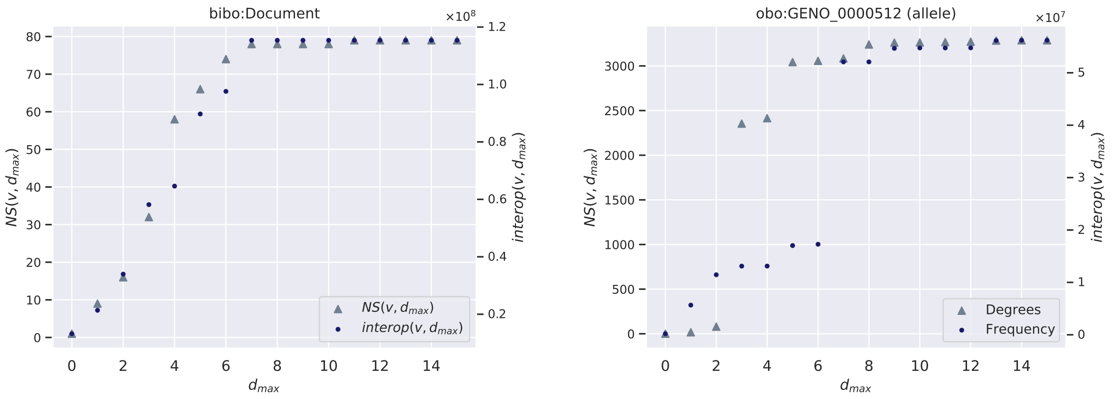
Interoperability focuses on maximising the integration between ontology resources selected for the data model and the different schemas in the knowledge graph. To the best of our knowledge, there is currently no ground truth that can reliably evaluate this component of the framework. Since different data publishers have different interpretations of what the best entity type is for their use-case, manually creating and curating a ground truth is also not a viable option. For example, in the case of the library data domain, some datasets define the concept of a book as Work, Manifestation, or Bibliographic Resource, while others adopt more specific terms such as Book or Document. One of the selected datasets (British) even uses both general and specific terms to define the concept of a book. Therefore, even though, theoretically, we could compare the top-ranked candidate with approaches that match entity types, similarly to the content score evaluation, to the best of our knowledge, there is no ground truth that contains a ranking of entity type candidates that evaluates their interoperability and it is unlikely to exist since, as mentioned, the best candidate is often use-case dependent. Furthermore, interoperability is not focused on improving the ranking based on precision and most ground truths contain evaluations that are focused on that metric. Instead, interoperability is focused on maximising connectivity with the knowledge graph.

Therefore, in this section we present an analysis and discussion of the effects the interoperability score has on the rankings produced by the framework. We perform this analysis for the library and biomedical use-cases.

Interoperability Changes with Weighted Distance

For our first experiment, we analyse the evolution of neighbourhood size $NS(v, d_{max})$ and interoperability metric $\mathtt{interop}(v, d_{max})$ with the distance from candidates.

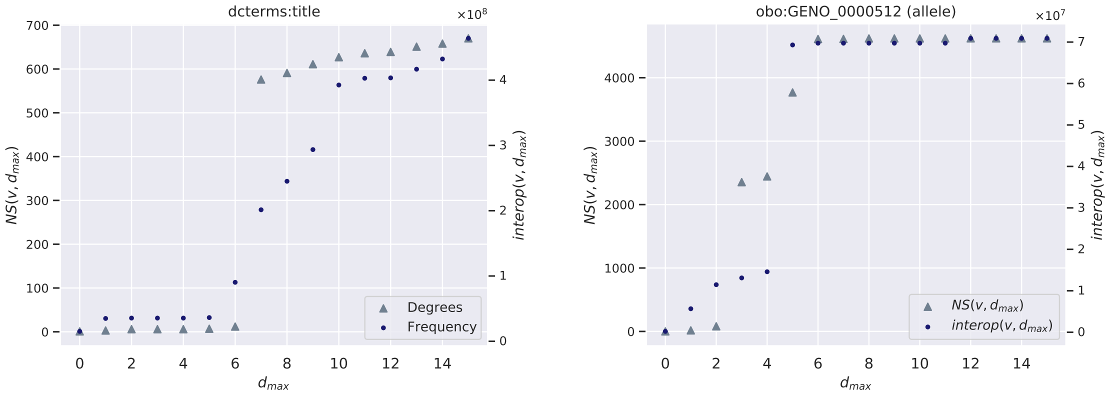
Figure 4.16 has an example representative of the result patterns found for entity type candidates in the two use-cases, with $d_{max} = 15$. The figures show that both interoperability metrics follow similar trends. At distance 0, the measures consider the candidate by itself only. Therefore, $\mathtt{interop}(v, 0)$ is equivalent to the frequency of the entity type in the document store, and consequently $NS(v, d_{max})$ is 1. As the neighbourhood expands, for the library use-case, these values increase rapidly until distance 10 where the neighbourhood size tends not to increase significantly more. For the biomedical use-case, a similar pattern is verified where by $d_{max} = 10$, both the interoperability metric and the neighbourhood size have converged to a maximum value.



(a) Library use-case

(b) Biomedical use-case

Figure 4.16: Neighbourhood size and interoperability metric changes with Weighted Distance for one example of entity types of the (a) library and (b) biomedical use-case



(a) Library use-case

(b) Biomedical use-case

Figure 4.17: Neighbourhood size and interoperability metric changes with Weighted Distance for one example of properties of the (a) library and (b) biomedical use-case

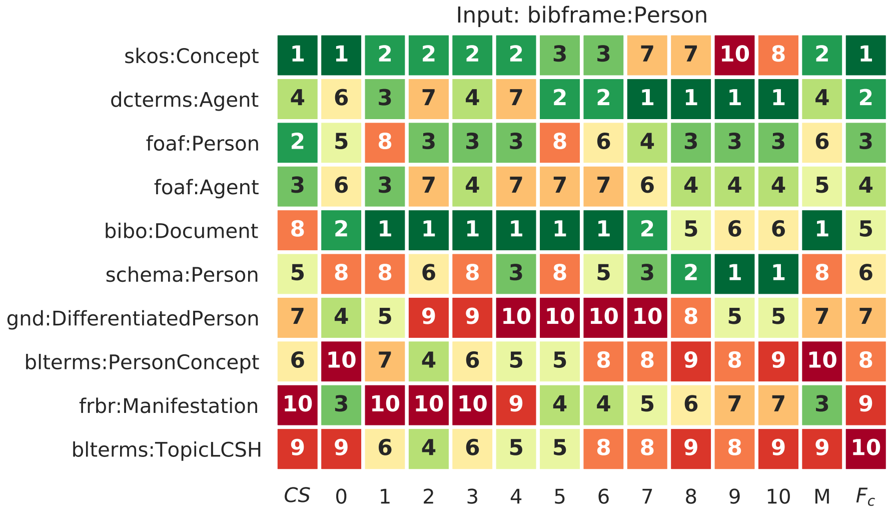
In figure 4.17, the same experiment is performed over two representative examples of properties. The pattern seen in Figure 4.17b is common among the properties in the biomedical use-case properties. This pattern shows that the property is connected with 3 neighbours that are at a distance of 2 from the input property. The biomedical use-case features ontologies with rich entity type hierarchies but shallow property hierarchies. From all the ontologies in the biomedical ontology graph, the total number of the hierarchical relationships for classes is 2 630 492, while for properties it is 713. Therefore, the properties in the ontology graph and not well-connected which leads to interoperability patterns as seen in Figure 4.17b. For the library use-case, in Figure 4.17a, we verify that properties in the library use-case still follow a pattern similar to entity type candidates, where around $d_{max} = 10$ the values of the metrics do not significantly increase anymore.

From the observation of the patterns with entity types and properties with the interoperability measures, we generalise for the remaining experiments and set $d_{max} = 10$.

Interoperability Ranking Analysis

The next experiment compares the data model element candidates scores at different stages to analyse the impact of the content score (CS) and the interoperability score (IS) in the overall ranking of candidates. We include a final score F_c which is the combination of the content score with the interoperability score, following the function $F_c = pow(cs, w_{cs}) \cdot pow(is, w_{is})$, similarly to the function $scoring$ from Algorithm 4.8, with weights of 1.0 for both scores. For the purposes of this experiment, we show the rankings up to $d_{max} = 10$ but calculate F_c with the median, M, of the interoperability scores, IS, which is equivalent to $d_{max} = 5$. For each experiment, we present examples representative[9] of the diverse results we obtained that illustrate the points requiring discussion.

ENTITY TYPES Figures 4.18 through 4.22 show the rankings for representative input entity type examples in the library use-case datasets according to different measures and parameters. The figures show on the Y-axis the top-10 candidates according to F_c. On the X-axis we find CS which is the ranking of the candidate according only to the content score, then neighbourhood distances d from 0 (only the candidate) to 10, then the ranking according to the median M of the scores and, finally, the ranking according to F_c. Figures 4.23 through 4.25 capture the same results for the biomedical use-case.

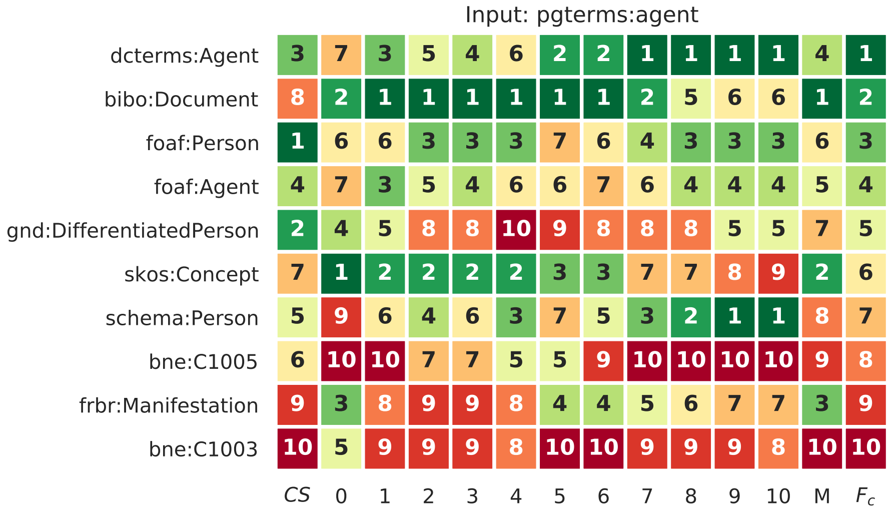


	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
skos:Concept	1	1	2	2	2	2	3	3	7	7	10	8	2	1
dcterms:Agent	4	6	3	7	4	7	2	2	1	1	1	1	4	2
foaf:Person	2	5	8	3	3	3	8	6	4	3	3	3	6	3
foaf:Agent	3	6	3	7	4	7	7	7	6	4	4	4	5	4
bibo:Document	8	2	1	1	1	1	1	1	2	5	6	6	1	5
schema:Person	5	8	8	6	8	3	8	5	3	2	1	1	8	6
gnd:DifferentiatedPerson	7	4	5	9	9	10	10	10	10	8	5	5	7	7
blterms:PersonConcept	6	10	7	4	6	5	5	8	8	9	8	9	10	8
frbr:Manifestation	10	3	10	10	10	9	4	4	5	6	7	7	3	9
blterms:TopicLCSH	9	9	6	4	6	5	5	8	8	9	8	9	9	10

Figure 4.18: Heatmap of rankings for example entity type in the University dataset

Figures 4.18 and 4.19 include entity type candidates for author entities in the University and Gutenberg datasets, respectively. In the University heatmap, we observe that

[9] The extended results are available at `https://github.com/danielapoliveira/phd-thesis-additional-materials/tree/master/interop-results`



	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dcterms:Agent	3	7	3	5	4	6	2	2	1	1	1	1	4	1
bibo:Document	8	2	1	1	1	1	1	1	2	5	6	6	1	2
foaf:Person	1	6	6	3	3	3	7	6	4	3	3	3	6	3
foaf:Agent	4	7	3	5	4	6	6	7	6	4	4	4	5	4
gnd:DifferentiatedPerson	2	4	5	8	8	10	9	8	8	8	5	5	7	5
skos:Concept	7	1	2	2	2	2	3	3	7	7	8	9	2	6
schema:Person	5	9	6	4	6	3	7	5	3	2	1	1	8	7
bne:C1005	6	10	10	7	7	5	5	9	10	10	10	10	9	8
frbr:Manifestation	9	3	8	9	9	8	4	4	5	6	7	7	3	9
bne:C1003	10	5	9	9	9	8	10	10	9	9	9	8	10	10

Figure 4.19: Heatmap of rankings for example entity type in the Gutenberg dataset

`skos:Concept` was the highest ranked candidate according to CS, while `foaf:Person` was the highest for the Gutenberg input. However, for the Gutenberg dataset, when the content score is combined with the interoperability score, the highest ranked candidate is `dcterms:Agent`, which is also boosted in the University candidates to the second position instead of fourth. `dcterms:Agent` is more generic than `foaf:Person`, therefore, in terms of interoperability, it is more desirable since it is more likely to link to more entity types in the ontology graph. In the University dataset, however, it obtained a lower raw content score, CS, which leads to its lower ranking. In this dataset, `skos:Concept` combined a strong content score with a high interoperability and, therefore, maintained the top-ranked position.

When looking at the ranking of `bibo:Document` in both datasets, we verify that the interoperability boosted an undesirable candidate since this entity type is well-connected and one of the most frequent in the knowledge graph (used by all books in the German Library). Therefore, this is a case where interoperability is enabling the introduction of noise in the ranking. These cases can be counteracted by balancing the weights given to the content and interoperability scores.

Therefore, the next experiment analyses how the scores changed by balancing the weight given to the content score and interoperability score. Figure 4.20 shows the results of this experiment. In both datasets, we observe that `bibo:Document` is the best ranked in terms of interoperability by a significant margin, however in terms of content score it is ranked much lower. For the Gutenberg dataset, except `bibo:Document`, `frbr:Manifestation`, and `bne:C1003` (Manifestation), the remaining candidates can arguably be considered acceptable candidates to represent an author in a library dataset. On the University side, both `bibo:Document` and `blterms:TopicLCSH` are arguably incorrect. Therefore, a balance between content and interoperability score should be decided by the user, considering that increasing the weight of interoperability might bring more

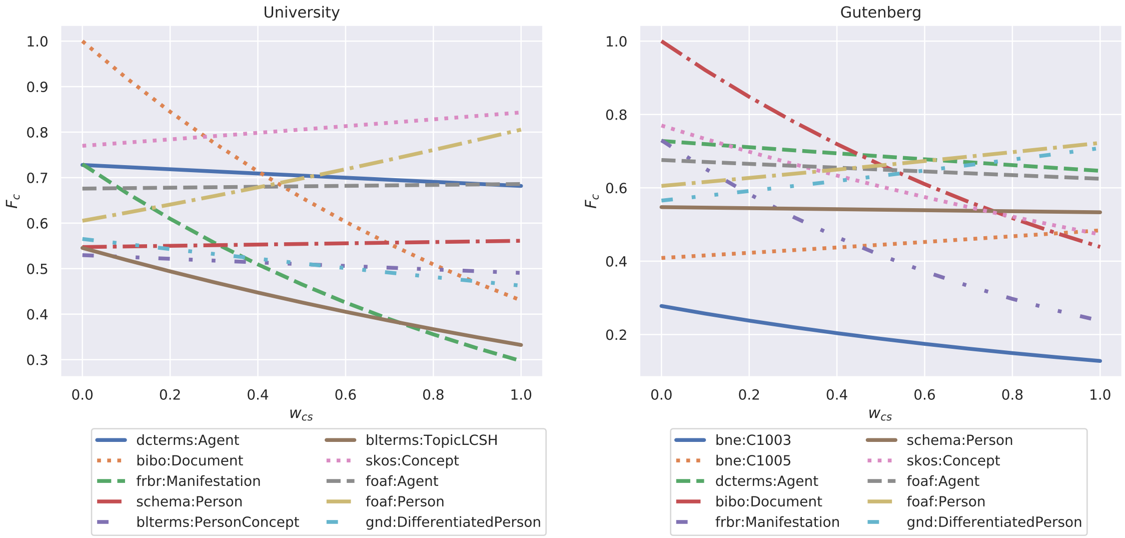
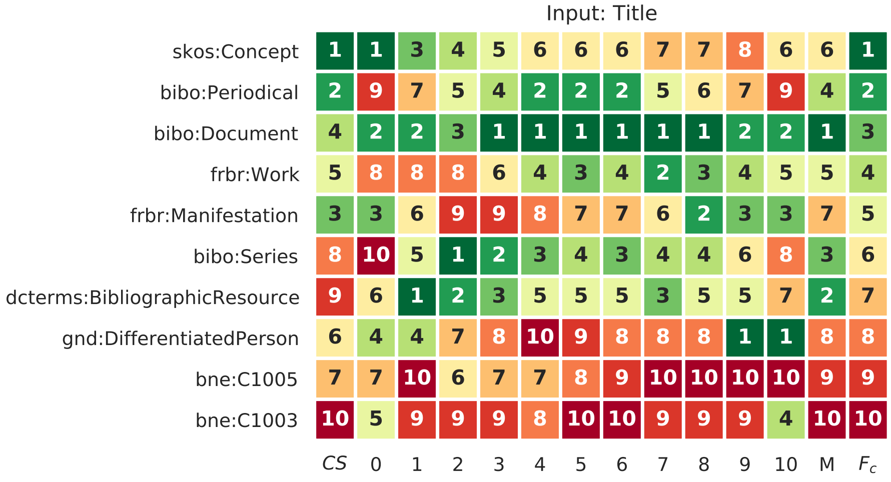


Figure 4.20: Balance between weights giving to the final score combination between content and interoperability score

noise to the candidate ranking, while reducing the interoperability score will give preference to highly specific candidates that might be more challenging to integrate with existing and future resources.

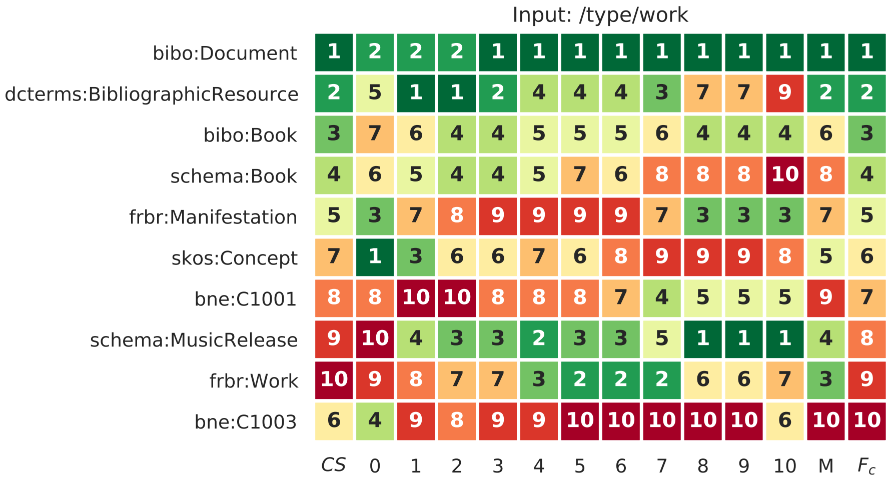
Figures 4.21 and 4.22 show the same weighted distance analysis for the non-RDF input datasets of the library use-case, using magazines entities for the Institute library and books for the Open Library. Similar conclusions to the RDF inputs are drawn from the non-RDF datasets.


Input: Title

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
skos:Concept	1	1	3	4	5	6	6	6	7	7	8	6	6	1
bibo:Periodical	2	9	7	5	4	2	2	2	5	6	7	9	4	2
bibo:Document	4	2	2	3	1	1	1	1	1	1	2	2	1	3
frbr:Work	5	8	8	8	6	4	3	4	2	3	4	5	5	4
frbr:Manifestation	3	3	6	9	9	8	7	7	6	2	3	3	7	5
bibo:Series	8	10	5	1	2	3	4	3	4	4	6	8	3	6
dcterms:BibliographicResource	9	6	1	2	3	5	5	5	3	5	5	7	2	7
gnd:DifferentiatedPerson	6	4	4	7	8	10	9	8	8	8	1	1	8	8
bne:C1005	7	7	10	6	7	7	8	9	10	10	10	10	9	9
bne:C1003	10	5	9	9	9	8	10	10	9	9	9	4	10	10

Figure 4.21: Heatmap of rankings for example entity type in the Institute dataset

A notable mention includes the candidate `bibo:Periodical` in Figure 4.21. Considering that the column Title in the CSV file included magazine titles, this candidate is arguably the most specific to describe the input. From the analysis of the interoperability, we observe that by itself (i.e, distance 0), this candidate has a low interoperability (ranked

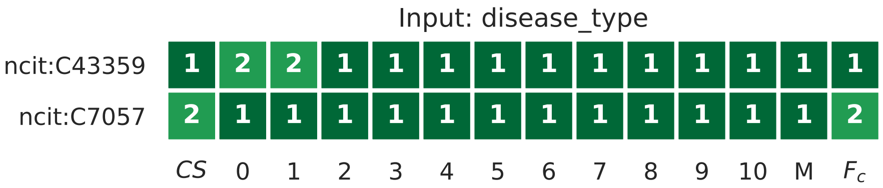

Input: /type/work

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
bibo:Document	1	2	2	2	1	1	1	1	1	1	1	1	1	1
dcterms:BibliographicResource	2	5	1	1	2	4	4	4	3	7	7	9	2	2
bibo:Book	3	7	6	4	4	5	5	5	6	4	4	4	6	3
schema:Book	4	6	5	4	4	5	7	6	8	8	8	10	8	4
frbr:Manifestation	5	3	7	8	9	9	9	9	7	3	3	3	7	5
skos:Concept	7	1	3	6	6	7	6	8	9	9	9	8	5	6
bne:C1001	8	8	10	10	8	8	8	7	4	5	5	5	9	7
schema:MusicRelease	9	10	4	3	3	2	3	3	5	1	1	1	4	8
frbr:Work	10	9	8	7	7	3	2	2	2	6	6	7	3	9
bne:C1003	6	4	9	8	9	9	10	10	10	10	10	6	10	10

Figure 4.22: Heatmap of rankings for example entity type in the OpenL dataset

9th). However, at distance one, it has `frbr:Work` as an `owl:equivalentClass` and, at distance 2, its neighbourhood includes `bibo:Document`. Therefore, `bibo:Periodical` has not only high content score but also high interoperability when accounting for its neighbourhood and ends up being one of the most accurate and interoperable candidates.

Figures 4.23 through 4.25 show the same analysis for the three biomedical use-case datasets.

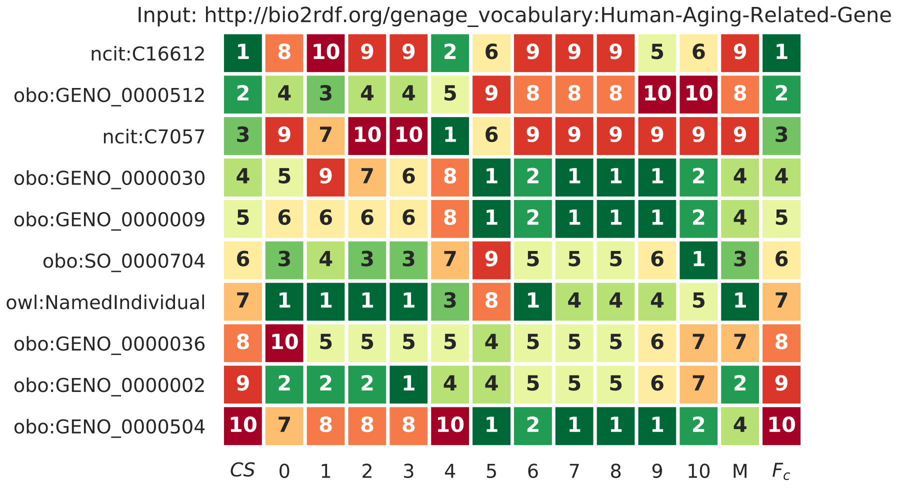

Input: disease_type

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
ncit:C43359	1	2	2	1	1	1	1	1	1	1	1	1	1	1
ncit:C7057	2	1	1	1	1	1	1	1	1	1	1	1	1	2

Figure 4.23: Heatmap of rankings for example entity type in the GDC dataset

For the results of the GDC dataset, shown in Figure 4.23, only two candidates were found: `ncit:C7057` (Disease, Disorder or Finding) and `ncit:C43359` (Group). Since not a lot of diverse and overlapping data sources were loaded in the knowledge graph, the framework was not able to find a wide variety of candidates. Nonetheless, it managed to find at least two candidates among the possibilities, with one of them (`ncit:C7057`) more clearly related to the input than the other. The match with `ncit:C43359` comes from the DisGeNET dataset, which identifies groups of diseases (e.g., *Abdominal Neoplasms*) with this entity type. The GDC dataset includes also groups of diseases (e.g., *Adenomas and Adenocarcinomas*) and, therefore, this input was matched with the candidate `ncit:C43359` (Group).

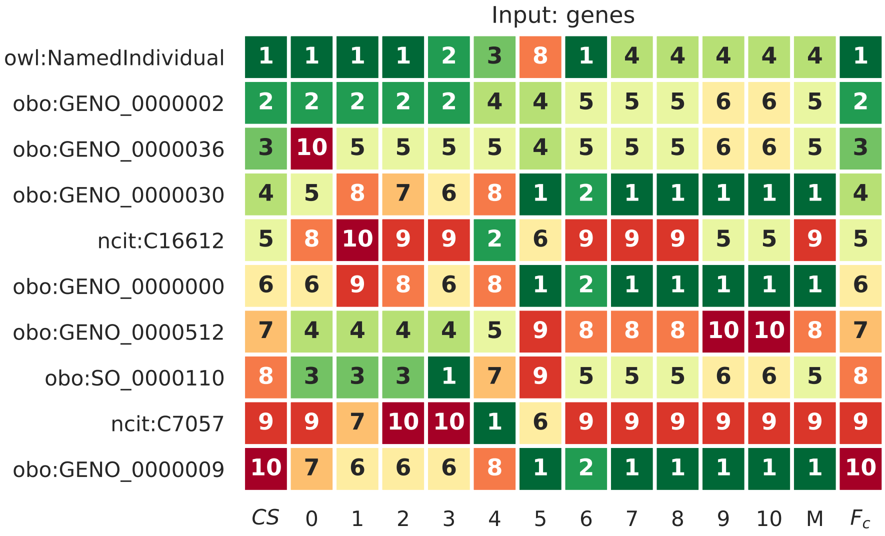
In Figures 4.24 and 4.25, both inputs are gene entities described by their name and symbol, respectively. The data sources loaded into the knowledge graph covered this domain more and, therefore, a wider variety of candidates was found for these datasets.



Input: http://bio2rdf.org/genage_vocabulary:Human-Aging-Related-Gene

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
ncit:C16612	1	8	10	9	9	2	6	9	9	9	5	6	9	1
obo:GENO_0000512	2	4	3	4	4	5	9	8	8	8	10	10	8	2
ncit:C7057	3	9	7	10	10	1	6	9	9	9	9	9	9	3
obo:GENO_0000030	4	5	9	7	6	8	1	2	1	1	1	2	4	4
obo:GENO_0000009	5	6	6	6	6	8	1	2	1	1	1	2	4	5
obo:SO_0000704	6	3	4	3	3	7	9	5	5	5	6	1	3	6
owl:NamedIndividual	7	1	1	1	1	3	8	1	4	4	4	5	1	7
obo:GENO_0000036	8	10	5	5	5	5	4	5	5	5	6	7	7	8
obo:GENO_0000002	9	2	2	2	1	4	4	5	5	5	6	7	2	9
obo:GENO_0000504	10	7	8	8	8	10	1	2	1	1	1	2	4	10

Figure 4.24: Heatmap of rankings for example entity type in the GenAge dataset



Input: genes

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
owl:NamedIndividual	1	1	1	1	2	3	8	1	4	4	4	4	4	1
obo:GENO_0000002	2	2	2	2	2	4	4	5	5	5	6	6	5	2
obo:GENO_0000036	3	10	5	5	5	5	4	5	5	5	6	6	5	3
obo:GENO_0000030	4	5	8	7	6	8	1	2	1	1	1	1	1	4
ncit:C16612	5	8	10	9	9	2	6	9	9	9	5	5	9	5
obo:GENO_0000000	6	6	9	8	6	8	1	2	1	1	1	1	1	6
obo:GENO_0000512	7	4	4	4	4	5	9	8	8	8	10	10	8	7
obo:SO_0000110	8	3	3	3	1	7	9	5	5	5	6	6	5	8
ncit:C7057	9	9	7	10	10	1	6	9	9	9	9	9	9	9
obo:GENO_0000009	10	7	6	6	6	8	1	2	1	1	1	1	1	10

Figure 4.25: Heatmap of rankings for example entity type in the PharmGKB dataset

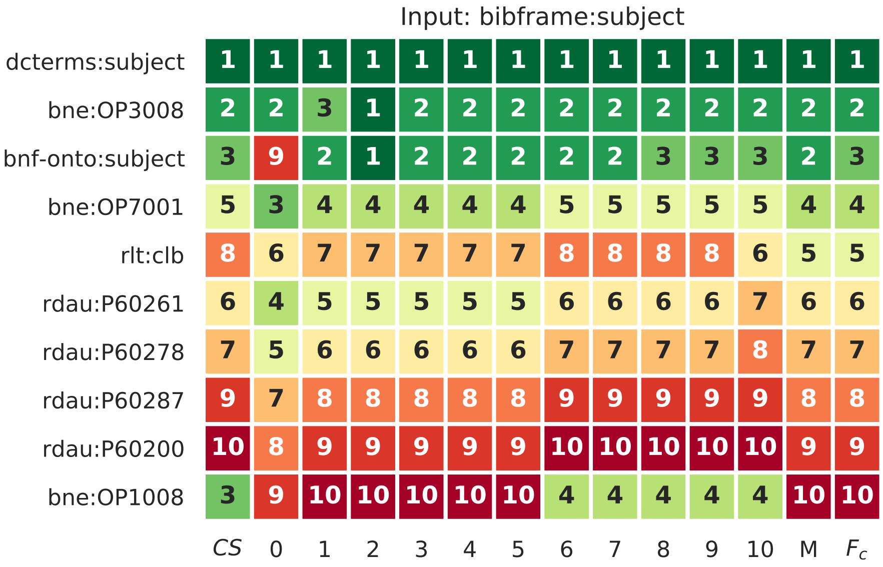
Both datasets feature, for example, `obo:GENO_0000512` (allele) and `ncit:C16612` (Gene) with the remaining candidates being closely or loosely related to genes or genetic terminology. Since this use-case is related to less general domain knowledge than the library use-case, a domain expert is required to determine the adequacy of the candidates generated.

Nonetheless, for both use-cases, the final result of the entity type ranking should be analysed by a domain expert, which has the final say in which data model is chosen. Since the goal of the framework is to provide candidates ranked in a specific ordering focused on different characteristics of the data, it is up to a data publisher to determine the best parameters for their data.

PROPERTIES We performed the same analysis over the object and datatype property candidates of the four input datasets of the library use-case. Figures 4.26 and 4.29 show an example of the result of these experiments for each of the datasets in the library use-case and Figure 4.30 shows an example for a dataset in the biomedical use-case.

Figures 4.26 and 4.27 show the results of this experiment for two object properties in the University and Gutenberg datasets, respectively. For the `bibframe:subject` property in Figure 4.26, the top-3 properties are arguably considered correct (`bne:OP3008` - has subject). In this instance, the top-3 candidates have similar interoperability scores and, therefore, there were not any changes in relation to the content score ranking. It is also likely that no more correct matches exist since both the German and British libraries use `dcterms:subject`, and the remaining two properties are used by the Spanish and French libraries. The Portuguese library does not use this property, and therefore, the top-3 cover the best candidates in each library.

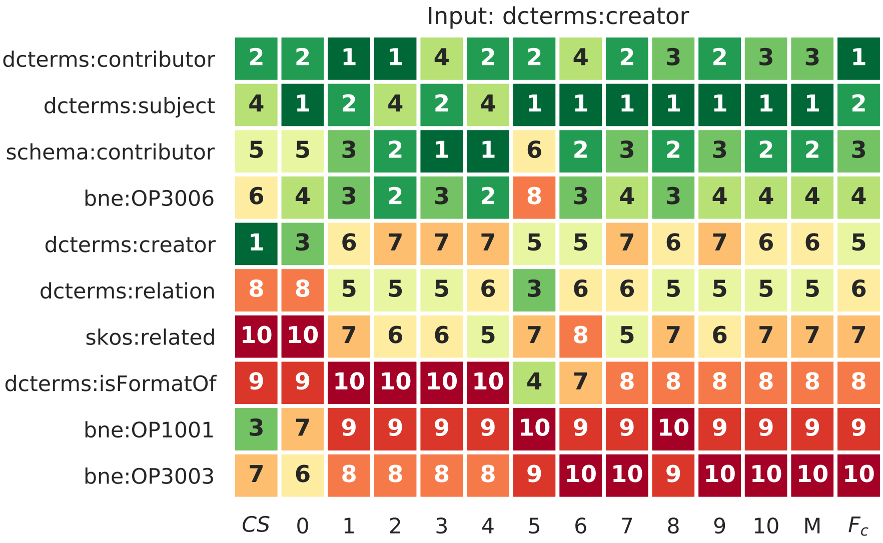
In Figure 4.27, the top ranked candidates are not so fitting since `dcterms:subject` was ranked second, with the same candidate as the input (`dcterms:creator`) ranked sixth due to its lower interoperability. This situation is another issue of attributing equal weight to the content and interoperability scores since the most accurate candidate according to the content score ends up being ranked low due to its low connectivity. However, in cases where interoperability with multiple data sources is desirable, the data publisher might prefer to select a highly connected candidate over a lower one by sacrificing some data model specificity.



	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dcterms:subject	1	1	1	1	1	1	1	1	1	1	1	1	1	1
bne:OP3008	2	2	3	1	2	2	2	2	2	2	2	2	2	2
bnf-onto:subject	3	9	2	1	2	2	2	2	2	3	3	3	2	3
bne:OP7001	5	3	4	4	4	4	4	5	5	5	5	5	4	4
rlt:clb	8	6	7	7	7	7	7	8	8	8	8	6	5	5
rdau:P60261	6	4	5	5	5	5	5	6	6	6	6	7	6	6
rdau:P60278	7	5	6	6	6	6	6	7	7	7	7	8	7	7
rdau:P60287	9	7	8	8	8	8	8	9	9	9	9	9	8	8
rdau:P60200	10	8	9	9	9	9	9	10	10	10	10	10	9	9
bne:OP1008	3	9	10	10	10	10	10	4	4	4	4	4	10	10

Figure 4.26: Heatmap of rankings for example object property in the University dataset

Figures 4.28 and 4.29 show the results of the same experiment for two datatype properties in the Institute and OpenL datasets, respectively. In Figure 4.28, the input property has book titles as values. Despite having similar features as the previous datasets, due

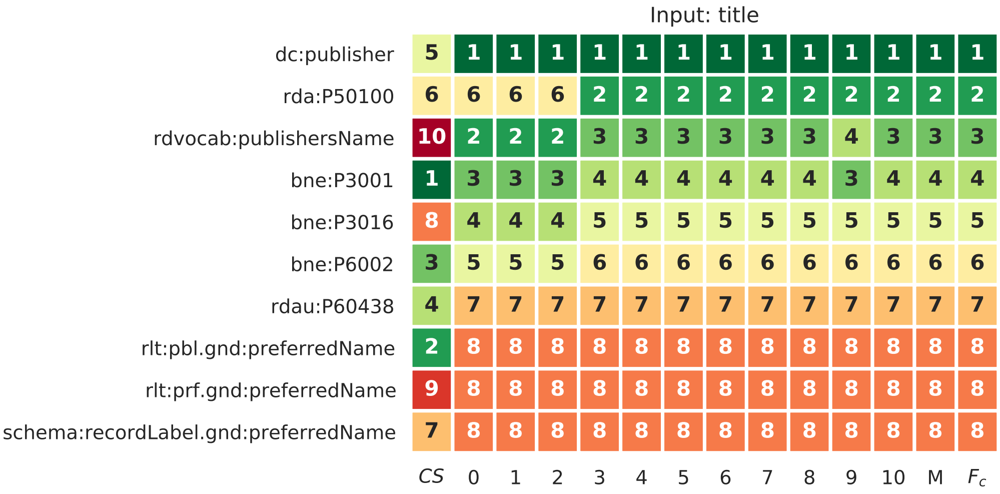


Input: dcterms:creator

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dcterms:contributor	2	2	1	1	4	2	2	4	2	3	2	3	3	1
dcterms:subject	4	1	2	4	2	4	1	1	1	1	1	1	1	2
schema:contributor	5	5	3	2	1	1	6	2	3	2	3	2	2	3
bne:OP3006	6	4	3	2	3	2	8	3	4	3	4	4	4	4
dcterms:creator	1	3	6	7	7	7	5	5	7	6	7	6	6	5
dcterms:relation	8	8	5	5	5	6	3	6	6	5	5	5	5	6
skos:related	10	10	7	6	6	5	7	8	5	7	6	7	7	7
dcterms:isFormatOf	9	9	10	10	10	10	4	7	8	8	8	8	8	8
bne:OP1001	3	7	9	9	9	9	10	9	9	10	9	9	9	9
bne:OP3003	7	6	8	8	8	8	9	10	10	9	10	10	10	10

Figure 4.27: Heatmap of rankings for example object property in the Gutenberg dataset

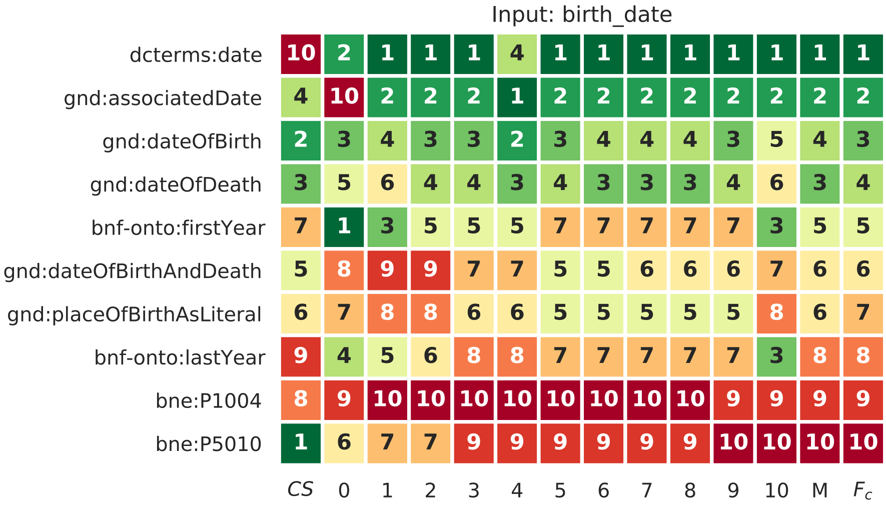
to the low amount of samples extracted from this dataset, we observe that none of the candidates ranked in the top-10 are obvious matches. The low sample size leads to a poor performance of the classification model, which predicts properties incorrectly and translates to Borda scores that are not in line with the best candidates for the input property. On the other hand, Figure 4.29 shows that the classification model and the content score are able to produce good results in the top-10 that are further re-ordered according to the interoperability score. This produces candidate dates that are more interoperable even if less specific. For example, `dcterms:date` is arguably correct, even if `gnd:dateOfBirth` is more specific to the date in question. Nonetheless, once again, it is up to the data publisher to decide between the more specific or more interoperable candidates.



Input: title

	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dc:publisher	5	1	1	1	1	1	1	1	1	1	1	1	1	1
rda:P50100	6	6	6	6	2	2	2	2	2	2	2	2	2	2
rdvocab:publishersName	10	2	2	2	3	3	3	3	3	3	4	3	3	3
bne:P3001	1	3	3	3	4	4	4	4	4	4	3	4	4	4
bne:P3016	8	4	4	4	5	5	5	5	5	5	5	5	5	5
bne:P6002	3	5	5	5	6	6	6	6	6	6	6	6	6	6
rdau:P60438	4	7	7	7	7	7	7	7	7	7	7	7	7	7
rlt:pbl.gnd:preferredName	2	8	8	8	8	8	8	8	8	8	8	8	8	8
rlt:prf.gnd:preferredName	9	8	8	8	8	8	8	8	8	8	8	8	8	8
schema:recordLabel.gnd:preferredName	7	8	8	8	8	8	8	8	8	8	8	8	8	8

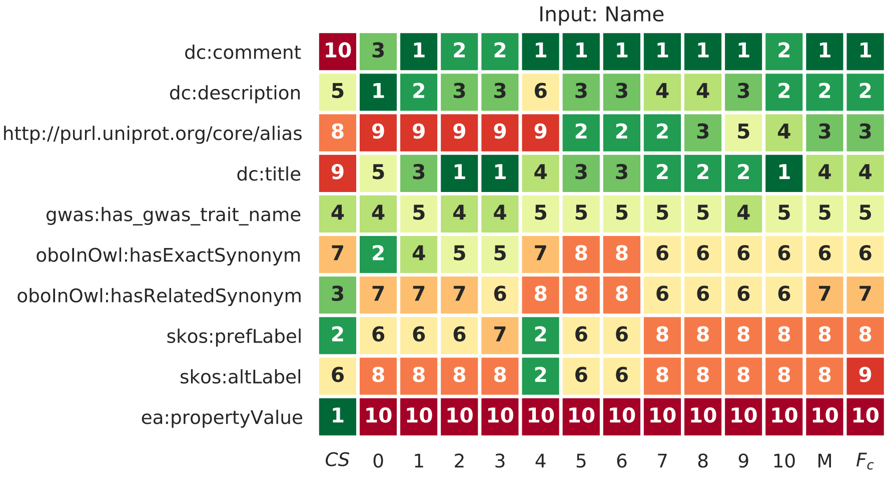
Figure 4.28: Heatmap of rankings for example datatype property in the Institute dataset



	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dcterms:date	10	2	1	1	1	4	1	1	1	1	1	1	1	1
gnd:associatedDate	4	10	2	2	2	1	2	2	2	2	2	2	2	2
gnd:dateOfBirth	2	3	4	3	3	2	3	4	4	4	3	5	4	3
gnd:dateOfDeath	3	5	6	4	4	3	4	3	3	3	4	6	3	4
bnf-onto:firstYear	7	1	3	5	5	5	7	7	7	7	7	3	5	5
gnd:dateOfBirthAndDeath	5	8	9	9	7	7	5	5	6	6	6	7	6	6
gnd:placeOfBirthAsLiteral	6	7	8	8	6	6	5	5	5	5	5	8	6	7
bnf-onto:lastYear	9	4	5	6	8	8	7	7	7	7	7	3	8	8
bne:P1004	8	9	10	10	10	10	10	10	10	10	9	9	9	9
bne:P5010	1	6	7	7	9	9	9	9	9	9	10	10	10	10

Figure 4.29: Heatmap of rankings for example datatype property in the OpenL dataset

As previously mentioned, the ontology graph of the biomedical use-case does not include many properties or relationships between them. Furthermore, overall the input data contains only 5 object properties. Figure 4.30 shows an example of a datatype property in the PharmGKB dataset. In this case, the property name includes names of chemicals, drugs, genes, phenotypes, and variants. Due to its variety, the classification model and scores are unable to find and rank matches appropriately. However, in other cases where matches should be easily findable in the knowledge graph, the classification models and the ranking still underperforms because the datatype property values are not well represented in the training data. Therefore, the model is not able to make accurate predictions. Similarly, situations are found in the remaining datatype properties of the biomedical use-case and, therefore, the property ranking mostly underperforms for this use-case.



	CS	0	1	2	3	4	5	6	7	8	9	10	M	F_c
dc:comment	10	3	1	2	2	1	1	1	1	1	1	2	1	1
dc:description	5	1	2	3	3	6	3	3	4	4	3	2	2	2
http://purl.uniprot.org/core/alias	8	9	9	9	9	9	2	2	2	3	5	4	3	3
dc:title	9	5	3	1	1	4	3	3	2	2	2	1	4	4
gwas:has_gwas_trait_name	4	4	5	4	4	5	5	5	5	5	4	5	5	5
oboInOwl:hasExactSynonym	7	2	4	5	5	7	8	8	6	6	6	6	6	6
oboInOwl:hasRelatedSynonym	3	7	7	7	6	8	8	8	6	6	6	6	7	7
skos:prefLabel	2	6	6	6	7	2	6	6	8	8	8	8	8	8
skos:altLabel	6	8	8	8	8	2	6	6	8	8	8	8	8	9
ea:propertyValue	1	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 4.30: Heatmap of rankings for example datatype property in the PharmGKB dataset

4.5.5 Consistency

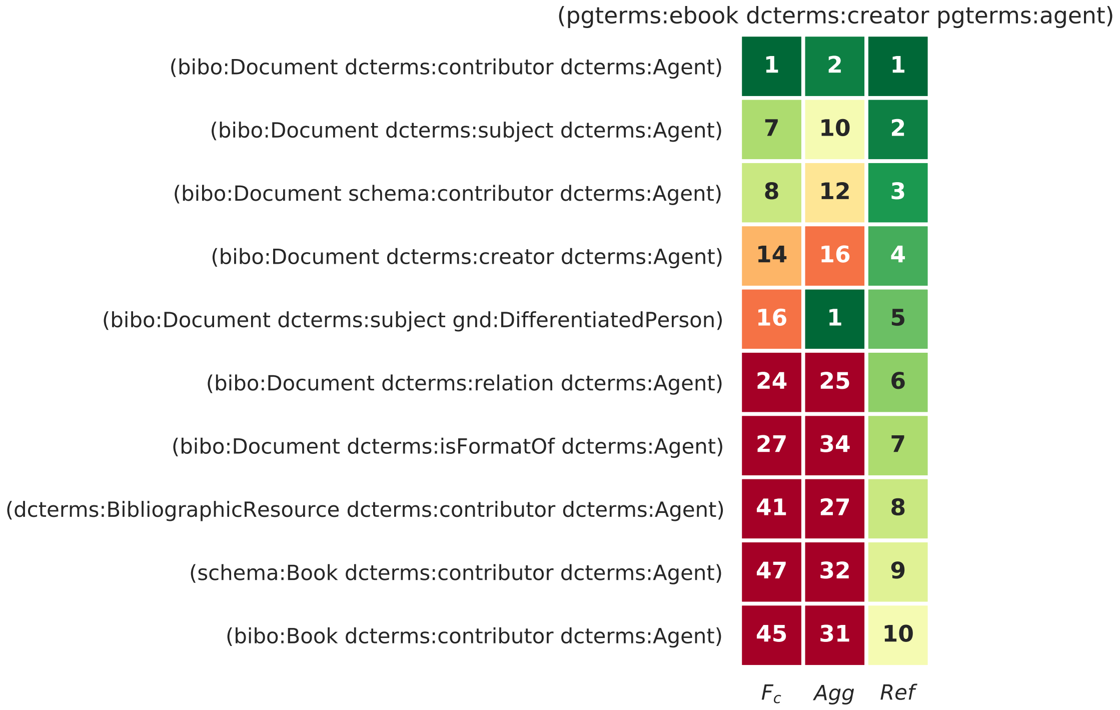
The last step of the framework aims to uniformise the results suggested when comparing with the knowledge graph and within the proposed data model. Until this step, candidates are ranked independently of each other. The consistency score increases the likelihood of the same candidate being suggested for the same input entity type or property and, at the same time, boosts triples that are more commonly encountered in the knowledge graph. Similarly to interoperability, to the best of our knowledge, no ground truth exists that can reliably evaluate the performance of these methods, therefore, we also proceed with an exploratory analysis based on observation and comparison with the previous framework components.

Figure 4.31 shows a representative example[10] for the Gutenberg library of the effects of the consistency score in the candidate rankings for the input triple ⟨`pgterms:ebook` – `dcterms:creator` ⇾ `pgterms:agent`⟩.

The F_c rankings represent the content score CS times the interoperability score IS with weights of 1.0. Agg represents the score obtained after the aggregation step of the consistency score algorithm, which combines the score F_c with the co-occurrences of the triple elements in the knowledge graph. Finally, Ref represents the rankings after the refinement step of the consistency score, which aims to provide an homogeneous data model where the same inputs are represented by the same candidates.

In this example, the top ranked candidate triple is not changed in any of the steps, however, the remaining triples are significantly changed by the consistency steps. Generally triples with `bibo:Document` as domain and `dcterms:Agent` as range are ranked higher since these were the triples that obtained higher scores and, therefore, are more frequently ranked first. The predicate `dcterms:subject` ranked above other more desirable candidates due to its high interoperability. This is another case where a different balance between the weights given to interoperability and content score could positively affect the final results. The remaining candidates as boosted by co-occurrence frequency and the score refinement to appear in the top-20 recommended triples for the input triple pattern. The remaining candidates follow similar patterns, where despite obtaining lower scores when combining content and interoperability metrics, the triple score is boosted by co-occurrence in the Agg step, and further refined in the last step. This pattern is also verified for all remaining triple patterns of the datasets of both use-cases. Previous issues are not resolved in this step, but when the rankings follow the expected outcomes, the resulting rankings follow a similar pattern to the example in Figure 4.31.

[10] The extended results are available at `https://github.com/danielapoliveira/phd-thesis-additional-materials/tree/master/consistency-results`



(pgterms:ebook dcterms:creator pgterms:agent)

	F_c	*Agg*	*Ref*
(bibo:Document dcterms:contributor dcterms:Agent)	1	2	1
(bibo:Document dcterms:subject dcterms:Agent)	7	10	2
(bibo:Document schema:contributor dcterms:Agent)	8	12	3
(bibo:Document dcterms:creator dcterms:Agent)	14	16	4
(bibo:Document dcterms:subject gnd:DifferentiatedPerson)	16	1	5
(bibo:Document dcterms:relation dcterms:Agent)	24	25	6
(bibo:Document dcterms:isFormatOf dcterms:Agent)	27	34	7
(dcterms:BibliographicResource dcterms:contributor dcterms:Agent)	41	27	8
(schema:Book dcterms:contributor dcterms:Agent)	47	32	9
(bibo:Book dcterms:contributor dcterms:Agent)	45	31	10

Figure 4.31: Ranking of triple candidates according to final score F_c, score aggregation Agg, and score refinement Ref

4.5.6 Evaluating Distance to Source

The final experiment focuses on comparing the ranking obtained by our framework with the original entity types from RDF datasets. This evaluation was obtained by taking each RDF dataset, excluding their entities from the document store, and running the framework to produce a set of entity type candidates. The Gutenberg Library was excluded due to `pgterms` not being publicly available.

Again, we use the term *lenient* to refer to a metric that does not take into account a binary classification but considers the distance in the graph to make an assessment. Therefore, we distinguish two types of evaluation: (1) *strict* where the correctness of the rank is binary, and (2) *lenient* which considers the distance from the source type $t \in T$ to the candidate type $c \in C$.

The strict evaluation uses MRR and Precision@k (P@k), where $k \in [1, 3, 5]$ and the results are shown in Table 4.6. The lenient evaluation instead of binary correctness, considers the distance in the ontology graph G_e. We compute lenient precision by considering correct candidates at three distances $d \in [1, 2, 3]$, i.e., any candidate at a distance less or equal to d is considered correct. Then we compute P@k with $k \in [1, 3, 5]$. We calculate these metrics for every input entity type in each library, average the results, and present the aggregated result per library in Table 4.7.

Table 4.6 shows the results of the strict evaluation, which obtained a poor performance in all datasets. This result was expected since the removal of data from the document store has a high impact on both CS and IS due to frequency counts being lowered in the search results and document store for the original entity types. Therefore, the likelihood of our framework suggesting the same entity type as the original is low. Nonetheless, the strict evaluation shows that, for the French library, a reasonable MRR and P@1 are achieved after the content score re-ranking. This higher performance when compared with the remaining libraries is due to the schema overlap between the French library with elements with other libraries in the knowledge graph. This evaluation, therefore, leads us to conclude that when libraries use the same or similar schemas, they become easier to integrate since even after removing every instance of the French library, the framework managed to rank some entity types accurately.

Table 4.6: Results of the strict evaluation of the distance from source

		Strict			
Library	Method	MRR	P@1	P@3	P@5
British	CS	0.16	0.14	0.06	0.04
	IS	0.10	0.07	0.04	0.03
Spanish	CS	0.17	0.17	0.06	0.03
	IS	0.17	0.17	0.06	0.03
University	CS	0.00	0.00	0.00	0.00
	IS	0.00	0.00	0.00	0.00
German	CS	0.05	0.06	0.02	0.01
	IS	0.01	0.00	0.00	0.00
French	CS	0.33	0.30	0.17	0.10
	IS	0.18	0.10	0.10	0.08
Portuguese	CS	0.00	0.00	0.00	0.00
	IS	0.00	0.00	0.00	0.00

Table 4.7 shows the results for the lenient evaluation, in which we verify that the entity type candidates suggested are close to the original types. The German and Portuguese libraries have the lowest performances because they adopted data models that are structurally distant from the ones used in the other libraries.

This evaluation, together with the previous experiments, demonstrates that our framework proposes candidates that are potentially more interoperable with existing domain datasets while maintaining the original meaning intended in the library datasets by proposing similar entity types.

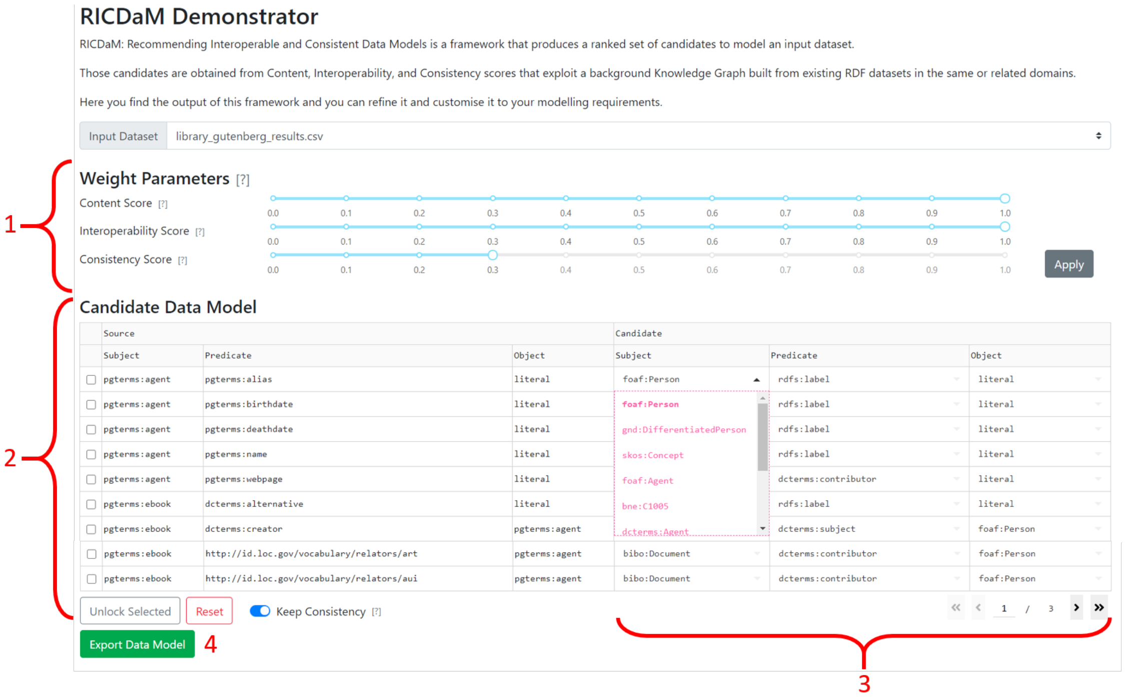
4.6 DEMONSTRATION

To showcase a potential final product of the framework, we created the RICDaM (Recommending Interoperable and Consistent Data Models) demonstrator. The demonstrator is available at `http://afel.insight-centre.org/ricdam/`. This demonstrator presents

Table 4.7: Results of the lenient evaluation of the distance from source

		Lenient								
Library	Method	P@1			P@3			P@5		
		d_1	d_2	d_3	d_1	d_2	d_3	d_1	d_2	d_3
British	CS	0.50	0.64	0.89	0.44	0.56	0.86	0.35	0.52	0.85
	IS	0.43	0.64	0.86	0.25	0.48	0.85	0.21	0.43	0.84
Spanish	CS	0.83	0.83	1.00	0.72	0.78	0.94	0.53	0.70	0.97
	IS	0.67	0.67	1.00	0.39	0.67	0.94	0.37	0.63	0.97
University	CS	0.14	0.41	0.67	0.11	0.44	0.65	0.11	0.41	0.65
	IS	0.16	0.53	0.65	0.08	0.43	0.64	0.07	0.41	0.64
German	CS	0.17	0.28	0.75	0.11	0.28	0.73	0.1	0.33	0.74
	IS	0.03	0.33	0.78	0.06	0.34	0.76	0.06	0.29	0.72
French	CS	0.70	1.00	1.00	0.53	0.87	0.93	0.46	0.88	0.96
	F_c	0.50	0.90	0.90	0.04	0.80	0.93	0.34	0.82	0.94
Portuguese	CS	0.00	0.00	1.00	0.00	0.33	1.00	0.00	0.40	1.00
	IS	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.40	1.00

the top candidate data model after the refinement step of the consistency score for an input dataset. The interface gives an overview of the best ranked candidates for each triple but also allows the user to adapt the data model to their preference and use-case.



RICDaM Demonstrator

RICDaM: Recommending Interoperable and Consistent Data Models is a framework that produces a ranked set of candidates to model an input dataset.

Those candidates are obtained from Content, Interoperability, and Consistency scores that exploit a background Knowledge Graph built from existing RDF datasets in the same or related domains.

Here you find the output of this framework and you can refine it and customise it to your modelling requirements.

Input Dataset: library_gutenberg_results.csv

Weight Parameters [?]

Content Score [?]: 1.0

Interoperability Score [?]: 1.0

Consistency Score [?]: 0.3

Candidate Data Model

	Source			Candidate		
	Subject	Predicate	Object	Subject	Predicate	Object
□	pgterms:agent	pgterms:alias	literal	foaf:Person	rdfs:label	literal
□	pgterms:agent	pgterms:birthdate	literal	foaf:Person	rdfs:label	literal
□	pgterms:agent	pgterms:deathdate	literal	gnd:DifferentiatedPerson	rdfs:label	literal
□	pgterms:agent	pgterms:name	literal	skos:Concept	rdfs:label	literal
□	pgterms:agent	pgterms:webpage	literal	foaf:Agent	dcterms:contributor	literal
□	pgterms:ebook	dcterms:alternative	literal	bne:C1005	rdfs:label	literal
□	pgterms:ebook	dcterms:creator	pgterms:agent	dcterms:Agent	dcterms:subject	foaf:Person
□	pgterms:ebook	http://id.loc.gov/vocabulary/relators/art	pgterms:agent	bibo:Document	dcterms:contributor	foaf:Person
□	pgterms:ebook	http://id.loc.gov/vocabulary/relators/aui	pgterms:agent	bibo:Document	dcterms:contributor	foaf:Person

Keep Consistency [?]: on

Figure 4.32: Screenshot of the RICDaM demonstrator

Figure 4.32 shows a screen shot of the RICDaM webpage and includes the illustration of the main functionalities, which are:

1. tuning of the parameters to produce different candidate rankings,

2. an overview of the output of the framework,
3. customisation of the data model, and
4. exporting the data model as a set of mappings between the input and the produced data model.

When the user makes a manual change to the data model, they can choose to propagate that change to maintain consistency across the dataset or keep the change locally to the modified cell. The tuning parameters allow the user to customise the ranking of the candidates, obtaining different top data models that can speed up the modelling process by suggesting the candidates the user is looking for more easily. Finally, the user can export the data model produced and apply the mappings to translate their original input data to an RDF dataset that is potentially more interoperable with the datasets in the knowledge graph.

The demo uses the library use-case knowledge graph and includes the Gutenberg and Open Library input datasets. For example, for `pgterms:ebook` in Project Gutenberg, the suggestion is to use `bibo:Document` as the most interoperable, relevant, and consistent entity type, together with properties such as `dcterms:contributor`. Through the interface, it can be changed, for example, to `schema:Book`, which is often used as well in the knowledge graph.

In general, designing a data model is not a trivial task. When considering integration with existing datasets, the task becomes more complex. This demo facilitates the task of finding the best possible data model according to certain criteria by producing a ranked list of candidates to match entity types and properties in a dataset. In the future, a complete tool using this framework would allow the user to provide input datasets to match loaded knowledge graphs, but also should allow the user to load their own data sources for the knowledge graph. The tool should also allow to manually search the indexed ontologies or add new ontologies to the graph to complete the model when the data model fails to find the desired class or property. For JSON or CSV input datasets, it would also be possible to generate a RML mapping file to facilitate the process of translating the data to RDF.

4.7 CONCLUSIONS

In this chapter, we focused on describing and experimenting with the methods for generating and ranking data model candidates. This task was motivated by the question:

RQ2 → *What measures can be used based on the built knowledge structures to select and rank possible data models?*

We used methods based on AOR to generate entity type candidates, classification models to generate datatype properties, and extracted object property candidates from the ontology graph by obtaining the vertices equivalent to the entity type candidates for two linked entity types in the input dataset.

Each one of the approaches posed different challenges. For entity type candidate generation, the task involves first mapping input entities to knowledge graph entities, and then extracting entity type mappings from these matches. Therefore, one of the first challenges of this approach, which was observed in the library use-case, is the integration of datasets in different languages. In the case of the library domain, enough overlap exists between the entities since, for example, book titles or author names are not always translated. By randomly selecting entities, we were able to find matches in different libraries. However, considering that our input datasets were all in English, knowledge graph data sources which feature more entities in this language are favoured over other languages. Therefore, in the future, investigating how to align and extend textual labels to provide multi-language support could improve the effectiveness and applications of the framework.

In terms of entity type generation, currently, the framework only uses `owl:sameAs` links as a means to enhance the ontology graph to improve scoring metrics that rely on traversing the graph. However, in the future, these links can be directly explored during the entity generation step to produce a more reliable candidate list. For example, similarly to the methods to create the `owl:sameAs` ground truth, when using an RDF dataset as input, if it features `owl:sameAs`, these can be extracted to generate candidates. If the input dataset does not use these links, but they exist in the knowledge graph, they can be explored to provide a more complete list of entity type candidates by extracting the entities that are indicated as values of the `owl:sameAs` property.

When generating property candidates, the first challenge is recognising if a predicate, attribute, or column is supposed to be a datatype or object property. This issue is less critical in RDF input datasets since distinguishing is a matter of finding if the dataset features the URI in the object of the predicate. This is the approach we use in general in this framework. However, for semi-structured CSV and JSON datasets, entities are rarely referenced via a unique identifier. Therefore, distinguishing datatype from object properties is not trivial. For example, in a CSV file, a column might contain book titles, while another contains author names. Looking at the existing data models in the library data domain, it is common that both these columns represent entities with a relationship between them. Although this seems like a trivial example, in the context of an automatic approach, there are several considerations. First, if the input dataset labels overlap with the knowledge graph labels, most columns will find some entity to match, even if incorrectly. For example, a column with page numbers can include a row for a book with 451 pages that can be matched incorrectly to the book title *Fahrenheit 451*. Therefore, robust measures need to be put in place to guarantee that only entities are obtaining entity type candidates. Currently, our generation methodology does not take this into account and the filtering of bad candidates is left to the content scoring. In the future, the framework should integrate more robust methods to distinguish between datatype and object properties, consider all the factors, and improve the candidate generation methodology.

Finally, the last challenge of entity type and property candidate generation is missing concepts, i.e., the input dataset includes data that is not found in the knowledge graph. For example, the University dataset was automatically converted to the BIBFRAME schema. Due to the scope of this schema, it includes several concepts that are not found in other library schemas, such as the entity type `bibframe:ColorContent` or the property `bibframe:baseMaterial`. These concepts do not have a direct overlap in any of the data models adopted by the European libraries in study. Therefore, finding suitable candidates for these concepts in the knowledge graph is not possible. In this case, investigation would need to be carried out to assess how to best handle the situation, considering that the immediate approach would most likely be to extend the ontologies in the graph and manually produce the annotations to complete the proposed data model. In a future implementation of this framework, ideally the user would have an integrated feature that would allow this extension without leaving the interface that implements the framework.

In the biomedical use-case, the candidate generation results for both entity types and properties were lacking due to the low conceptual overlap between the chosen inputs and the knowledge graph. As mentioned in Section 3.6, it is not trivial to find suitable datasets that overlap with the input and are adequate to build a knowledge graph. Research is progressing in this domain and including a step in the framework that facilitates this process would broaden the possible applications and increase the pool of possible users of this framework.

We developed two scoring methodologies: one focused on improving the score of potentially good candidates and another to increase the score of candidates that are well connected in the ontology graph and frequently used in the knowledge graph.

For the content scoring methodology, we present an evaluation and analysis of the effect of the different metrics integrated in the framework. In general, the methodologies obtain a reasonable performance, with the entity type methodology achieving better results than the property scoring methodologies. Nonetheless, for the primary library use-case, the methodology is successful in re-ordering the candidates that were generated. However, issues with the generation methodology are propagated for the ranking stage. Therefore, if poor results were achieved at that stage, the scoring methodologies are not going to achieve optimal performance.

Furthermore, due to the modular nature of the framework, for a particular dataset, a new measure might perform exceedingly well and, in that case, the framework can be extended to include the results of that metric.

In terms of the evaluation performed on the content score rankings, we were only able to use the ground truths to evaluate precision and recall since, to the best of our knowledge, no ground truth to evaluate ranked entities, entity types, or property candidates for data modelling. Therefore, we were not able to evaluate the results using the standard measures to validate the rankings produced, e.g., NDCG. Considering the nature of the data, a reliable ranking ground truth is not easy to obtain, since the correctness of a term over another is, in most cases, a matter of opinion and context. Therefore, choosing a candidate over another is not obvious and, in our evaluation, we consider each resource in the ground truth at an equal ranking and evaluate using precision only.

In regards to the first subquestion:

RQ2.1 → *Can the connections within the ontology graph be used to measure the accuracy and interoperability of possible data models, having a significant effect on the ranking of their components?*

Different datasets have different characteristics and the metrics used in each methodology can be adapted to improve them. For example, a use-case with a loosely connected ontology graph is going to achieve better results if the metrics not focused on the ontology graph are weighted more heavily than the ones that are focused on traversing the graph. On the other hand, small knowledge graphs with tightly connected ontology graphs benefit from the opposite situation. Therefore, further experiments and analyses can be carried out to assess the strengths and features of the contexts in which the proposed metrics should be boosted or penalised.

For the interoperability score, the choice of d_{max} is the parameter to take into account, which is also use-case dependent. At $d_{max} = 1$, the interoperability considers only direct neighbours which include only subclasses, equivalences, and perfect mappings. As d_{max} increases, the neighbourhood of the candidate is expanded to include more related entity types, potentially introducing more noise into the neighbourhood. Therefore, the user's choice is a balance between specificity and maximimisation of interoperability. In our experiments, we used a $d_{max} = 10$ to present a broader view of the impact of interoperability but empirically chose to use the median of all IS scores (i.e., maximum weighted distance of 5) to combine with the CS score. This median distance represents a middle point between using the generic values found at distance 10, when the interoperability score stabilises, and considering a significant distance from the source to boost candidates that are well-connected in the graph.

RQ2.2 → *Can the connections between components of the proposed data models be used to measure their consistency, with a significant effect on their ranking?*

The final scoring methodology presented focuses on producing a single data model candidate with candidates proposed per triple pattern in the input data, i.e., for each domain, property and range that exist in the input data model, a corresponding ranking of triple candidates with domain, property, and range candidates is produced. This score considers two sides: consistency with the knowledge graph and consistency within the proposed candidate data model. The first is concerned with modifying the score of a triple based on the frequency of co-occurrence of its parts in the knowledge graph, and the second is focuses on boosting the candidates that are being suggested more commonly for the same entity types in the input. We performed experiments that demonstrate how consistency works and discussed its impact on the results. For the chosen use-cases, we observed that the consistency methodology fulfilled its purpose and led to more consistent data models in terms of frequency in the knowledge graph and the candidates being suggested within the data model.

In the future, this score can include a third type of consistency, which considers the consistency in terms of independent data sources in the knowledge graph. Currently, the knowledge graph is treated by the framework as a single source. However, if it distinguishes between data sources, it is possible to boost the patterns that exist within the same source data model, ensuring that logical coherence is maintained from the original source. In this sense, depending on which data source is being mostly ranked first, the framework would boost the triple candidates that belong to that source. For example, if candidate triples from the British library are frequently ranked first, the framework would boost candidates from that library to improve the consistency of the model at the data source level.

A feature of this score is that it combines the previous scores with the co-occurrence scores to obtain a final triple score. This combination is achieved by balancing parameters that should align with the preferences of the user. In our experiments, we choose neutral weights (1.0 for w_{cs} and w_{is} and 0.5 for w_{cns}), however, these might not be optimal for different use-cases. In the case of the library data, we obtained reasonable results for the final rankings, but it is not guaranteed that it will be similar for other datasets. Besides, these parameters are meant to be adjusted to the preferences of the users. Therefore, the user can focus the framework by boosting the content, interoperability, or consistency scores. However, the user should also consider that the strength of this framework lies not on achieving the highest precision possible for each individual candidate but, instead is focused on producing a ranking of candidate triples that will be accurate, consistent, and interoperable. Maximising one of these scores, while ignoring the others will produce results that might be less optimal.

Another consideration in terms of weighting scheme for the scores, is that, in the future, these should not be taken globally for each entity in the triple candidate. Data publishers can have different requirements for the data model elements, considering, for example, that entity types should focus more on interoperability, while properties should be as accurate as possible. Therefore, in the future, these weights should be expanded to provided a more fine-grained control over the weights given to the scoring methodologies.

Overall, in this chapter, we proposed and described methodologies that are able to produce candidates and rank them according to different parameters to produce a ranked list of candidate triples to integrate a data model. We evaluated and described different experiments that show the effectiveness and potential of the methodologies described. Therefore, returning to the main question: *What measures can be used based on the built knowledge structures to select and rank possible data models?*

The answer is a combined methodology that focuses not only on the entities and on obtaining the most precise data model according to a background knowledge graph but also looks at the underlying ontology graph to produce a set of candidates that are interoperable with existing published datasets. Therefore, as an output of our framework, the user is presented with a ranked set of candidates and, depending on the use-case, it may be desirable to prioritise one parameter over another. For example, for the library use-case, if the goal is to maximise interoperability with other datasets, `foaf:Agent` is the best choice. On the other hand, `foaf:Person` provides higher specificity in the description of an entity, at the cost of potential interoperability. An application of the output of the framework is demonstrated with RICDaM, which shows the potential for a user interface that integrates the full framework.

5 BENCHMARKING ONTOLOGY RESOURCE RETRIEVAL METHODS

In this chapter, we explore the application of ontology resource retrieval tools and techniques for modelling data with ontologies. As described in Chapter 1, in the context of the Semantic Web, data modelling approaches can be categorised into internal or external, depending on the background knowledge considered when defining a data model. The framework proposed in Chapters 3 and 4 focuses on providing an external approach to data modelling by considering existing RDF datasets as the base for the data model. However, when the framework is unable to produce a complete data model, the data publisher needs to complement the approach by introducing concepts from internal approaches to data modelling. Internal approaches deal with finding ontology resources that produce a data model for an input dataset without directly considering background knowledge from external datasets. These approaches often rely on the data publisher knowing or finding the appropriate ontologies, which is not always a trivial task.

This chapter is guided by the following overall research question:

RQ3 → *Can existing methods for ontology resource retrieval support the process of creating a data model using an internal approach?*

This question is answered by dividing it into the following subquestions:

RQ3.1 → *Are IR algorithms effective for retrieving and ranking top-k ontology classes that match a given keyword?*

RQ3.2 → *Are ontology resource search engines effective for retrieving and ranking top-k ontology classes that match a given keyword?*

RQ3.3 → *What requirements should be considered when choosing the best technique to retrieve ontology resources when using internal approaches to create a data model?*

The contributions of this chapter include (1) a detailed comparison of ontology resource retrieval techniques and tools, evaluated with an expert-based ground truth and an automatically generated ground truth, (2) a set of recommendations geared specifically towards data modelling with ontologies in the biomedical domain, (3) a set of recommendations on using the techniques and tools evaluated to support or complement the creation of a data model with an external approach.

5.1 INTRODUCTION

Ontologies are a key technology to integrate knowledge on the Web since they tackle data organisation challenges by unequivocally representing concepts and, therefore, optimising the structural and semantic integrity in and between datasets. When used to model a Knowledge Graph, ontologies enable more accurate and effective information retrieval, data integration, decision support, and reasoning [182]. However, creating an ontology to model a domain is not a trivial task and requires significant investment of time and effort, and an expert with specific domain knowledge and ontology engineering experience. Instead of creating new ontology models from scratch, however, ontology engineers are encouraged to reuse existing ontologies [183, 184] by modifying, extending, or pruning them to adapt to a specific use-case. Ontology reuse is defined as *the process in which existing ontological knowledge is used as input to generate new ontologies* [185]. Ontology reuse provides a machine- and human-understanding of the agreement on the description of concepts in a domain, enabling the design of interoperable applications, while reducing the cost of data modelling with ontologies. The ontology reuse approach is faced with different challenges [186], and several independent studies have found its application to be lacking to fulfil its interoperability goals [48, 63–65, 187], leading to high overlap between the concepts described by ontologies in the same domain. Nonetheless, data publishers using internal approaches rely on ontology reuse to guarantee a conceptual agreement with other datasets even if the extent of this agreement is not directly known.

A key barrier for a data publisher, therefore, is to find the right set of ontologies to model datasets. When looking for appropriate ontologies to model data, the three main challenges are [21]: (1) finding ontologies in the right subject domain, (2) from these domain ontologies, assessing which ones provide sufficient coverage of the concepts of the intended application and are of good quality, and (3) assessing if the ontology is in the format required by the application (e.g., OWL format or SPARQL endpoint). Ontology repositories aim to facilitate these challenging tasks by providing an environment where ontologies can be deposited, accessed, discovered, and potentially reused.

An ontology repository is a *Web-based system that provides access to an extensible collection of ontologies with the primary purpose of enabling users to find and use one or several ontologies from this collection* [21]. A detailed overview of the proposed ontology repository solutions is found in d'Aquin and N. F. Noy [21]. Following this definition, by itself, an ontology repository provides the means to find ontologies, possibly in a relevant domain, but it does not facilitate the process of selecting the most appropriate ontologies or specific ontology resources (i.e. classes or properties) to model a dataset. A considerable portion of ontology repositories, however, also provide AOR services, i.e., search and ranking of ontologies and/or ontology resources. However, often the search results over the ontology repositories are overwhelming, with dozens of synonyms matching in different ontologies, as well as a common disagreement between search engines in the ranking of ontological resources in their search results. Due to different naming conventions, textual descriptions, synonyms, and granularity of the entities, it is an open research problem to precisely identify an ontological resource which best describes a given concept.

Among the different domains that adopted ontologies, the biomedical domain has a long history of using formal codes and ontologies to describe datasets [188, 189]. This domain has been one of the early adopters of Semantic Web technologies, resulting in the development of several biomedical repositories and ontologies. Biomedical ontologies are distinguished from other domain ontologies because they are typically large [182], covering thousands of concepts represented by the same number of classes (e.g., the Gene Ontology [58] has almost 50K classes). Most ontologies in this domain use a rich vocabulary in labels, synonyms, and textual definitions associated with classes and properties [55]. For example, the class `http://purl.obolibrary.org/obo/GO_1905294` has (1) a preferred label: *positive regulation of neural crest cell differentiation*; (2) a textual definition: *any process that activates or increases the frequency, rate, or extent of neural crest cell differentiation*; and (3) synonyms: *up regulation of neural crest cell differentiation; up--regulation of neural crest cell differentiation; upregulation of neural crest cell differentiation; activation of neural crest cell differentiation*.

Therefore, this domain posed an interesting use-case to assess the search and ranking capabilities of different approaches when finding the best ontology resource candidates that match a set of keywords. In this chapter, therefore, we test state-of-the-art IR algorithms, ontology ranking approaches, and established search engines for searching and ranking resources in biomedical ontologies. The algorithms and search applications/engines are tested by searching a defined set of queries obtained from a cancer genomics scenario. Using these queries, we established a Ground Truth (GT) by asking ten biomedical and ontology engineering experts to manually rank the search results for each query. Since building a manual ground truth is an expensive process, we also created an automated Probabilistic Ground Truth (PGT). The PGT led to a better understanding of the GT and allowed the expansion of the ontology collection and the number of queries tested. We then evaluated the results of the algorithms and applications using the ground truths with P@k, Average Precision@k (AP@k), MAP, and NDCG. This evaluation provided the necessary knowledge to explore the advantages and disadvantages of using each technique studied, which resulted in a set of recommendations that can be followed when searching for ontology resources to model a dataset. Furthermore, these recommendations can be extrapolated and applied to the case of supporting an external data modelling approach, such as the one proposed in this thesis.

5.2 BACKGROUND AND RELATED WORK

Ranking ontological resources can be based on different criteria, for example, how well an ontology meets the requirements of certain evaluation tests [190] or on methods to evaluate general properties of an ontology based on some requirements [191]. However, only limited work has been proposed to rank resources based on user-posed queries. AKTiveRank [192] is a system that uses structural metrics (i.e., Semantic Similarity, Betweenness, Density, and Class Match Measure [192]) to evaluate different representational aspects of an ontology and calculates its ranking in relation to a set of search queries specified by a user.

IR approaches have been quite successful in finding and ranking relevant documents. IR algorithms have been successfully applied in a few open-source indexing and search engines, such as Lucene[1], Solr[2], and ElasticSearch[3]. These applications include API's to provide an easy implementation and fast search. The user has control over most aspects of the inner workings of these applications and can adapt them to serve specific needs, e.g., ontology search. In the Web environment, IR search engines are primarily keyword-based and analyse the relevance of a document using content-based or graph-based methods. AOR engines focus on retrieving entities based on semantic information. For instance, Swoogle [90], Sindice.com [91], Watson [92], or Yars2 [93] allow searching for ontology resources through user queries.

General search services and algorithms have been developed for Linked Data applications, for instance, LOV[4], OntoKhoj [193], OntoSearch [194], or OntoSelect [195]. However, OntoSelect, OntoSearch, and OntoKhoj are not longer supported or available. OntoSelect provided an evaluation methodology [196] by creating a benchmark that associated topics from Wikipedia pages with ontologies and then compared the retrieval results of OntoSelect with Swoogle. However, the authors concluded that, on average, OntoSelect did not perform better than Swoogle.

LOVER [197] is an iterative search that supports the user in the process of finding the best classes and properties to model their dataset. Each choice made by a user provides context for the next iteration and LOVER supports the data publisher by following ontology data modelling recommendations when providing data model, entity type, and property candidates.

OntoCAT [198] provides uniform access for search across different public online repositories (BioPortal and OLS) but also allows the inclusion of local ontology files in standard OWL or OBO formats. This software is available as an R package [199] and is an easy method to programmatically search and integrate ontologies from different origins in the R environment.

[1] http://lucene.apache.org/

[2] http://lucene.apache.org/Solr

[3] https://www.elastic.co/

[4] http://lov.okfn.org/dataset/lov/

The biomedical community has made a significant effort to develop services such as BioPortal [83] and the Ontology Lookup Service (OLS) [84] for searching and applying ontological resources. However, they often suggest large, vague, or loose search results for a given query. Searching for the right concept in the most appropriate ontology is, therefore, a strenuous task since a significant number of available ontologies exist, in the same or in closely related domains that describe overlapping, closely related or the same concepts. BioPortal also developed a tool (Ontology Recommender[5]) that, from a set of keywords or a text, returns a set of ontologies that best cover the input considering a series of metrics, such as popularity and granularity of the ontology.

The CBRBench [200] is benchmark that was developed to evaluate the ranking of IR algorithms in an AOR setting. The evaluations are made with an expert-based ground truth where a series of queries were manually ranked and then compared with the ranking output of eight different search algorithms. We expand on this work by not only evaluating state-of-the-art IR algorithms but also include a standalone search engine (Solr), two ontology repositories, and one ontology recommendation tool. We also expand on the ground truth by going beyond the expert-based ground truth with an automatic ground truth that allows further conclusions to be drawn. Contrary to CBRBench, we limited our evaluation to the biomedical domain since it is an activate domain in the Semantic Web environment where a vast number of resources is available and keeps being developed.

Recently, another evaluation was performed on the ontologies included in the LOV repository [201]. This work proposes an evaluation based on user clicks to infer relevance labels. They provide a set of practical recommendations for ranking ontology in repositories. Similarly to this work, in this chapter, we provide a set of recommendations. However, they are focused on presenting guidelines to facilitate the choice of methodology to use when following an internal approach to data modelling, instead of focusing on improving the ranking of the ontologies in a repository.

5.3 ONTOLOGY SEARCH: APPLICATIONS & ALGORITHMS

For this benchmark, we compare seven IR algorithms (boolean, tf-idf, BM25, Vector Space Model (VSM), PageRank, Class Match Measure (CMM), and SMM), two search engines of ontology repositories (Bioportal and OLS), one standalone search engine (Solr), and a recommendation tool (Zooma). The online applications have APIs publicly available that were searched in the version available in December 2017.

[5] https://bioportal.bioontology.org/recommender

5.3.1 BioPortal

BioPortal is a repository containing both open-access and licensed biomedical ontologies and terminologies. Since its inception, the BioPortal library has grown substantially, from 72 ontologies in 2008 to over 200 in 2011 and, in 2017, to more than 500 ontologies. Besides being a repository for biomedical ontologies, BioPortal includes other resources and services. One of them is providing a search mechanism to find ontologies or ontology resources through keyword search. This search usually returns several matches and the results are ranked by the popularity (i.e., number of visits), in BioPortal of their source ontology.

5.3.2 Solr

Solr is a platform that extends the Apache Lucene search library for full-text indexing and search. One of Solr's major features is a REST-like API for easy integration with any programming language. The Lucene engine used by Solr scores documents using a combination of the Boolean model and a Vector Space Model algorithm. First, it uses the Boolean model to narrow down the number of documents it needs to score and then uses the VSM to attribute a final score to a document in relation to a user's query.

5.3.3 Ontology Lookup Service

The OLS is a repository for biomedical ontologies. As of January 2018, OLS had 206 ontologies and provided a search mechanism to match query words with ontological concepts. This search uses Apache Solr to index ontologies, but it applies specific boosts to some of the results, such as labels or ID exact matches.

5.3.4 Zooma

Zooma[6] provides mappings between a free-text input and a curated repository of annotation knowledge. This repository contains the annotations that were manually associated with data from sources such as the Expression Atlas [202] and the Genome-Wide Association Studies catalogue [203]. When no mappings are found in the curated data repository, OLS search is used instead to increase coverage. Due to its reliance on background knowledge, Zooma can considered an external data modelling approach that facilitates internal approaches. However, Zooma finds matches through keyword search and, therefore, does not take into consideration the relationships in the original data-

[6] http://www.ebi.ac.uk/spot/zooma/

set, finding correspondences for entity types with an internal/external hybrid approach limited to the data sources loaded in the tool.

5.3.5 IR Algorithms

Similarly to [200], we implemented seven commonly used ranking algorithms for documents and adapted them to give a free-text query, to rank resources in a collection of ontologies. For content-based algorithms (i.e., tf-idf, BM25, VSM, and CMM), instead of using words as the base unit, we considered a resource r (class or property) in the ontology as the measuring unit. A resource is matched to a query if any of the query words exist in the values for the label, synonyms, or description. When we wish to retrieve only exact matches, the query words have to be strictly the same as the value matched from the label, synonym, or description of a resource. The graph-based models (PageRank and Semantic Similarity) do not consider properties, only classes. However, less than 1% of all resources in the collection are properties.

Table 5.1: Notation used

Variable	Description
$\mathbb{O}$	Ontology collection
N	Number of ontologies in $\mathbb{O}$
O	An ontology: $O \in \mathbb{O}$
R	Collection of all resources (i.e., classes and properties) with $R \in O$
r	A resource : $r \in O$ & $r \in R$
Q	Query String
q_i	Query word i of Q
σ_O	Set of matched resources r for Q in O
$\sigma_O(q_i)$	Set of matched resources r for q_i in $O : \forall\, r_i \in \sigma_O$, $r_i \in O$ & r_i matches q_i

Table 5.1 lists the formal notations applied in the description of the algorithms. The following sections describe the algorithms with their adaptation for ranking ontologies.

Boolean Model

The Standard Boolean model is based on boolean algebra, where a query is viewed as a Boolean expression. Therefore, for a set of ontologies and queries, the retrieval is binary and based on whether or not the retrieved results contain the query words.

tf-idf

Term frequency-inverse term frequency (tf-idf) [204] quantifies how important a term is in an ontology by analysing the frequency of the term in the resources of that ontology and in the overall collection of ontologies.

$$
\begin{aligned}
\text{tf}(r, O) &= 0.5 + \frac{0.5 \cdot f(r, O)}{max\{f(r_j, O) : r_j \in O\}} \\
\text{idf}(r, \mathbb{O}) &= \log \frac{N}{|\{O \in \mathbb{O} : r \in O\}|} \\
\text{tf-idf}(r, O, \mathbb{O}) &= \text{tf}(r, O) \cdot \text{idf}(r, \mathbb{O})
\end{aligned} \tag{5.1}
$$

Here tf(r, O) is the term frequency of r in O obtained by dividing the frequency of r by the maximum frequency of any resource $r_j \in O$. The inverse document frequency $idf(r, \mathbb{O})$ is a measure of commonality of a resource across the collection. It is obtained by dividing the total number of ontologies in the collection, N, by the number of ontologies containing the resource r and then computing the logarithm of that quotient. The final tf-idf of r is the product of the tf and the idf.

BM25

BM25 [205] is a weighting scheme that takes into account not only term frequency, but also ontology size without introducing too many additional parameters in relation to tf-idf. Usually the BM25 score is computed for $\forall q_i \in Q$, but, to tailor this statistic for ontology ranking, we compute the sum of the score of each $r_j \in \sigma_O(q_i)$ for each query term q_i. Therefore, given a resource $r \in \sigma_O(q_i)$, with a value (e.g., label) containing the words r_1, ..., r_n, the BM25 score of the ontology O is computed by:

$$
score(O, Q) = \sum_{j=1}^{n} idf(r_j, \mathbb{O}) \frac{tf(r_j, O) \cdot k + 1}{tf(r_j, O) + k \cdot \left(1 - b + b \cdot \frac{|O|}{avgol}\right)} \tag{5.2}
$$

where $tf(r_j, O)$ is the term frequency for the matched resource r_j in the ontology O and $idf(r_j, \mathbb{O})$ is the inverse document frequency of the resource $r_j \in \sigma_O(q_i)$. |O| is the total number of resources (i.e., 3 × |axioms|) in the ontology, and $avgol$ is the average ontology size in the ontology collection. k and b are free parameters, usually chosen in the absence of an advanced optimisation, as $k \in$ [1.2,2.0] and b = 0.75. For the current implementation, we used k = 2.0, b = 0.75.

Vector Space Model (VSM)

A Vector Space Model [206] assumes that ontology resources and queries can be represented by the same type of vector. Non-binary weights are assigned to indexed terms, usually using weighting schemes such as tf-idf. The degree of similarity between query and ontology resources is calculated by comparing the vectors that represent the query and each ontology resource. The VSM score was calculated as follows:

$$
sim(O, Q) = \frac{\sum_{i=1}^{n} w(q_i, O) \cdot w(q_i, Q)}{|O| \cdot |Q|} \tag{5.3}
$$

Here $w(q_i, O)$ and $w(q_i, Q)$ are the weights of q_i in the ontology O and query Q, respectively. |O| is the ontology vector norm and |Q| is the query vector norm. For this implementation, we consider tf-idf as the vector weight. Therefore, the similarity of an ontology to query Q is computed as:

$$\begin{aligned}
sim(O, Q) &= \frac{\sum_{i=1}^{n} \text{tf-idf}(q_i, O) \cdot \text{tf-idf}(q_i, Q)}{|O| \cdot |Q|} \\
\text{tf-idf}(q_i, O) &= \sum_{j=1}^{z} \text{tf-idf}(r_j, O) : r_j \in \sigma_O(q_i) \\
\text{tf-idf}(q_i, Q) &= \sum_{j=1}^{n} \text{tf-idf}(q_i, Q) : q_i \in Q \\
|O| &= \sqrt{\sum_{j=1}^{z} (\text{tf-idf}(r_j, O))^2} \\
|Q| &= \sqrt{\sum_{i=1}^{n} (\text{tf-idf}(q_i, Q))^2}
\end{aligned} \tag{5.4}$$

PageRank

PageRank [207] is an iterative method to analyse links and it was adapted to assign a numerical score to each ontology in a set of ontologies. This implementation considers ontologies as nodes and `owl:imports` (imports of other ontologies into the current ontology, i.e., outlinks) as edges. In each successful iteration, the score of the ontology O is determined as the sum of the PageRank score of the previous iterations of all the ontologies that import ontology O divided by their number of outlinks. For the k_{th} iteration, the PageRank score of ontology O is given by:

$$\begin{aligned}
score_k(O) &= \frac{\sum_{j \in deadlinks(\mathbb{O})} PR_{k-1}(j)}{N} + \\
&+ \sum_{i \in inlinks(O)} \frac{PR_{k-1}(i)}{|outdegree(i)|} \\
score_k(O) &= d \cdot score_k(O) + \frac{1-d}{N}
\end{aligned} \tag{5.5}$$

Here, $deadlinks(\mathbb{O})$ are ontologies in the collection that have no outlinks. All nodes are initialised with an equal score (i.e., $\frac{1}{N}$, where N is the total number of ontologies in $\mathbb{O}$ before the first iteration). In the experimental evaluation, we set the damping factor d equal to 0.85 (common practice).

Ontologies with no `owl:imports` statement can still reuse classes from other ontologies following the MIREOT [208] guidelines for referring external terms from a target ontology. In our experiment, whenever these references existed within the ontology classes, an `owl:imports` statement was introduced to identify the link between the two ontologies.

Class Match Measure (CMM)

Class Match Measure [192] calculates the coverage score of an ontology in relation to a set of given queries. Despite not ranking each query individually, this algorithm represents the type of search one could expect from a user that requires the lowest number of ontologies to cover all the queries in their search.

The CMM algorithm looks for exact and partial matches and scores an ontology depending on the number of matches. A higher number of matches means a higher CMM score. The score for an ontology is computed as:

$$score_{CMM}(O, \mathbb{Q}) = \alpha score_{EMM}(O, \mathbb{Q}) + \beta score_{PMM}(O, \mathbb{Q}) \tag{5.6}$$

where $score_{CMM}(O, \mathbb{Q})$ is the final score for class match measure, $score_{EMM}(O, \mathbb{Q})$ is the exact match measure, and $score_{PMM}(O, \mathbb{Q})$ is the partial match measure for the ontology O with respect to the set of queries $\mathbb{Q}$. α and β are the exact matching and partial matching weight factors respectively. Exact matching is favoured over partial matching, therefore $\alpha > \beta$. Here α = 0.6 and β = 0.4.

$$\begin{gathered}
score_{EMM(O,\mathbb{Q})} = \sum_{r \in O} \sum_{Q \in \mathbb{Q}} \varphi(r, Q) \\
\varphi(r, Q) = \begin{cases} 1 & \text{if } label(r) = Q \\ 0 & \text{if } label(r) \neq Q \end{cases} \\
score_{PMM}(O, \mathbb{Q}) = \textstyle\sum_{r \in O} \sum_{Q \in \mathbb{Q}} \psi(r, Q) \\
\psi(r, Q) = \begin{cases} 1 & \text{if } label(r) \text{ contains } \forall q_i : q_i \in Q \\ 0 & \text{if } label(r) \text{ does not contain } q_i \in Q \end{cases}
\end{gathered} \tag{5.7}$$

$\varphi(r, Q)$ counts the number of exacts matches and $\psi(r.Q)$ counts the number of partial matches. $score_{EMM(O,\mathbb{Q})}$ and $score_{PMM}(O, \mathbb{Q})$ sums the number of matches (exact and partial, respectively) that exist in every ontology for a set of queries.

Semantic Similarity Measure (SSM)

The Semantic Similarity Measure [192] applied here takes advantage of the ontological graph structure to calculate how close resources are in the ontology structure. It is a collective measure of the shortest path lengths for all classes that match the query string.

The semantic similarity measure score $score_{SSM}(O, Q)$ of ontology O for a given query Q is given by:

$$
\begin{gathered}
score_{SSM}(O, Q) = \frac{1}{z} \sum_{i=1}^{z-1} \sum_{j=i+1}^{z} \Psi(r_i, r_j) : \forall_{q \in Q}((r_i, r_j) \in \sigma_O)) \\
\Psi(r_i, r_j) = \begin{cases} \frac{1}{length(min_{p \in P}\{r_i \xrightarrow{p} r_j\})} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \\
z = |(r_i, r_j)|
\end{gathered}
\tag{5.8}
$$

Summary

Table 5.2 presents a summary of the characteristics of the algorithms. The table shows: (1) the main scoring mechanism of each algorithm, (2) if the algorithm attributes a global score to the ontology or scores each resource in the ontology individually, (3) if there is any distinction between partial matches and exact matches (yes) or if they are treated equally (no) and finally, and (4) a summary of the conclusions presented in Butt, Haller and Xie [200].

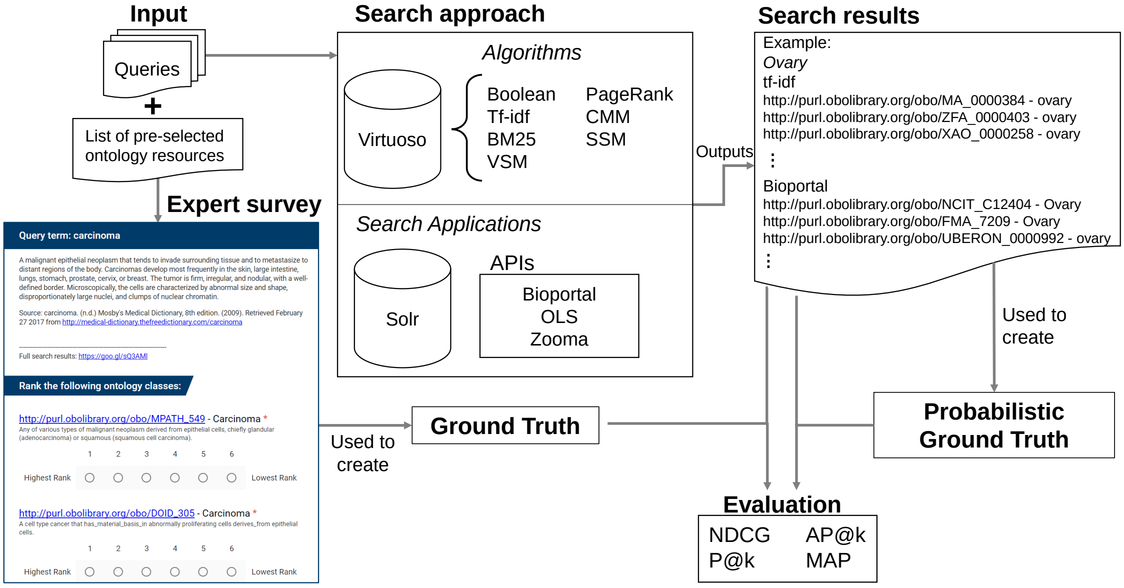
Table 5.2: Summary of IR algorithms. Scoring summarises the main scoring method of the algorithm. Global indicates if the score attributed by the algorithm is per resource or per ontology. WPM (Weights Partial Matches) shows if the ontology distinguishes between partial and exact matches

Algorithm	**Scoring**	**Global**	**WPM**	**Remarks**
tf-idf	Term frequency	No	No	Frequent resources in the collection have a low score. In ontologies, a common term does not necessarily mean less relevant. Frequent terms can be a product of reuse by other ontologies.
BM25	Term frequency	Yes	No	Suffers from the same issue has tf-idf but the cumulative score ranks domain ontologies higher.
VSM	Vector similarity	No	No	Uses tf-idf to weight vectors and considers the tf-idf of the query, aggravating the tf-idf drawback.
PageRank	Links between ontologies	Yes	No	Ranks based on popularity which may lead to popular but less relevant resources being ranked higher.
CMM	Coverage of the set of queries	Yes	Yes	Ontologies with a large number of partial matches will be scored higher than ontologies with few exact matches.
SMM	Closeness between ontological resources	Yes	No	Although this algorithm can be useful when considering similarity among the matched resources of two or more query terms of a multi-keyword query, it performs poorly on single word queries.

5.4 EVALUATION: ONTOLOGY SEARCH APPLICATIONS & ALGORITHMS

The workflow of the benchmark was divided into two separate but comparable analyses that give a complete overview of the performance of the chosen applications and algorithms. The first approach evaluated the results against a GT obtained from a questionnaire answered by experts. The second analysis was based on an automated PGT obtained from the consensus between the algorithms and four search applications. Figure 5.1 illustrates workflow from the queries to the different algorithms/applications and presents an example of the search results. The figure shows the evaluation process starting from the creation of the GT and the PGT and their comparison with the search results, and finally obtaining the evaluation results.

We evaluate the algorithms and tools with P@k, AP@k, MAP, and NDCG (see Section 2.3.3 for more details) using the GT and PGT. Both ground truths have a number of defined relevant search results that vary for each query. For example, in the GT, the query *MYH7* only had one relevant result while the query *Ovary* had five. In terms of metrics, this difference means that, with a fixed $k = 3$, if a search of the query *MYH7* returned more than one result, the precision would be lower than expected, even if the first result was the correct one. Therefore, instead of choosing a fixed cut-off, the parameter k is chosen independently for each query and each ground truth depending on the number of search results present in the respective ground truth. Therefore, the query *MYH7* was evaluated with $k = 1$ while the query *Ovary* had $k = 5$. This adaptation evaluates if the algorithms and search applications return all the relevant results in the first k positions.


Input
Queries
+
List of pre-selected ontology resources
Expert survey
Search approach
Algorithms
Virtuoso
Boolean
Tf-idf
BM25
VSM
PageRank
CMM
SSM
Search Applications
Solr
APIs
Bioportal
OLS
Zooma
Outputs
Search results
Example:
Ovary
tf-idf
http://purl.obolibrary.org/obo/MA_0000384 - ovary
http://purl.obolibrary.org/obo/ZFA_0000403 - ovary
http://purl.obolibrary.org/obo/XAO_0000258 - ovary
⋮
Bioportal
http://purl.obolibrary.org/obo/NCIT_C12404 - Ovary
http://purl.obolibrary.org/obo/FMA_7209 - Ovary
http://purl.obolibrary.org/obo/UBERON_0000992 - ovary
⋮
Used to create
Probabilistic Ground Truth
Ground Truth
Used to create
Evaluation
NDCG
AP@k
P@k
MAP

Query term: carcinoma

A malignant epithelial neoplasm that tends to invade surrounding tissue and to metastasize to distant regions of the body. Carcinomas develop most frequently in the skin, large intestine, lungs, stomach, prostate, cervix, or breast. The tumor is firm, irregular, and nodular, with a well-defined border. Microscopically, the cells are characterized by abnormal size and shape, disproportionately large nuclei, and clumps of nuclear chromatin.

Source: carcinoma. (n.d.) Mosby's Medical Dictionary, 8th edition. (2009). Retrieved February 27 2017 from http://medical-dictionary.thefreedictionary.com/carcinoma

Full search results: https://goo.gl/sQ3AMl

Rank the following ontology classes:

http://purl.obolibrary.org/obo/MPATH_549 - Carcinoma *
Any of various types of malignant neoplasm derived from epithelial cells, chiefly glandular (adenocarcinoma) or squamous (squamous cell carcinoma).

	1	2	3	4	5	6	
Highest Rank	○	○	○	○	○	○	Lowest Rank

http://purl.obolibrary.org/obo/DOID_305 - Carcinoma *
A cell type cancer that has_material_basis_in abnormally proliferating cells derives_from epithelial cells.

	1	2	3	4	5	6	
Highest Rank	○	○	○	○	○	○	Lowest Rank

Figure 5.1: Evaluation workflow: from input search queries to evaluation results

5.4.1 Ontology Loading

The ontology resources were stored using the Virtuoso triplestore and, in total, defined around 20M triples and 645K distinct classes (subjects of `rdf:type owl:Class`). In Solr, ontologies were loaded using the method provided by the OLS development team,[7] which uses the owlapi Java API [209] to manipulate the provided ontologies formatted with the Web Ontology Language (OWL)[8].

5.4.2 Building the Expert-Based Ground Truth

The ground truth was established with a study[9] involving ten experts that were asked to rank the ontology resources matched with the ten query terms.

Ontology Collection & Search Queries

A collection of 23 ontologies (see Table 5.3) representative of different domains in the biomedical field was used. The domains chosen range from chemical compounds to diseases or phenotypes, among others. The collection also includes different species such as mouse and zebrafish. The set of ontologies has some of the most popular and freely accessible biomedical ontologies, with more than half of them included in the top-50 of BioPortal's most visited ontologies (as of December 2017).

The searches tried to match each query with all ontological resources available in each platform, i.e., online applications used their services and local tests used the Virtuoso database or local Solr server. When using the search applications, results that included ontologies outside this list were excluded. The search applications and the ranking algorithms were tested using a set of queries in the domain of ovarian cancer. Table 5.4 presents the ten search terms chosen and the abbreviations used for the remainder of this chapter. Although the ten selected search terms are from the ovarian cancer domain, they represent several sub-domains not only related to ovarian cancer. For instance, these search terms are classified into five different types of sub-domains (disease, drug, tumour, organ, and gene) covering a broad set of terminologies.

Their general frequency was assessed with a Google search, which showed that *Carcinoma* and *Ovary* were the queries with most results in the set, disease names were less common, and the gene name *MYH7* was the query with the least search results, due to its specificity. Finally, all the queries were used as input in BioPortal and OLS' Web search, with the default parameters (search through all the ontologies and show exact and partial matches).

[7] `https://github.com/EBISPOT/OLS/tree/master/ols-apps/ols-solr-app`

[8] `https://www.w3.org/OWL`

[9] The questionnaire is available at `https://goo.gl/pQUvte`

Table 5.3: Ontologies used in the GT with name, acronym, number of triples, and reference

Name	Acronym	# Triples
Chemical Entities of Biological Interest Ontology [210]	ChEBI	8 187 078
Cell Ontology [136]	CL	69 796
Human Disease Ontology [211]	DOID	203 125
The Drug Ontology [212]	DRON	138 898
EMBRACE Data And Methods [213]	EDAM	33 300
Experimental Factor Ontology [214]	EFO	469 954
Foundational Model of Anatomy [215]	FMA	612 982
Gene Ontology [58]	GO	1 575 776
Human Phenotype Ontology [216]	HP	350 017
Mouse Adult Gross Anatomy Ontology [217]	MA	25 523
Mammalian Phenotype Ontology [218]	MP	335 821
Mouse Pathology Ontology [219]	MPATH	11 992
Neuro Behavior Ontology [220]	NBO	10 376
National Cancer Institute Thesaurus [221]	NCIT	5 784 846
Ontology of Adverse Events [222]	OAE	54 334
Ontology of Genes and Genomes [223]	OGG	1 211 539
Phenotypic Quality Ontology [224]	PATO	31 644
Plant Ontology [225]	PO	59 932
Uber Anatomy Ontology [226]	UBERON	690 529
Vertebrate Trait Ontology [227]	VT	44 183
C. elegans Phenotype Vocabulary [228]	WPhenotype	31 991
Xenopus Anatomy and Development Ontology [229]	XAO	40 611
Zebrafish Anatomy and Development Ontology [230]	ZFA	82 964

Table 5.4: Cancer-related queries and their number of search results on Google, BioPortal and OLS in April 2017

Query Terms	Abbreviation	Type	Google	BioPortal	OLS
Ovary	Ovary	Organ	25 400 000	29	1054
MYH7	MYH7	Gene	86 500	8	22
Paclitaxel	Paclitaxel	Drug	4 640 000	18	149
Carcinoma	Carcinoma	Disease	32 800 000	25	4025
Carboplatin	Carboplatin	Drug	2 710 000	19	212
Ovarian teratoma	OT	Tumour	434 000	18	1164
Ovarian cystadenoma	OCys	Tumour	148 000	18	1100
Ovarian Choriocarcinoma	OChor	Tumour	317 000	20	1129
Ovarian embryonal carcinoma	OEC	Tumour	164 000	19	5069
Ovarian mucinous adenocarcinoma	OMA	Tumour	117 000	15	2235

Experts

The experts were sourced from IBM Research, USA; King Abdullah University of Science and Technology, Kingdom of Saudi Arabia; Maastricht University, Netherlands; Medical University of Graz, Austria; Indian Institute of Technology (IIT) Bombay, India; Saarland University, Germany; Universite de Rennes 1, France; and the U.S. National Library of

Medicine, USA. The areas of expertise of these judges included knowledge engineering and the biomedical domain, with all of them having at least some experience in both domains.

Table 5.5: Level of self-assessed knowledge of the experts in the biomedical field. BD refers to Biomedical Data

Expert	Biomedical Knowledge	Works with BD	Produces BD	Applies BD
1	5	Yes	Yes	Yes
2	3	Yes	Yes	Yes
3	5	Yes	No	Yes
4	4	Yes	No	Yes
5	5	Yes	Yes	No
6	4	Yes	No	Yes
7	5	No	No	Yes
8	4	Yes	Yes	No
9	5	Yes	Yes	Yes
10	5	No	No	Yes

To assess the level of experience in the biomedical and knowledge engineering fields, each expert was asked to rate his knowledge in a Likert scale of five levels, from "No Knowledge" to "Expert Knowledge". Table 5.5 and Table 5.6 show that most experts have considered themselves to have strong to medium knowledge in both domains, and apply or work with biomedical data. All the judges have worked with ontologies, six worked specifically with biomedical ontologies, and some have developed ontologies.

In the biomedical domain, six of the evaluators considered themselves to have "Expert Knowledge", three considered themselves to have "Strong Knowledge" and one had "Some Knowledge". Nine of the experts work with biomedical data generated by others and seven of them apply biomedical data in their work. Five of the judges generate biomedical data with their research. Regarding knowledge engineering experi-

Table 5.6: Level of self-assessed knowledge of the experts in the knowledge engineering field. Ont. means Ontology and BmO is Biomedical Ontology

Expert	Knowledge Engineering	Worked with Ont.	Developed a BmO
1	2	Yes	No
2	5	Yes	Yes
3	5	Yes	Yes
4	5	Yes	Yes
5	4	Yes	No
6	5	Yes	Yes
7	3	Yes	Yes
8	3	Yes	No
9	5	Yes	Yes
10	5	Yes	Yes

ence, in a scale from "No Experience" to "Expert", six of the judges consider themselves experts, one considered himself to have above average experience, two have average experience, and one evaluator has below average experience. All the judges have worked with ontologies and six worked specifically with biomedical ontologies. Seven of the experts have developed biomedical ontologies.

Questionnaire

The experts were given a list of ontology classes to rank in relation to ten queries (see Table 5.4). These classes were obtained by searching BioPortal and OLS and from the search results of the IR algorithms. All results were merged and taken out of order. The results presented to the judges were composed by the classes with labels that were an exact match of the query terms. However, the judges still had access to all the retrieved classes and could introduce classes they deemed relevant but were not shown. The questionnaire presented definitions from medical dictionaries to establish the search intention of each query. The definitions did not follow ontological definitions patterns. Their main goal was to lower the ambiguity of the queries and to provide some guidance to the judges. For each of the classes displayed in the questionnaire, we provided the class URI, preferred label and definition. The judge could rank the classes in a Likert-type scale with a number of options equal to the number of items to rank, with the first option corresponding to the best rank, plus the last rank reserved to mark the search term as "not relevant".

Validation

The questionnaire answers of each judge were evaluated with two different approaches:

1. Rank agreement considers the observed agreement between the ranks allocated to each search result.

2. Relevancy agreement analyses the results in terms of the observed agreement in a binary scale of relevant/not-relevant.

For each approach, the answers of each judge were compared in pairs and the final result was obtained by averaging the pairwise agreement.

The answers were further analysed with a Chi-Square Goodness-of-Fit test [231], which is a non-parametric statistical test to determine if an observed value is significantly different than its theorised value. Therefore, this test was applied to assess the randomness of the expert's answers. The null hypothesis considered was "each rank has an even number of answers", which leads to the alternative hypothesis "the ranks are not equally chosen among experts". This test was performed for each query with a significance level of 0.05.

5.4.3 Building the Probabilistic Ground Truth

To validate the ground truth as well as to include a more diverse set of queries and ontologies, we extended the approach used in Lamiroy and Sun [232] to build a PGT without involving human experts. In Lamiroy and Sun [232], the authors present an approach to calculate the probability that each document in a collection belongs to a possible ground truth by using the consensus from multiple systems as the reference. The main idea in Lamiroy and Sun [232] is that a probabilistic ground truth is equivalent to an unknown ground truth and contains the probabilities of each document (δ_i) that appears in the search results to belong to a real ground truth. These probabilities are calculated with the following:

$$P(\delta_i) = \frac{1}{s} \sum_{k=1}^{s} S_k(\delta_i) \tag{5.9}$$

where s is the set of systems being tested and $S_k(\delta_i)$ represents the search result of document δ_i by system S_k. Lamiroy and Sun [232] consider the result to be binary, i.e., the document δ_i is either present or absent of the search results of system S_k. By using these probabilities, the authors then calculate a probabilistic precision and recall. For our work, however, a comparison with the GT was necessary. Therefore, we developed an extended probabilistic ground truth which can be compared with the expert-based ground truth using the performance measures.

Extending and building the PGT

To extend the original approach to the ontology context, instead of using ontologies as documents, each ontology resource was considered the measuring unit. This adaptation allows the ranking of several resources matched in the same ontology.

Using the principle of the Discounted Cumulative Gain, a ranking approach was added to the creation of the PGT to obtain a non-binary classification of each resource. The method employed a penalty measure – in a logarithmic scale [233] – for search results that appear lower on the list. The penalty is proportional to their position p. Considering all search results relevant, the Discount Metric (DM) is obtained with:

$$DM_p = \frac{1}{\log_2(p+1)} \tag{5.10}$$

The probability of each ontology resource belonging to a possible ground truth is:

$$P(\delta_i) = \frac{1}{s} \sum_{k=1}^{s} DM_p(S_k) \tag{5.11}$$

Table 5.7: Expanded query set obtained from Gavankar, Y.-F. Li and Ramakrishnan [234]

Type	Queries
General	concentration unit, daily living, electron microscopy, health belief, health services, body weight, cell mass, cell proliferation, disease staging, dose response, clinical trial, compound treatment, differential scanning calorimetry, growth protocol, high performance liquid, high throughput, sequence alignment.
Cell or tissue	bone marrow, brown adipose, connective tissue, connective tissue development, granulosa cell, hemoglobin e.
Anatomy	collecting duct, digestive system, embryonic structure, frontal lobe, harderian gland, heart ventricle.
Genetic	copy number, gene expression phenotype, gene regulation, genetic modification.
Condition	convulsive status epilepticus, fatty liver, generalized anxiety, heart failure, heart rate, venous thrombosis.
Disease	breast cancer, eye disease, hemoglobin e thalassemia, hepatitis b, hepatitis c, ovarian cancer.
Disorders	cystathione synthase deficiency, dowling degos syndrome, epileptic encephalopathy, fever infection syndrome, goldstein hutt, nephrotic syndrome.

Resources with a $P(\delta_i) \leqslant 0.1$ were removed, which closely translates to at least one system ranking a resource as first. The remaining resources were sorted in descending order.

Ontology Collection & Search Queries

The PGT was compared against two sets of queries and ontologies. The first set included the same search queries and ontologies as the GT evaluation. The second was extended to include 51 extra queries obtained by Gavankar, Y.-F. Li and Ramakrishnan [234] from the BioPortal query log (see Table 5.7) and added 130 ontologies from the OBO Foundry[10], which, considering the previous 23 ontologies, led to a collection of 153 ontologies.

Since the method to obtain the PGT relies on the systems obtaining at least some relevant results for constructing the PGT, the search parameters were changed to return only exact matches.

5.4.4 Comparison between GT and PGT

The GT and the PGT were compared by calculating their relevancy agreement and Rank agreement, i.e., the agreement over which concepts are relevant and the agreement of the rank to attribute a resource, respectively. These comparisons were performed over the collection of ten queries and 23 ontologies used in the GT. The evaluation results of both ground truths were compared using the Pearson correlation coefficient and the

[10] `http://obofoundry.org`

linear distance between the values obtained for each query with each algorithm/search application. The results were then averaged for each performance metric.

5.5 RESULTS

The results are presented in relation to the GT, the comparison between the GT and the PGT, and the extended results using only the PGT.

5.5.1 Ground Truth Results

Figure 5.2 presents a box plot for each query with the respective answers that were included in the questionnaire and the distribution of rankings in the y-axis. The best ranking is 1 which the lowest ranking is equivalent to the experts considering the search result not relevant. The results for each box plot are ordered by the mean, with ties not yet resolved, meaning that the order from left to right corresponds to the order of search results in the GT. Overall, the queries show a high dispersion of answers. The query *MYH7* had only one search result and the opinions of the experts were divided equally between relevant and not relevant. None of the remaining means indicate that the judges, on average, considered a search result not relevant, i.e., the mean would have to be equivalent to the lowest rank. The experts did not suggest more classes to be added to the pre-selected classes.

Table 5.8 shows the rank obtained for the query *Carcinoma* by calculating the mean of the scores attributed to each class by the judges. If any mean calculation resulted in a draw, the final rank was obtained by searching the popularity of each ontology in BioPortal and ranking the most popular higher than the least popular. For example, the last two rankings in the Table 5.8 have a same mean value (3.2) and the EFO result was ranked above the MPATH due to EFO's higher popularity in BioPortal.

Table 5.8: Ranking of *Carcinoma* in the ground truth

Rank	Mean	URI
1	1.9	`obo:NCIT_C2916`
2	2.1	`obo:HP_0030731`
3	3.7	`obo:DOID_305`
4	3.2	`efo:EFO_0000313`
5	3.2	`obo:MPATH_549`

Validation

The observed agreement between the judges in relation to the rank was, in average, 30% with a standard deviation of 20%. The relevancy agreement was, in average, 85%, with

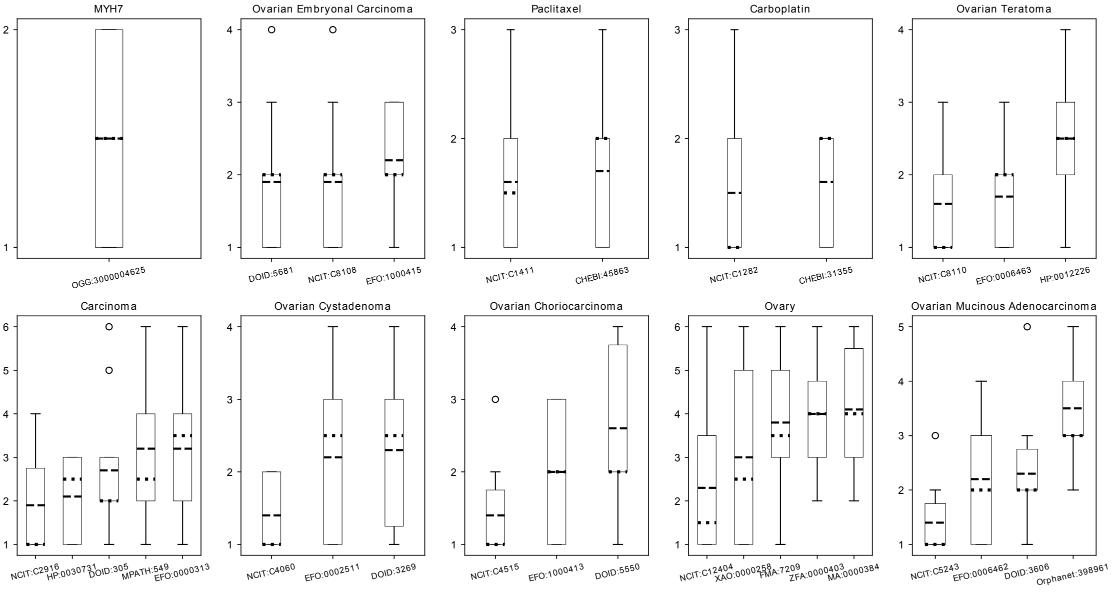


Figure 5.2: Box plot of the results of the ground truth questionnaire. The y axis displays the possible number of ranks for each item (*i.e* number of answers plus the additional not-relevant rank). The x axis shows the class id for each of the possible answers for the queries. The dotted line represents the median and the dashed line represents the mean by which the results were ordered

a standard deviation of 15%. These results show that the judges have a high level of agreement when considering which classes are relevant or not relevant but have a low agreement when ranking the ontology resources.

The expected values and the observed values of each query for the Goodness-of-Fit Chi-Square test are shown in Figure 5.3. The line represents the expected value, i.e., the value each rank should have if the rankings were equally chosen by the judges. The bars represent the actual number of times each rank was chosen. Except for the *MYH7* query, the ranks in all queries differ from the expected value.

The Chi-Square Goodness-of-Fit test indicated that, with $\alpha = 0.05$, half of the answers reject the null hypothesis and the other half accept it, which implies that there is some disagreement between the judges. The *MYH7* query had a χ^2 of zero due to the point previously raised of a single pre-selected relevant class. Queries with more general domains, such as *carcinoma* and *ovary*, have a lower p-value, which strongly suggests that the judges' responses are less likely to be random.

5.5.2 Comparison between GT and PGT

Figure 5.4 shows the relevancy and ranking agreement between the GT and the PGT. Except for *Ovarian Choriocarcinoma* (OChor), the ground truths agree about which search results are considered relevant. *Ovarian Choriocarcinoma* has a lower agreement due to the presence of the class *choriocarcinoma of ovary* (`obo:DOID_5550`) which is not an

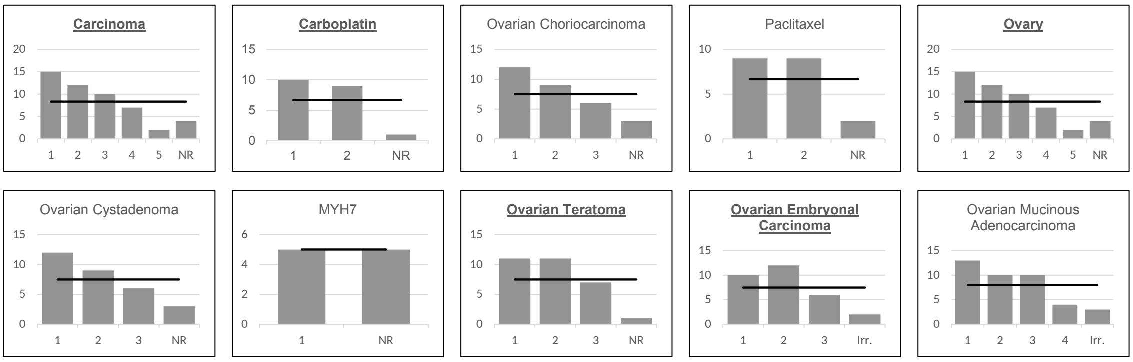


Figure 5.3: Goodness of fit Chi-Square expected and observed results, represented by a line and bars, respectively. Each chart contains a bar for the number of rankings available for each query and one extra one representing the ranking of "Not-Relevant" (NR). A bold and underlined query term indicates that the test rejected the null hypothesis, with $\alpha = 0.05$

exact match. Since the PGT was built based on exact matches only, this result was not considered and, therefore, is not featured on the final PGT. The rank agreement between ground truths is lower than the relevancy agreement with an average of 63% agreement between the ground truths.

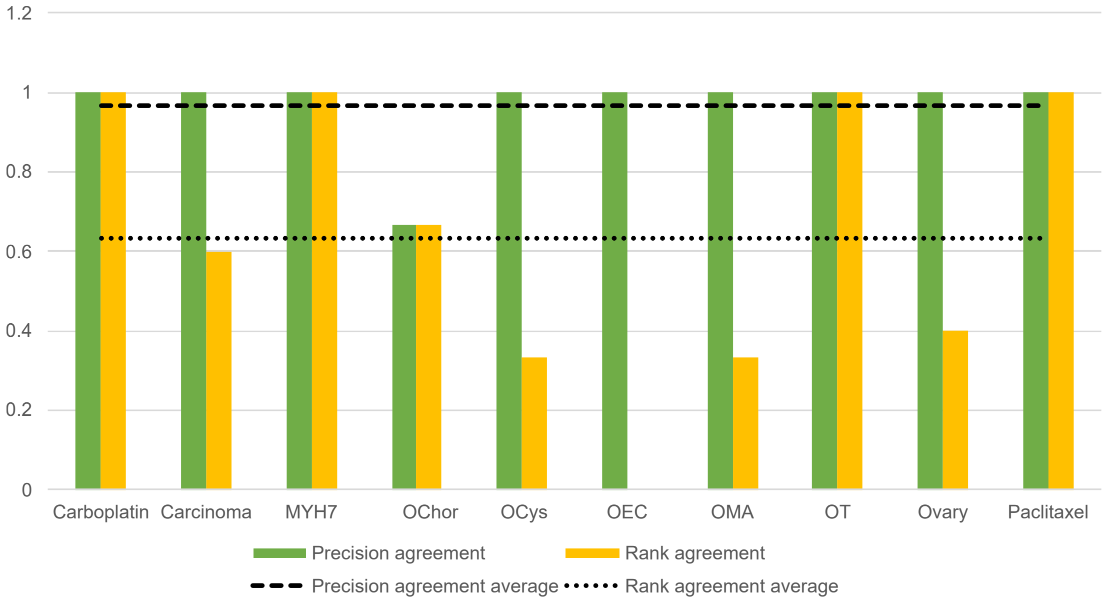


Figure 5.4: Relevancy and ranking agreement between the GT and the PGT

NDCG uses the ground truth ranking to compute the iDCG. However, none of the remaining metrics takes into account the ground truth ranking to evaluate the performance of the algorithm/system. Therefore, due to the high relevancy agreement between GT and PGT, P@k, AP@k, and MAP are considered more reliable than the NDCG when evaluating searches with the PGT.

5.5.3 Evaluation with performance metrics

The first evaluation of the algorithms and systems compares the search results in response to queries with the GT, followed by a similar comparison against the PGT. The next sections present the results of these evaluations and discuss their implications.

Against the GT

Tables 5.9 and 5.10 present the AP@3 and NDCG of each algorithm and application tested for the set of ten queries. Figure 5.5 shows boxplots for each of the metrics studied with consideration of partial and exact matches and 5.6 examines exact matches only.

Table 5.9: AP@3. The colours code the AP@3 values and range from dark green (highest AP@3, i.e., 1.0) to red (lowest AP@3, i.e., 0.0). The last column and last row represent the mean of each column/row, colour coded from blue (high mean) to light yellow (low mean)

	MYH7@1	Carboplatin@2	Paclitaxel@2	OChor@3	OCys@3	OTer@3	OMA@3	OEC@3	Carcinoma@5	Ovary@5	
OLS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.80	0.96
Bioportal	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.71	0.95
Solr	1.00	1.00	1.00	0.76	0.67	1.00	1.00	0.11	1.00	0.71	0.83
Zooma	0.00	0.50	0.50	1.00	1.00	0.33	0.33	1.00	0.20	0.00	0.49
tf-idf	1.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.22
pagerank	1.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.22
VSM	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10
boolean	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BM25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.45	0.45	0.45	0.34	0.33	0.30	0.30	0.28	0.25	0.24	

Table 5.10: NDCG. The colours code the NDCG values and range from dark green (highest NDCG, i.e. 1.0) to red (lowest NDCG, i.e. 0.0). The last column and last row represent the mean of each column/row, colour coded from blue (high mean) to light yellow (low mean)

	MYH7@1	Carboplatin@2	Paclitaxel@2	OCys@3	OChor@3	OEC@3	OMA@3	OTer@3	Carcinoma@5	Ovary@5	
OLS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00
Bioportal	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.95	0.98
Solr	1.00	1.00	1.00	0.95	0.76	1.00	1.00	1.00	1.00	0.83	0.95
Zooma	0.00	0.61	0.61	1.00	1.00	0.62	0.47	0.47	0.34	0.00	0.51
tf-idf	1.00	0.61	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.21	0.24
pagerank	1.00	0.61	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.21	0.24
VSM	0.63	0.61	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.19
SMM	0.63	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.08
boolean	0.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06
BM25	0.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06
CMM	0.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06
	0.74	0.49	0.49	0.36	0.34	0.33	0.33	0.32	0.29	0.29	

ALGORITHMS RESULTS ANALYSIS: Tables 5.9 and 5.10 show that for the algorithms most of the queries had an AP@3 and a NDCG equal to zero. The main contributors to these results were the partial matches. Since none of these algorithms (except CMM)

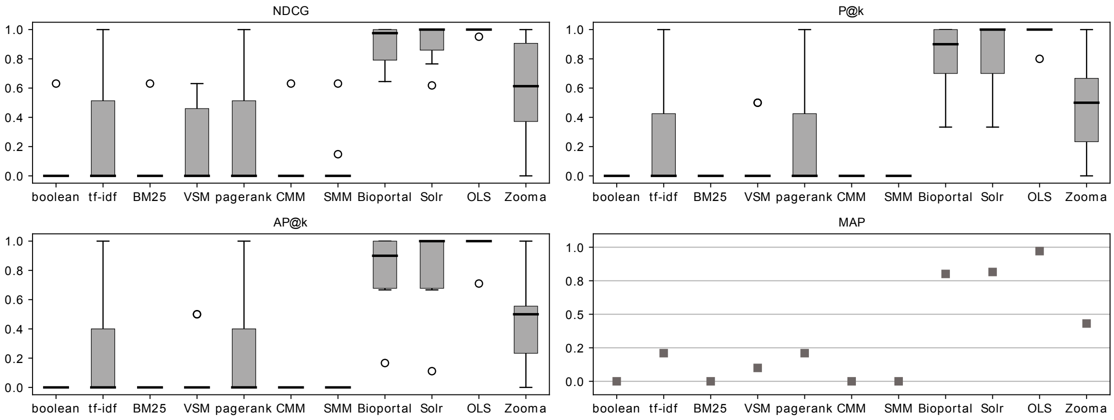


Figure 5.5: NDCG, P@k, AP@k and MAP results for the ten query collection, considering partial matches, against the GT

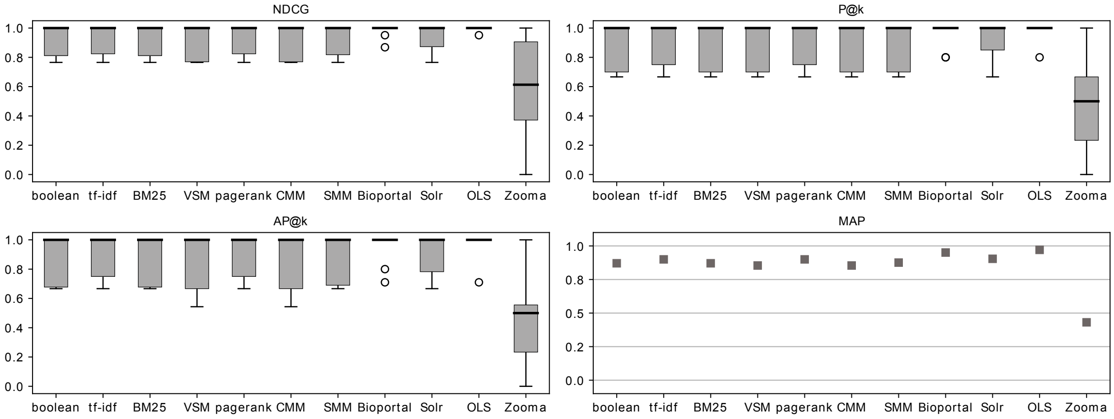


Figure 5.6: NDCG, P@k, AP@k, and MAP results for the ten query collection, considering exact matches only against the GT

weights partial or full matches differently, a class that contained any of the query words was considered a match and was ranked according to the algorithm. Most of the algorithms, except tf-idf and VSM, also rank matches globally, which leads to several non-relevant results having a high score due to the global score of the ontology. CMM is one of the algorithms that ranks ontologies globally, but since it evaluates the coverage of the set of queries by an ontology, it goes even further by considering the query set globally as well. *MYH7*, *Carboplatin*, and *Paclitaxel* achieved the best AP@3 and NDCG results due to their low number of possible matches.

The comparison of Figure 5.5 with Figure 5.6 shows that forcing exact matches considerably increases the performance of the IR algorithms. However, the consequence of this change is that the algorithms ignore synonyms and search applications ignore small edits between matches. By forcing exact matches, even the same label with a different order will not be returned by the algorithms. However, in the small scale considered, this was not a significant issue since only one query matched with a synonym (`efo:0002511`

- *simple cystadenoma*) and only one matched with a label with a different word order (obo:DOID_5550 - *choriocarcinoma of ovary*). Furthermore, when the ontology and query set is later expanded using the PGT (see Figure 5.8), the performance does not degrade substantially, meaning that the cases where this situation occurs are in the minority.

SEARCH APPLICATIONS RESULTS ANALYSIS: BioPortal focuses on precision, only showing the best hit in the ontologies with a match, while OLS focuses on recall, with the highest scoring terms ranked first, but also showing all possible partial matches. The search applications show high performance with a MAP of 0.97 for OLS, 0.82 for Solr, 0.80 for Bioportal, and 0.43 for Zooma.

NDCG evaluates the results considering not only the ontology classes present but also the position in which they appear. OLS achieved the highest NDCG performance with an average of 0.99 over all queries, BioPortal obtained an average NDCG of 0.90, and Solr of 0.92. Zooma obtained the lowest NDCG performance with an average of 0.58.

Figure 5.5 and Figure 5.6 show that the difference between the results with or without forcing exact matches does not have a major effect in the search applications tested since they were already ranking the exact and relevant matches in the first k positions. Zooma does not allow exact match search. Therefore, the results are the same in both figures.

Against the PGT

Figure 5.7 presents the results for the collection of ten query terms and 23 ontologies with forced exact matches against the PGT. Except for Zooma, the metrics show high performance for all algorithms and search applications. Overall, BioPortal and OLS slightly outperform the remaining methods with Zooma having the lowest performance in this setting.

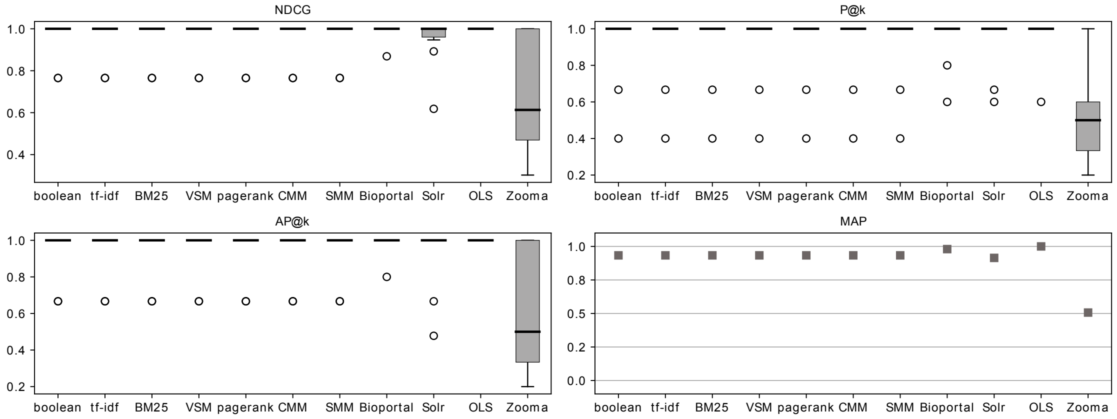


Figure 5.7: NDCG, P@k, AP@k and MAP results for the ten query collection, considering exact matches only, against the PGT

Figure 5.7 is directly comparable with Figure 5.6 since it used the same search parameters but the results were compared with the PGT instead of the GT. Even though the boxplots appear different, the medians are aligned and Table 5.11 shows a high correlation between the results with the GT and the PGT with a very low average distance between the results. This correlation indicates that the results with the PGT show lower dispersion but are correlated to the results against the GT (which show higher dispersion in the boxplot). The MAP comparison between the two figures also shows a high degree of similarity, with OLS and BioPortal slightly outperforming all other algorithms and Zooma obtaining the lowest results.

Table 5.11: Correlation and average distance between GT and PGT results for each metric tested (p-value < 0.01)

Metric	Pearson's R	Average Distance
NDCG	0.75	0.03
P@K	0.64	0.07
AP@K	0.69	0.05
MAP	0.93	0.05

From these results, we concluded that the PGT is a reliable complement for an expert-based ground truth in the setting described. Therefore, we extended the number of queries and ontology collection and tested the algorithms and search applications against the respective PGT. Figure 5.8 shows that the overall conclusions of the extended search are similar to the ones obtained with the smaller query set. The search applications achieve a superior performance against the IR algorithms, with OLS having the best performance, followed by BioPortal, Solr, and Zooma having the lowest results.

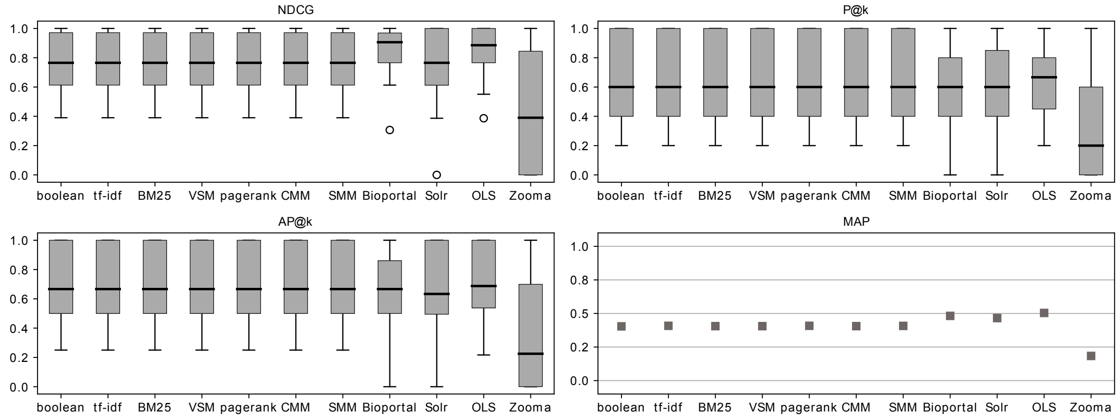


Figure 5.8: NDCG, P@k, AP@k and MAP results for the extended query and ontology collection, considering exact matches only, against the PGT

5.6 DISCUSSION

These are some of the key search factors that influenced the process of ranking resources in ontologies.

GROUND TRUTH: some judges ranked a class as not relevant, while other judges chose the same class as the most relevant. The principles of ontology engineering guide that ontologies should be orthogonal, which means that the same concept should not be independently created in each ontology, but, if it already exists, it should be reused from existing ontologies. In reality, however, ontologies are neither complete nor orthogonal and labels do not have a perfect representation of the semantics of a concept. The same concept can, therefore, be described in different ontologies, and experts do not agree which one should be the unifying one, as some experts will have different interpretations of the semantics of the label or the semantic categorisation of the ontology of a concept. This disagreement makes the task of building a gold standard for ontology search results difficult. The same issue served as the motivation for the design of the framework presented in Chapters 3 and 4. The framework presents a set of candidate data models instead of presenting a single best ranked candidate for each entity type or property. This allows for the presentation of a full ontological picture for the data publisher or ontology engineer to choose which candidate best fits their use-case and intended application, while also providing support to maximise interoperability with existing ontologies either by direct annotation or by relational links such as `owl:equivalentClass`.

PROBABILISTIC GROUND TRUTH: the main goal of building a probabilistic ground truth was to perform a deeper evaluation without involving more human experts. Finding experts both in the biomedical domain and in knowledge engineering to fill an extensive questionnaire is not a trivial task. The PGT was shown to have a significant agreement with the GT and the extended search showed that even with a broader domain of queries, the search applications still outperform the IR algorithms and OLS and BioPortal are the best performing systems.

PARTIAL MATCHES: the performance of all the IR algorithms suffered from too many partial matches. In biomedical ontologies, it is common to find multi-word labels since this domain describes complex concepts such as different phenotypes or anatomical parts.

For example, when the input for the algorithms is the query *Ovarian Cystadenoma*, the results consist mostly of partial matches of the word *ovarian*. The large number of partial matches led to a precision and NDCG of zero for most algorithms tested. The CMM algorithm is the only one in this set that distinguishes between partial and exact matches. However, it did not perform better than others since it does not evaluate each query individually. The relevance of partial matches to the performance of the IR algorithms was demonstrated with the limitation of the search to exact matches only. However, ideally, algorithms should not be limited to exact matches since they may exclude complex semantics (e.g., synonyms or descriptions) of ontologies from the search.

TIE-BREAKING: In the ground truth, ties were resolved by ordering them by ontology popularity, which was obtained from BioPortal. This method is also the main boost factor for search results in BioPortal. In our experiments, we do not believe this method introduced bias towards BioPortal since the number of ties was low (only two ties for all ten query words) and the remaining search applications also had a good overall performance, with OLS slightly outperforming BioPortal.

CONTROLLED VS. OPEN ACCESS ONTOLOGIES The search process included a set of 23 ontologies that was later expanded to 153 open-access ontologies. BioPortal contains several restricted access ontologies (e.g., SNOMED CT) that were featured in the search results but could not be included in the ground truth and, therefore, could not be evaluated. In some cases, this means that even though the ground truth contains some of the possible classes, BioPortal can have several more and there was no way, at this point, to evaluate their relevance in relation to the open access ontologies.

GENERAL VS. SPECIALISED TERMINOLOGIES: The search applications tested did not agree on the ranking for some queries. This issue was more noticeable when the query was more general, such as in the search for *Ovary*. This query has several exact matches in different ontologies, and all the applications ranked them differently. Some of those matches are species specific, but their descriptions are general. Table 5.12 shows the comparison of the results of the distinct applications tested, the ground truth and the probabilistic ground truth. This consideration also contributed to the larger dispersion in the precision of the expanded set of queries, since this set included less specific queries that can have several exact matches ranked in different orders.

Table 5.12: Comparing the ranking for *Ovary* between the GT, BioPortal (BP), OLS, Solr and PGT

Class URI	GT	BP	OLS	Solr	PGT
`http://purl.obolibrary.org/obo/NCIT_C12404`	1	1	1	3	1
`http://purl.obolibrary.org/obo/XAO_0000258`	2	4	3	2	4
`http://purl.obolibrary.org/obo/FMA_7209`	3	2	5	-	5
`http://purl.obolibrary.org/obo/ZFA_0000403`	4	3	4	1	3
`http://purl.obolibrary.org/obo/MA_0000384`	5	-	2	4	2

SOLR CONSIDERATIONS: Solr achieved a good performance and its biggest advantage is that any user can index and search through their own set of ontologies.

OLS CONSIDERATIONS: OLS uses Solr as its base for indexing and searching and boosts specific ontologies. The value of the boost is unknown and the process of attributing a boost to an ontology is not explicitly explained.

ZOOMA CONSIDERATIONS: Zooma's performance suffered not only from obtaining matches from annotated data but also from its focus on high precision. For all queries that found a match in the curated data, the search returned only one or two results. In most cases, these results did not match what the experts considered the most relevant ontology class. With only eight data sources to choose from, it is possible that the annotated data focused on domains not represented by the queries used. The queries that achieve top scores with Zooma are the queries that did not match any term in the curated data and, therefore, Zooma used OLS to find matches for the query.

5.7 RECOMMENDATIONS

In this chapter, we evaluated seven IR ranking algorithms and four search applications to assess their performance and find how they can support an internal approach for data modelling. This work established a ground truth through a user study with ten experts that ranked ten cancer-related queries and established a ground truth based on the consensus from the algorithms and systems tested. The ground truths were compared against the results from the ranking algorithms and the search applications. The evaluation experiment used 61 search queries (10 terms from the cancer genomics domain plus 51 general biomedical terms) and 153 biomedical ontologies (23 ontologies related to cancer genomics terms plus 130 general biomedical ontologies). Based on this analysis, we are able to conclude that (1) in their current state, the algorithms cannot handle partial matches, but forcing exact matches boosts their performance with possible loss of information and (2) the search applications are already robust in finding the relevant concepts for search queries in the correct order, with high precision and recall. The performance of search applications severely degrades with ambiguous search queries when compared to specific/concise queries. After evaluating the technologies, we conclude that, even though BioPortal and OLS outperform all other applications, one should not be chosen over the others by performance alone, but instead each situation (i.e., search scenario) should be analysed to choose which application to use:

SEARCHING FOR TOP-K. Both BioPortal and OLS have good precision in the top-3 hits, but both of them return a lot more results for general queries. It is possible to tune both tools to only return exact matches to reduce the number of matches, but the applications can still obtain more classes than the user is expecting. On the other hand, Zooma's smaller repository returns only one or two results, but that means that the queries have to be tuned towards the domains annotated by the curated data. This benchmark, in conjunction with existing benchmarks of ontology repositories (e.g., [192, 200, 201]), shows the effectiveness of ranking ontologies both with IR algorithms and using existing repository search engines. In general, ontology repositories greatly facilitate the discovery of relevant ontologies with a search engine expediting the process of finding relevant ontology resources. Therefore, external approaches to data modelling can be complemented by an ontology repository with a search function to complete a data model. Ontology repositories, however, are not available specifically for every domain, with the LOV providing search over wider domains and OLS and BioPortal focusing specifically on the biomedical domain.

SET OF ONTOLOGIES. If the set of ontologies used is a restriction for the search, BioPortal, OLS and Zooma can filter the ontologies shown in the results. Besides the ontologies available in OLS, Zooma also includes data sources used for the annotation process. OLS, however, does not index part of the ontologies indexed by BioPortal due to (1) BioPortal allowing user-submitted ontologies and OLS curating the ontologies allowed in the system and, (2) BioPortal having a large set of licensed ontologies (e.g., SNOMED CT) which are not indexed by OLS and, unless the user has access, cannot be indexed with Solr. In the biomedical domain, both of these conditions should be taken into account when choosing which service to use. If the user wants to use only open-access ontologies, it can filter them in BioPortal, but more easily can just search through all the indexed ontologies in OLS. However, if the user wants to search the largest possible set of ontologies, BioPortal would be the suggested choice. This recommendation can be extrapolated to every domain since a data publisher should carefully consider which ontologies are currently available in the domain of their dataset. If these ontologies are stored in a repository that can be searched, such as Bioportal, but are not freely available, the user cannot take advantage of a more personalised search unless necessary ontology licenses are acquired.

LOOKING FOR PARTIAL MATCHES. When the main goal of the search is not to find an exact match but to find related terms to the one being searched, OLS is the best solution. Contrary to BioPortal, which shows only the most relevant class in each ontology, OLS ranks and shows every possible match within the ontologies it has indexed. High scoring matches are shown first and then every possible partial match is also displayed. This consideration should also be taken into account in relation to the expertise of a data publisher. If a data publisher knows the exact concepts to model their data, exact match search will provide a speedier modelling process. However, if the data publisher is giving preference to a more exploratory search to better understand the concepts available to fit their data, then a search focusing on maximising recall is desirable.

CUSTOM SET OF ONTOLOGIES. In a use-case that a user has pre-selected a specific set of ontologies that wants to search and rank their concepts to match keywords from a data model, a standalone search engine can facilitate this scenario. These search engines provide greater liberty over the indexing and ranking processes, giving the data publisher more agency over the ontology resource recommendations that will best fit their use-case. In the benchmark studied, Solr was the search engine chosen and it performed comparably well with the best search engines, indicating that it is a good candidate for a custom ontology resource search.

SEARCH WITH CURATED ANNOTATIONS. Despite Zooma's lower performance in this context, it can be applied instead as an external approach to data modelling, where the dataset overlaps with terms from the data sources indexed by the tool. The best feature of Zooma is providing a backup search with OLS when no match is found in the data sources, providing a hybrid model of internal and external approaches to create a data model.

5.8 CONCLUSIONS

In this chapter, we focused on assessing the performance of ontology resource retrieval techniques and systems when matching a given keyword. This benchmark was motivated by the question:

RQ3 → *Can existing methods for ontology resource retrieval support the process of creating a data model using an internal approach?*

To answer this question, we divided it into the following subquestions:

RQ3.1 → *Are IR algorithms effective for retrieving and ranking top-k ontology classes that match a given keyword?*

We concluded that IR algorithms, without any modification or optimisation for the use-case, are not effective when ranking multi-word queries while accepting partial matches, but their performance substantially increases when forcing exact matches only.

RQ3.2 → *Are ontology resource search engines effective for retrieving and ranking top-k ontology classes that match a given keyword?*

The search engines tested performed well under the conditions tested both in the partial match and exact match settings. The different repositories and systems have different strengths and weaknesses that should be pondered when choosing a system to query.

RQ3.3 → *What requirements should be considered when choosing the best technique to retrieve ontology resources when using internal approaches to create a data model?*

We believe that the proposed benchmark is a good indicator of the performance of biomedical ontology search repositories tested, and we expect that, in the biomedical domain, this evaluation will aid researchers in finding the best application to fit their annotation needs will be easier. The characteristics of these biomedical repositories, however, can be extrapolated to other domains where similar considerations should be taken into account when choosing a system to find ontology resources. In general, we hope that the set of recommendations extracted from the results of the benchmark will allow data publishers and ontology engineers to find the best solution for their data modelling needs, be it reusing ontologies for an internal approach or complementing an external approach with ontology resource search.

Finally, returning to the overall question: *Can existing methods for ontology resource retrieval support the process of creating a data model using an internal approach?*

Tools and algorithms that facilitate ontology resource retrieval can support the process of creating a data model by providing effective means of retrieving relevant ontology resources by keyword search over indexed ontologies. We proposed a benchmark that compares how different tools and algorithms perform when retrieving the top-k ontology resources when searching with keywords. We compared the ontology resource retrieval search results from tools and algorithms with an expert-based ground truth and an automatically generated ground truth based on consensus between the tools and algorithms. We discussed the results in terms of performance and extracted a set of recommendations for choosing the best retrieval methodology for different use-cases. Similarly to Kolbe et al. [201], building a ground truth from query logs in the ontology repositories could further strengthen the analysis since user behaviours could be analysed in conjunction with the search results to provide recommendations that consider user preferences when choosing the top-k ontology resources.

Overall, we concluded that, specifically in the biomedical domain, the existing repositories provide a strong support for ontology resource retrieval, meaning that data publishers in the domain wishing to take an internal approach to data modelling have several tools at their disposal that can facilitate ontology reuse and extension by effectively finding the top-k best ontology resources that match a keyword. In this chapter, we focused on the biomedical domain. However, it can be advantageous to confirm that the same conclusions and recommendations hold true in other domains. The biomedical domain is privileged, since more than one repository and tool exists and several ontologies have been created to model a large portion of concepts in the domain. In other domains, where ontology repositories might not be available, general-purpose search engines, such as Solr and ElasticSearch, and IR algorithms provide the necessary customisation for a data publisher to load relevant ontologies and retrieve the top-k ontology resources that match keywords on their conceptual data model.

6 CONCLUSIONS

In this thesis, we proposed to answer the following overall research question:

How to facilitate the creation of a well-fitted and interoperable data model using a background knowledge graph built from existing RDF data sources?

To answer this question, we have designed a framework which is composed of several methods and algorithms to produce a ranked set of candidates to match the data of an input dataset. The work presented in this thesis was divided into three parts: (1) building the background knowledge graph from multiple data sources and the ontology graph extracted from those sources, (2) generating and ranking entity type and property candidates by exploiting information obtained from the knowledge graph and the ontology graph, and (3) examining the performance of ontology repositories and IR tools and algorithms when searching and ranking top-k ontology resources.

We experimented and evaluated the framework with diverse methods to examine its performance and analyse its impact in potential applications. Furthermore, we investigated how ontology repositories can further support the framework in the process of creating or integrating a schema with existing ontologies used in domain datasets.

Through the evaluations and experiments, we concluded that, within the set parameters, the framework achieves the goal of facilitating the process of creating a data model for an input dataset that can be adjusted to be accurate, interoperable, and consistent. The caveats of each methodology proposed were discussed and should be taken into consideration when implementing the framework and associated evaluations. Overall, one of the goals of the framework was to lower the entry level barrier to publish linked data. In our proposed framework, most of the underlying processes of identifying the best class or property to model the data are automatised and present the user with a recommendation of a data model for their data. Nonetheless, the user is presented with a ranked set of options and can customise to model to fit a certain use-case. This customisation can require some level of understanding of the data model and the underlying concepts of the RDF data model. Therefore, despite being able to streamline the process of creating a data model, the publishing barriers are not eliminated since some level of knowledge is still required for an optimal experience. On the other hand, the framework is more successful in aiding in selecting and linking the ontologies used to model data in a domain. The framework takes existing datasets and provides interoperable and consistent recommendations for data models and eliminate the need to manually analyse the data domain and empirically decide between existing data models.

6.1 LESSONS LEARNED

In terms of building a knowledge graph to extract data models (Chapter 3), we learned that the first challenge is selecting the appropriate sources to load and create the knowledge graph. Even though we did not present this particular use-case in this thesis, it would be possible to load general purpose knowledge graphs such as DBpedia, Wikidata, or YAGO to match general domain input datasets covered in these knowledge graphs. However, the issue of multiple data models to model the same domain is more prevalent in domain datasets that need to describe specific concepts, as in the use-cases chosen in this thesis. In general, domain knowledge, such as in the library domain, finding data sources and datasets was relatively easy since several libraries are producing and publishing RDF data (with different data models, thus creating the issue of this thesis). However, finding appropriate source data for an expert level domain, such as the life sciences, was more challenging since it required deeper investigation of the topics to find relevant resources. Therefore, as discussed in Chapter 3, the research field of dataset matching might prove to be essential for knowledge graph modelling, while the opposite might also be verified, i.e., having consistent data models for a domain might also prove to be fundamental for dataset matching.

In Chapter 3, in relation to the underlying ontology graph extracted from the knowledge graph, we also learned the importance of enrichment to improve the connectivity of the graph. Ontologies are commonly developed independently. Even though studies have shown a reduction in the use of proprietary vocabularies and an improvement to reuse [46, 48], in a given domain, multiple ontologies are likely to have conceptual overlaps. Therefore, we proposed and tested three methodologies to enhance the connections between the ontologies in the data models. We learned that even a small number of edges can have a significant impact on how tightly a graph is connected, making it hard to separate concepts within the graph, therefore, facilitating the use of graph traversal-based methodologies.

Finally, in this chapter, we also explored the use of classification models to predict datatype properties. We learned that the most important step in this process is knowing the data. This knowledge allows for the correct design of the features to extract from the values of the properties. In the case of the library datasets, the literal data they contain is not very varied, mostly consisting of textual strings, dates, or unique codes. However, once again the biomedical domain proved to be more challenging. For example, identifier codes are prevalent. However, while these can be similar in their representation, they conceptually identify different concepts (e.g., gene and protein symbols). Therefore, to successfully use a classification model, the characteristics of the literals in the dataset have to be well-defined beforehand.

After building the background knowledge graph, we focused on recommending a ranked list of suitable data models to fit an input dataset (Chapter 4). First, we learned about the challenges that each file format poses to the generation of entity type and property candidates. RDF is the most direct match between input and knowledge graph. However, since it was developed to be expressive, these datasets are usually the most complex to re-model or integrate with other data models. If a dataset is published with a structure, a data publisher has already put some thought into the representation of their data. Therefore, adapting this established data model to other existing models can be challenging due to the number of entity types and properties. In our case, we used RDF input datasets that were automatically generated (University and GenAge) or manually curated (Gutenberg). The automatically generated data models feature a set of entity types and properties developed for broad applications and cover several use-cases, such as the `bibframe:BaseMaterial` or `bibframe:ColorContent` entity types. These entity types will be useful to model specific data. However, when looking only at a library catalogue, they become less relevant. Therefore, the challenge for a data publisher here would be to analyse the results of the framework and decide which elements of the data model are worth keeping for their specific use-case and which are expendable for the sake of interoperability with the domain.

When looking at CSV and JSON, we learned that the pre-processing phase is the most relevant since the input data has to align with the requirements of the framework. In the case of these file formats, if the user provides no extra information, the framework is more likely to underperform or be more inefficient since it will start an exhaustive search instead of a targeted one. Therefore, the lesson learned once again is that it facilitates this matching problem to have some general knowledge of the input data and what is being looked for in a data model.

We also learned in Chapter 4 how the characteristics of different elements that compose an entity-relation RDF data model affect the generation and ranking process. For entity type generation and ranking, we learned that matching entities in the input data with the knowledge graph is a good starting point for entity type candidate generation. The methods based on AOR to match entities obtained a good performance that, combined with the content scoring methodology, managed to order the extensive list of candidates into a reasonable top-10 candidates to match the input entity type. Considering the motivation for this framework, we also learned that providing multi-language support systems could expand the application of this framework since it likely that data publishers will use different languages to design their own data model to fit their specific language. Nonetheless, conceptually, these data models should align and the framework might have the potential to provide this alignment of the data models.

We learned that ranking properties is more challenging than ranking entity types since entity types are supported by the entities they represent, which provides useful information in their ranking. Looking at the state-of-the-art in both domains, we found a larger proportion of work in matching entity types than properties. Therefore, we propose methods to generate and rank datatype and object properties that not only integrate well with the proposed framework but also produce promising results. Since the main goal of the framework is not optimal precision, the level of error obtained is acceptable. Nonetheless, further investigation into matching datatype and object properties in the context of a data model is necessary to improve the ranking of the recommendations provided to facilitate the recommendation process further.

In terms of interoperability with existing data models, we learned that important considerations include the weight to give to this parameter. We encountered several cases where the framework produced results that were well ranked in terms of content score, but the interoperability re-ordered the results to favour entity types or properties lower ranked but that are very interoperable within the background knowledge graph. Therefore, a data publisher should consider and test different weights for interoperability to achieve a balance suitable for their use case. The demonstrator developed provides an overview of what this testing phase could look like and shows how easily the framework could be changed to fit the desired parameters.

Finally, we learned that to bring the whole data model together, we had to guarantee not only correctness and interoperability but some measure of consistency to provide meaningful recommendations. For this, we used a consistency score that looks at triple patterns in the knowledge graph and the data model being proposed and boosts the triples that are more consistent with both. The use of this score achieved its goal and the aggregated data model suggestions were more consistent both with the knowledge graph and with itself. In the future, axioms from ontologies that denote logical relationships, such as disjointness or union, can be exploited to contribute the data model recommendations. This addition would create a score that not only provides a consistent data model in terms of the suggestions from the knowledge graph but also in terms of guaranteeing that the recommendations are logically correct when considering the logical relationships contained in the ontology graph.

In Chapter 5, we looked at the problem addressed in this thesis from a different perspective that fits the case where the requirements to use the proposed framework are not met and, therefore, a data publisher has to look at alternatives, which most likely will include an exhaustive search of ontology repositories. In this chapter, we learned that no single ontology repository or AOR search strategy can claim superiority over the others, which led us to propose a set of recommendations to decide which solution is more appropriate to each use case. These recommendations were created by analysing the results of the benchmark that compared different solutions to search for the top-k ontology resource search results to match input keywords. This analysis, together with the observation of the workflow of each solution, led us to recommend cases where one strategy might be preferable over others.

Finally, to the best of our knowledge, no other proposed work provides methodologies to recommend data models, considering the characteristics of the data and the interoperability with the domain but also focused on consistency. The proposed methods are not intended to provide a precise answer but, instead provide a holistic view of the recommended solutions. Therefore, we learned that the methods described provide a good starting point for further developments that can potentially be put in practice in real-world cases.

6.2 FUTURE WORK

Knowledge graphs play an important role in the organisation and discovery of information, and we believe that this role will become more relevant in the near future. Google was instrumental in the dissemination of the concept, now being adopted by small and large-scale enterprise initiatives. Improving and developing technologies and frameworks to support building and maintaining knowledge graphs facilitates the understanding of their benefits and users will be more keen to adopt the technology. In this thesis, we aim to provide methods to advance the state-of-the-art in knowledge graph integration and completion but also in the transformation of heterogeneous data sources into a knowledge graph format. However, the framework presented in this thesis still has limitations, which were individually presented in their respective chapters.

In addition, conceptually there is also future work that can extend and improve the framework and ideas proposed in this thesis. First, our framework builds upon the assumption that data publishers need frameworks that (1) facilitate the process of publishing data in RDF and (2) enable interoperability with already published datasets. These assumptions were drawn both intuitively and by considering the principles of ontology engineering, which strongly encourage reuse while building new ontologies (see [65, 185]). Nonetheless, in the future, a user study could be performed among data publishers and ontology engineers from different domains of knowledge to assess what they are looking for in a data model. This survey could include not only their previous experiences but also their current data needs. For example, a data publisher in the biomedical domain might be more concerned about the specificity of the annotations than a data publisher in the library data domain. In the library data domain, the conceptual complexity of the entities and properties is lower than in the biomedical domain where complex relationships exist between entities. From this survey, we could then extract this kind of information and use it to further improve our framework and make it more adaptable to the needs of different domain experts. Furthermore, this survey could also provide a subjective evaluation by the experts on the results of our framework and verify if balancing the parameters (i.e., weights of content and interoperability scores) matches their data modelling preferences. However, as previously mentioned, the task of finding the best data model is subjective to the data publisher and the data domain. Therefore, no definitive conclusions would be drawn without a large-scale approach since these would need to be drawn from consensus among the domain experts.

Overall, the most important next step would be to further optimise the process in terms of effectiveness and efficiency. In terms of effectiveness, further investigation is necessary to assess datatype and object property generation and ranking since the presented methods are still not performing as well as the methods for entity type generation and ranking. Examining ways to improve the classification model for datatype properties or exploring different machine learning approaches could lead to an improvement in the results. In terms of object property generation, the generation process is strongly dependent on the entity type candidate generation and ranking. Therefore, finding alternative methods to be combined with the ones presented in this thesis would be desirable so that the object property generation stage could be more independent from the rest of the framework.

In terms of ranking datatype properties, the best measure of ranking is the distance in the ontology graph, for which, if no ontology mappings or extended mappings were found, even a correct candidate will not be ranked favourably. Therefore, investigating content scoring methods geared towards properties would be beneficial to improve the performance of the framework when ranking these data model elements.

When a robust generation and ranking framework is achieved, the next step would involve optimising the process to facilitate the use by non-expert users. Ideally, a user interface would be developed that guided the user from the process of building a knowledge graph (with the possibility of integrating dataset matching research to facilitate this task) to an interface similar to the demonstrator presented in Section 4.6. The knowledge graphs produced by users, when possible, could be publicly shared, to allow other users in the same domain to extract candidate data models for their data too.

Ultimately, the use of such a framework could lead to an improvement of reuse of domain ontologies and allow the convergence of data models in the same domain. Both situations are desirable since lowering the entry level complexity to data modelling with ontologies brings more users and better adoption of the technologies being proposed by the Semantic Web and linked data paradigms.

Appendices

A ONTOLOGIES

In this appendix we can find the ontologies loaded into the use-case's ontologies graphs. The process of their extraction and processing is described in Chapter 3.

A.1 LIBRARY USE-CASE

For the library use-case we downloaded and parsed a total of 36 ontologies. Ontologies were download in August 2020.

Table A.1: Descriptive statistics of the ontologies in the library use-case.

Prefix	Namespace	# Classes	# Properties
agrelon	`https://d-nb.info/standards/elementset/agrelon`	48	83
bflc	`http://id.loc.gov/ontologies/bflc`	24	38
bibo	`http://purl.org/ontology/bibo`	69	117
bio	`http://purl.org/vocab/bio/0.1`	43	33
bne	`http://datos.bne.es/def`	6	145
bnf-onto	`https://data.bnf.fr/ontology/bnf-onto`	1	41
dcterms	`http://purl.org/dc/terms`	22	55
dnb	`https://d-nb.info/standards/elementset/dnb`	4	10
edm	`http://www.europeana.eu/schemas/edm`	14	35
egr2	`http://rdvocab.info/ElementsGr2`	0	59
event	`http://purl.org/NET/c4dm/event.owl`	6	23
foaf	`http://xmlns.com/foaf/0.1`	15	68
frbr	`http://rdvocab.info/uri/schema/FRBRentitiesRDA`	14	0
geo	`http://www.geonames.org/ontology`	2	5
gnd	`https://d-nb.info/standards/elementset/gnd`	68	234
gsp	`http://www.opengis.net/ont/geosparql`	3	34
intervals	`http://reference.data.gov.uk/def/intervals`	38	27
isbd	`http://iflastandards.info/ns/isbd/elements`	27	163
mads	`http://www.loc.gov/mads/rdf/v1`	58	92
marcrole	`http://id.loc.gov/vocabulary/relators`	0	0
mo	`http://purl.org/ontology/mo`	60	167
ore	`http://www.openarchives.org/ore/terms`	4	8
org	`http://www.w3.org/ns/org`	9	35
owl	`http://www.w3.org/2002/07/owl`	26	49
powder-s	`http://www.w3.org/2007/05/powder-s`	2	19
process	`http://www.daml.org/services/owl-s/0.9/Process.owl`	43	53
prov	`http://www.w3.org/ns/prov`	51	89
rdarw	`http://rdvocab.info/RDARelationshipsWEMI`	0	508
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns`	7	9
rdfs	`http://www.w3.org/2000/01/rdf-schema`	6	9
rdvocabe	`http://rdvocab.info/Elements`	10	456
schema	`http://schema.org`	618	878
skos	`http://www.w3.org/2004/02/skos/core`	4	28
umbel	`http://umbel.org/umbel`	36	50
vivo	`http://vivoweb.org/ontology/core`	385	420
wgs84_pos	`http://www.w3.org/2003/01/geo/wgs84_pos`	2	5

A.2 LIFE SCIENCES USE-CASE

For the life sciences use-case we downloaded and parsed a total of 33 ontologies. Ontologies were download in August 2020.

Table A.2: Descriptive statistics of the ontologies in the life sciences use-case.

Prefix	IRI	# Classes	# Properties
bfo	`http://purl.obolibrary.org/obo/bfo.owl`	35	22
bto	`http://purl.obolibrary.org/obo/bto.owl`	6428	36
chebi	`http://purl.obolibrary.org/obo/chebi.owl`	144 476	47
cl	`http://purl.obolibrary.org/obo/cl.owl`	10 616	477
clo	`http://purl.obolibrary.org/obo/clo.owl`	44 870	303
dcat	`http://www.w3.org/ns/dcat`	8	30
dcterms	`http://purl.org/dc/terms`	22	55
eco	`http://purl.obolibrary.org/obo/eco.owl`	2937	93
edam	`http://edamontology.org`	3421	77
faldo	`http://biohackathon.org/resource/faldo/`	16	10
foaf	`http://xmlns.com/foaf/0.1`	15	68
geno	`http://purl.obolibrary.org/obo/geno.owlo`	410	198
go	`http://purl.obolibrary.org/obo/go.owl`	50 331	59
hp	`http://purl.obolibrary.org/obo/hp.owl`	27 230	523
iao	`http://purl.obolibrary.org/obo/iao.owl`	249	118
ncbitaxon	`http://purl.obolibrary.org/obo/ncbitaxon.owl`	2 241 110	27
obi	`http://purl.obolibrary.org/obo/obi.owl`	2820	167
owl	`http://www.w3.org/2002/07/owl`	26	49
pato	`http://purl.obolibrary.org/obo/pato.owl`	2809	203
pav	`http://purl.org/pav`	0	42
po	`http://purl.obolibrary.org/obo/po.owl`	1993	61
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns`	7	9
rdfs	`http://www.w3.org/2000/01/rdf-schema`	6	9
ro	`http://purl.obolibrary.org/obo/ro.owl`	5	553
sepio	`http://purl.obolibrary.org/obo/sepio.owl`	131	137
skos	`http://www.w3.org/2004/02/skos/core`	4	28
so	`http://purl.obolibrary.org/obo/so.owl`	2661	91
stato	`http://purl.obolibrary.org/obo/stato.owl`	814	108
to	`http://purl.obolibrary.org/obo/to.owl`	1572	46
uberon	`http://purl.obolibrary.org/obo/uberon.owl`	15 244	352
uo	`http://purl.obolibrary.org/obo/uo.owl`	591	2
vivo	`http://vivoweb.org/ontology/core`	402	434
wbls	`http://purl.obolibrary.org/obo/wbls.owl`	792	153

B MATHEMATICAL NOTATION

B.1 DATA STRUCTURES

The following data structures are primarily used in the pseudocode through the thesis.

$S \leftarrow \{a, b, c, \dots\}$ a *set*, i.e., is an unordered and mutable collection of unique elements. A say may be be empty, which is denoted by $\emptyset$.

$L \leftarrow [l_1, l_2, \dots, l_n]$ a *list*, i.e., an ordered collection of elements. This collection is mutable and, therefore, elements can be removed ($L.\mathit{remove}(e)$) and added to its end ($L.\mathit{append}(e)$). A list may be empty, which is denoted by $[\,]$.

$T \leftarrow \langle t_1, t_2, \dots, t_n \rangle$ a *tuple*, which is an immutable ordered collection of elements. Individual elements of the tuple may be referenced by their original variable name, e.g., $T.t_1$ refers to the element t_1 in the tuple.

$M \leftarrow \{k_1 : v_1, \dots, k_n : v_n\}$ a *mapping*, i.e., a mutable associative array containing a collection of key-value pairs, where each value v is accessed by specifying its corresponding key k in square bracket, e.g., $M[k_1]$ refers to v_1. The key has to be defined by an immutable structure, while the value can be any data structure. A mapping can be empty, which is denoted by $\{\,\}$.

B.2 SYMBOLS, OPERATORS, AND FUNCTIONS

The following data structures are primarily used in the pseudocode through the thesis.

$|D|$ cardinality or number of elements inside list, set, or tuple D.

$\wedge$ logical `AND` used to combine statements.

$d \mid d \in D \wedge f(d)$ mutable structure-builder notation, i.e., initialises a list, set, or tuple by including the elements that satisfy the conditions stated.

$\forall a \in A$ *All* logical quantifier, which translates to for all a in A.

$\widetilde{a}$ represents a numerical value that has been normalising by the maximum of its data structure, e.g., $\widetilde{a} \leftarrow \frac{a}{\max\limits_{a \in A} f(s)}$ is the value a divided by the maximum of the values in set A to which a belongs.

$\times$ indicates a Cartesian product operation, i.e., the multiplication of n sets that forms a set of all ordered n-length tuples. For example, $A \times B = \{\langle a, b \rangle \mid a \in A \wedge b \in B\}$

General Functions

$\mathit{pow}(n, x)$ returns the power of a base number n with the exponent x, e.g., $\mathit{pow}(2, 3) = 2^3 = 8$.

$\mathit{mean}(a, b, \dots)$ returns the mean of the numerical elements provided, e.g., $\mathit{mean}(1, 2, 3) = \frac{1+2+3}{3} = 2$

Specific Functions

$\mathit{unique\_types}(D)$ extracts the unique types or type equivalent elements T for entities in D.

$\mathit{random\_sample}(D, x, n)$ returns a random sample of n documents of type x or when x is a property, documents where x has a value.

$\mathit{literal}(s)$ checks if string s is literal (True) or an entity URI that exists in the Knowledge Graph (False).

$\mathit{search}(S, l, w)$ searches the document store S for label l and returns a maximum of w search results.

$\mathrm{top}_k(M, sort\_key)$	takes as input a mapping with a list as value or a list. The function selects the k elements the highest value of $sort\_key$.

C ALGORITHM WORKFLOWS

In this appendix, we can find examples of workflows for the algorithms described in Chapter 4. These are examples of running the algorithm in a small toy dataset to illustrate the process of generating and ranking entity type and property candidates.

C.1 ENTITY TYPE GENERATION

Figure C.1 show the process described in Algorithm 4.1 with a hypothetical example for an input type $t =$ `/type/work`. A random sampling of entities E_t of type $t \in T$ from the input dataset D returns two entities e_1 and e_2. We extract the literals in each one of these entities obtaining set L_{e_1,t_1} and L_{e_2,t_1}, respectively. We search each one of these entities in the document store S and obtain two search results for e_1 and one search result for e_2. We extract the entity types and labels of each match and return the tuple of each entity type with the label set of the entity.

C.2 ENTITY TYPE CONTENT SCORING

Figure C.2 shows the process described in Algorithm 4.4 with the results of Figure C.1. The workflow shows the computation of each metric and their combination into a single content score per candidate per type in the input dataset.

C.3 DATATYPE PROPERTY GENERATION

Figure C.3 shows the process described in Algorithm 4.2 with a hypothetical example for the input datatype property `title`. We obtain a random sample of two entities with this property e_1 and e_2 and extract the value of the property `title` in each entity. For each value, we use the Random Forest models to obtain predictions of datatype property candidates that match the value. In this example, for the input property `title`, the models would predict the datatype properties in the Knowledge Graph `uk:has_title`

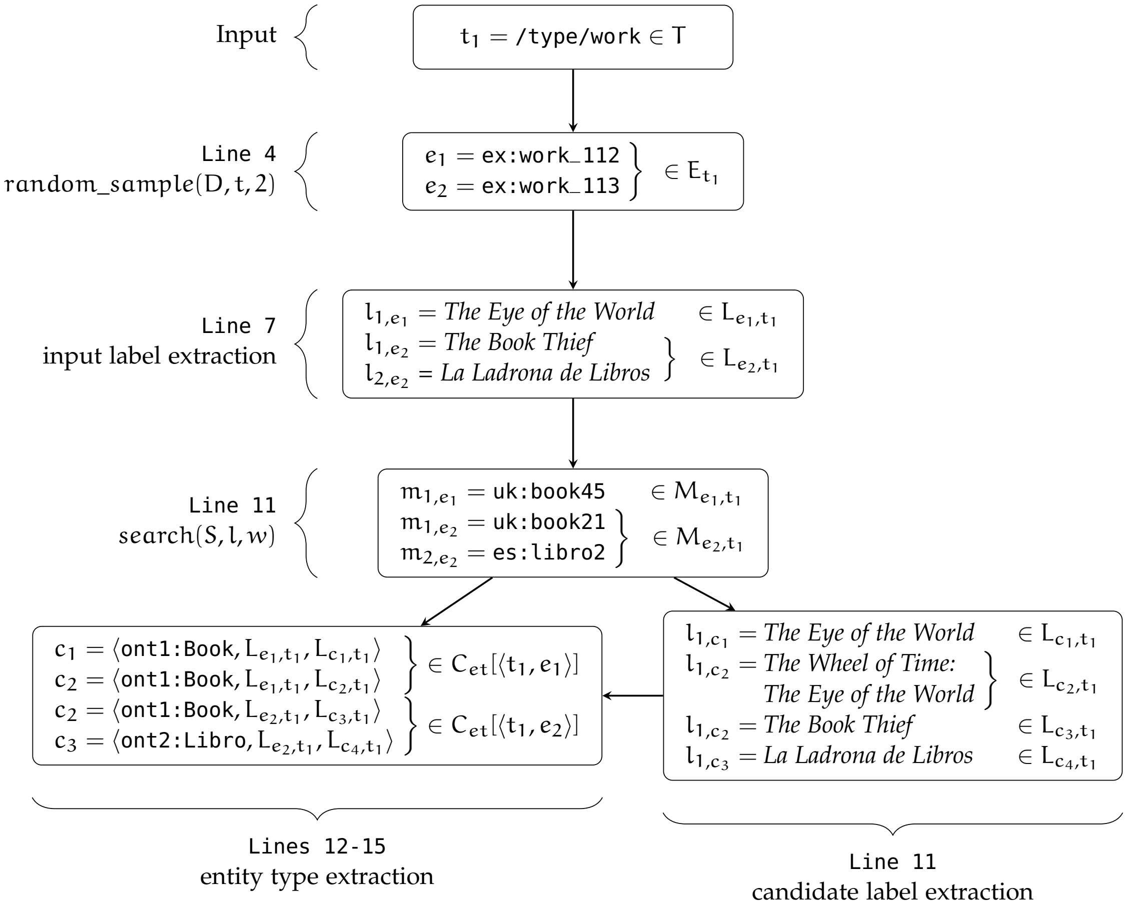

Input
$t_1 = $ /type/work $\in T$
Line 4
random_sample$(D, t, 2)$
$e_1 = $ ex:work_112
$e_2 = $ ex:work_113
$\in E_{t_1}$
Line 7
input label extraction
$l_{1,e_1} = $ *The Eye of the World* $\in L_{e_1,t_1}$
$l_{1,e_2} = $ *The Book Thief*
$l_{2,e_2} = $ *La Ladrona de Libros*
$\in L_{e_2,t_1}$
Line 11
search(S, l, w)
$m_{1,e_1} = $ uk:book45 $\in M_{e_1,t_1}$
$m_{1,e_2} = $ uk:book21
$m_{2,e_2} = $ es:libro2
$\in M_{e_2,t_1}$
$c_1 = \langle$ont1:Book$, L_{e_1,t_1}, L_{c_1,t_1}\rangle$
$c_2 = \langle$ont1:Book$, L_{e_1,t_1}, L_{c_2,t_1}\rangle$
$\in C_{et}[\langle t_1, e_1\rangle]$
$c_2 = \langle$ont1:Book$, L_{e_2,t_1}, L_{c_3,t_1}\rangle$
$c_3 = \langle$ont2:Libro$, L_{e_2,t_1}, L_{c_4,t_1}\rangle$
$\in C_{et}[\langle t_1, e_2\rangle]$
$l_{1,c_1} = $ *The Eye of the World* $\in L_{c_1,t_1}$
$l_{1,c_2} = $ *The Wheel of Time: The Eye of the World* $\in L_{c_2,t_1}$
$l_{1,c_2} = $ *The Book Thief* $\in L_{c_3,t_1}$
$l_{1,c_3} = $ *La Ladrona de Libros* $\in L_{c_4,t_1}$
Lines 12-15
entity type extraction
Line 11
candidate label extraction

Figure C.1: Overview of entity type candidate generation process with an example.

and es:título. The first one is predicted more than once which will be taken into consideration when re-ranking these candidates later.

C.4 DATATYPE PROPERTY CONTENT SCORING

Figure C.4 shows the process described in Algorithm 4.5 with a hypothetical example extracted from the results of Figure C.3. The workflow shows the computation of each metric and their combination into a single content score per candidate per type in the input dataset.

C.5 OBJECT PROPERTY GENERATION

Figure C.5 shows the process described in Algorithm 4.3 with a hypothetical example for an object property has_author, with the subject ex:work_112 and object ex:author_05. This property has a domain /type/work and a range /type/author which are used to

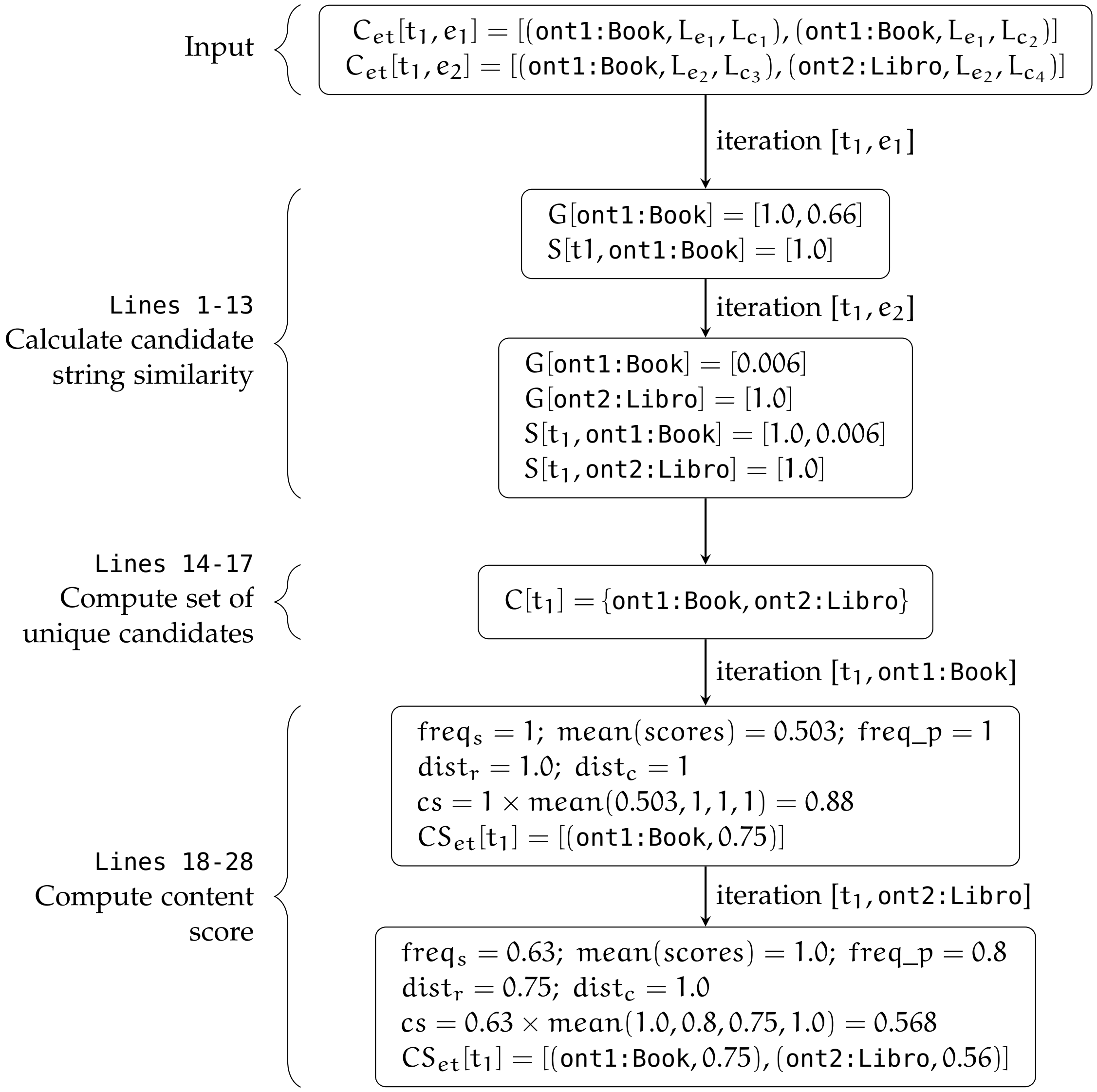

Input
$C_{et}[t_1, e_1] = [(\texttt{ont1:Book}, L_{e_1}, L_{c_1}), (\texttt{ont1:Book}, L_{e_1}, L_{c_2})]$
$C_{et}[t_1, e_2] = [(\texttt{ont1:Book}, L_{e_2}, L_{c_3}), (\texttt{ont2:Libro}, L_{e_2}, L_{c_4})]$
iteration $[t_1, e_1]$
$G[\texttt{ont1:Book}] = [1.0, 0.66]$
$S[\texttt{t1}, \texttt{ont1:Book}] = [1.0]$
Lines 1-13
Calculate candidate string similarity
iteration $[t_1, e_2]$
$G[\texttt{ont1:Book}] = [0.006]$
$G[\texttt{ont2:Libro}] = [1.0]$
$S[t_1, \texttt{ont1:Book}] = [1.0, 0.006]$
$S[t_1, \texttt{ont2:Libro}] = [1.0]$
Lines 14-17
Compute set of unique candidates
$C[t_1] = \{\texttt{ont1:Book}, \texttt{ont2:Libro}\}$
iteration $[t_1, \texttt{ont1:Book}]$
$freq_s = 1;\ mean(scores) = 0.503;\ freq\_p = 1$
$dist_r = 1.0;\ dist_c = 1$
$cs = 1 \times mean(0.503, 1, 1, 1) = 0.88$
$CS_{et}[t_1] = [(\texttt{ont1:Book}, 0.75)]$
Lines 18-28
Compute content score
iteration $[t_1, \texttt{ont2:Libro}]$
$freq_s = 0.63;\ mean(scores) = 1.0;\ freq\_p = 0.8$
$dist_r = 0.75;\ dist_c = 1.0$
$cs = 0.63 \times mean(1.0, 0.8, 0.75, 1.0) = 0.568$
$CS_{et}[t_1] = [(\texttt{ont1:Book}, 0.75), (\texttt{ont2:Libro}, 0.56)]$

Figure C.2: Overview of entity type candidate content scoring with an example.

retrieve the entity type candidates previously ranked with Algorithm 4.4. Then the Cartesian product of multiples domain and range candidates sets is computed, and C_{op} includes all the edges between domain and range candidates. For reasons of space and readability we do not show in the figure all possible product combinations between domain/range and property but show the results of each step individually and present the results of the combination.

C.6 OBJECT PROPERTY CONTENT SCORING

Figure C.6 shows the process described in Algorithm 4.6 with a hypothetical example extracted from the results of Figure C.5. The workflow shows the computation of each

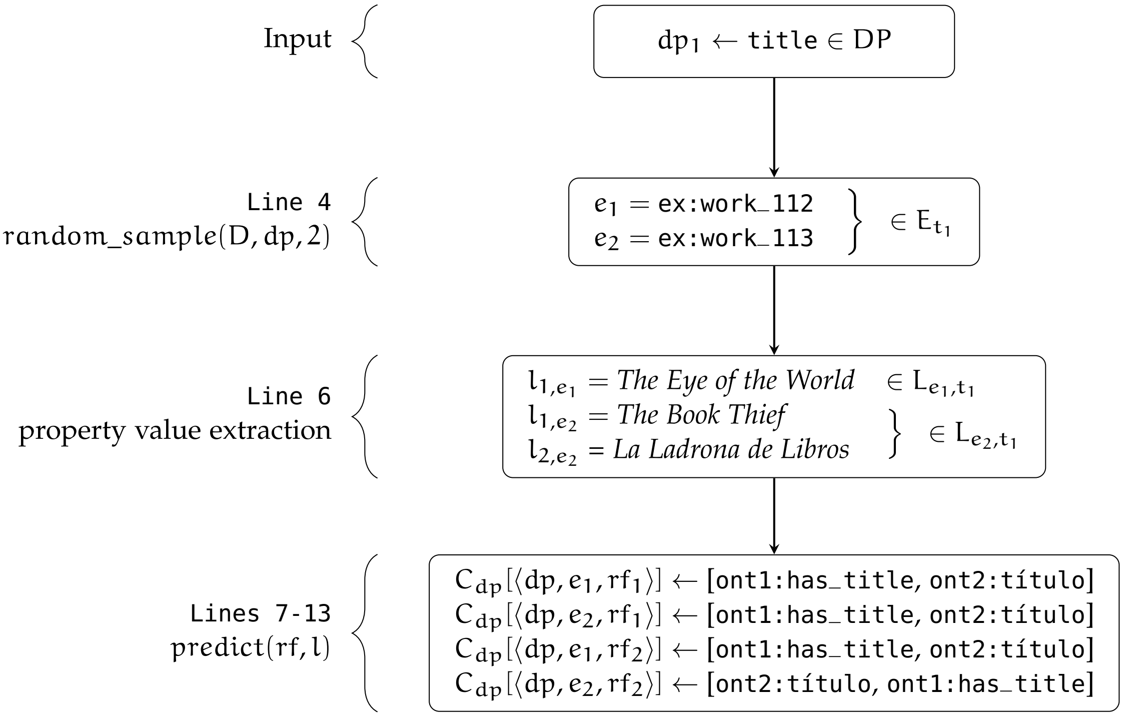

Input
$dp_1 \leftarrow$ title $\in DP$
Line 4
random_sample(D, dp, 2)
$e_1 =$ ex:work_112
$e_2 =$ ex:work_113
$\in E_{t_1}$
Line 6
property value extraction
$l_{1,e_1} =$ The Eye of the World $\in L_{e_1,t_1}$
$l_{1,e_2} =$ The Book Thief
$l_{2,e_2} =$ La Ladrona de Libros
$\in L_{e_2,t_1}$
Lines 7-13
predict(rf, l)
$C_{dp}[\langle dp, e_1, rf_1\rangle] \leftarrow$ [ont1:has_title, ont2:título]
$C_{dp}[\langle dp, e_2, rf_1\rangle] \leftarrow$ [ont1:has_title, ont2:título]
$C_{dp}[\langle dp, e_1, rf_2\rangle] \leftarrow$ [ont1:has_title, ont2:título]
$C_{dp}[\langle dp, e_2, rf_2\rangle] \leftarrow$ [ont2:título, ont1:has_title]

Figure C.3: Overview of datatype property candidate generation process.

metric and their combination into a single content score per candidate per type in the input dataset.

C.7 INTEROPERABILITY SCORING

Figure C.7 shows the process described in Algorithm 4.7 when computing the interoperability score for the hypothetical example presented for the generation and ranking of entity types. The interoperability process computes the two metrics, Neighbourhood Size and Interoperability, and combines them into a single interoperability score is for each candidate per source type.

C.8 CONSISTENCY SCORING - AGGREGATION

Figure C.8 shows the process described in Algorithm 4.8 when computing the aggregation phase of the consistency score for the hypothetical examples previously described. For brevity, due to the size of the results of the Cartesian product between candidates, we display only what would be the top ranked candidate ⟨domain – property ⇾ range⟩ triple patterns for each input triple pattern. In this example we use the following: $w_{cs} = 1.0; w_{is} = 1.0, w_{cns} = 0.3$. Again, due to the size of the tuple we use the

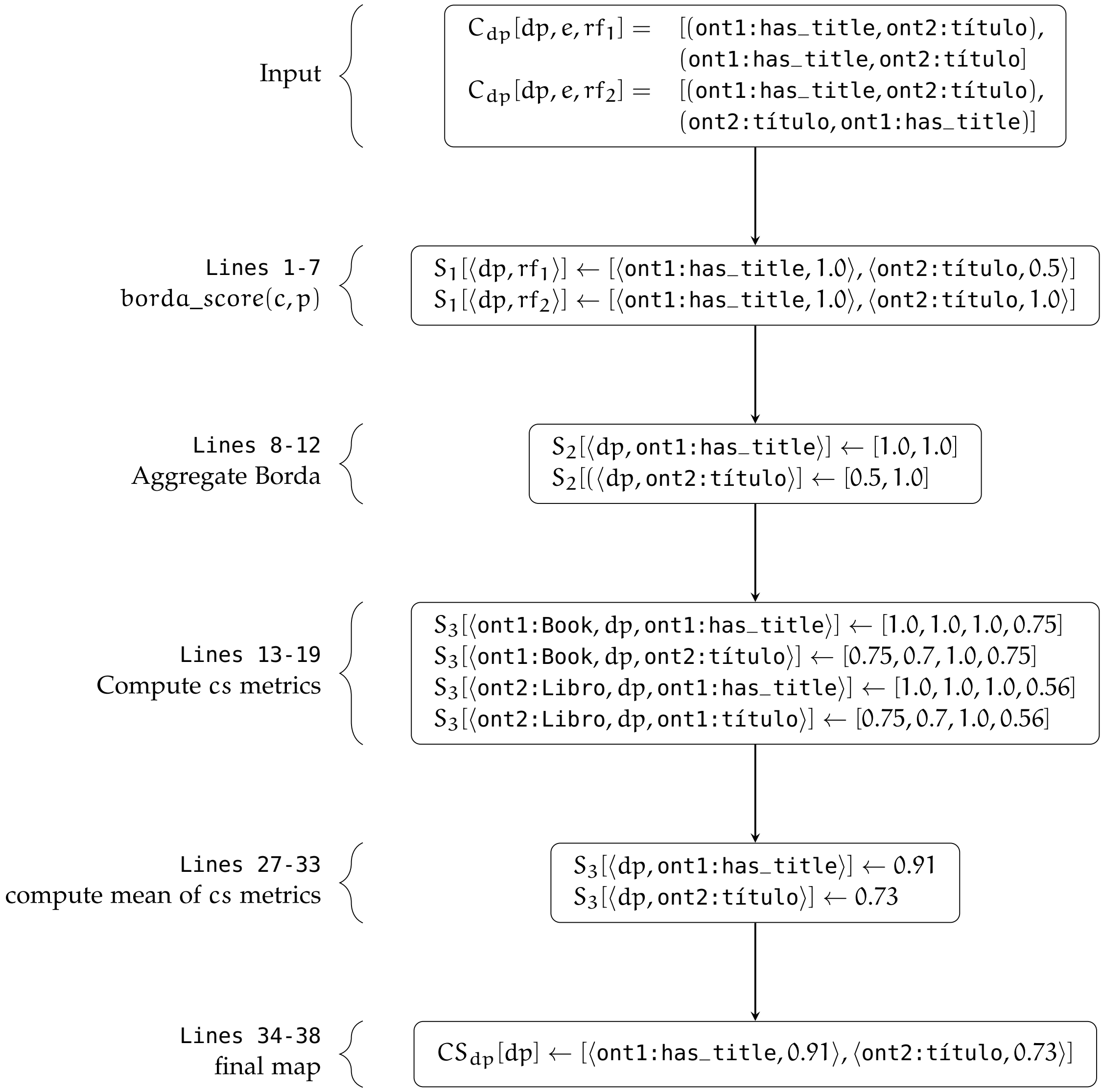

Input
$C_{dp}[dp, e, rf_1] =$ [(ont1:has_title, ont2:título), (ont1:has_title, ont2:título]
$C_{dp}[dp, e, rf_2] =$ [(ont1:has_title, ont2:título), (ont2:título, ont1:has_title)]
Lines 1-7
borda_score(c, p)
$S_1[\langle dp, rf_1\rangle] \leftarrow$ [⟨ont1:has_title, 1.0⟩, ⟨ont2:título, 0.5⟩]
$S_1[\langle dp, rf_2\rangle] \leftarrow$ [⟨ont1:has_title, 1.0⟩, ⟨ont2:título, 1.0⟩]
Lines 8-12
Aggregate Borda
S_2[⟨dp, ont1:has_title⟩] ← [1.0, 1.0]
S_2[(⟨dp, ont2:título⟩] ← [0.5, 1.0]
Lines 13-19
Compute cs metrics
S_3[⟨ont1:Book, dp, ont1:has_title⟩] ← [1.0, 1.0, 1.0, 0.75]
S_3[⟨ont1:Book, dp, ont2:título⟩] ← [0.75, 0.7, 1.0, 0.75]
S_3[⟨ont2:Libro, dp, ont1:has_title⟩] ← [1.0, 1.0, 1.0, 0.56]
S_3[⟨ont2:Libro, dp, ont1:título⟩] ← [0.75, 0.7, 1.0, 0.56]
Lines 27-33
compute mean of cs metrics
S_3[⟨dp, ont1:has_title⟩] ← 0.91
S_3[⟨dp, ont2:título⟩] ← 0.73
Lines 34-38
final map
$CS_{dp}[dp] \leftarrow$ [⟨ont1:has_title, 0.91⟩, ⟨ont2:título, 0.73⟩]

Figure C.4: Overview of datatype candidate content scoring with an example.

variable *scores* to represent the tuple with candidates and scores that results from the aggregation phase of the consistency score.

C.9 CONSISTENCY SCORING – REFINEMENT

Figure C.9 shows the process described in Algorithm 4.9 when computing the refinement phase of the consistency score for the hypothetical examples previously described. In this figure, we represent an example of a domain and range entity type candidate that would be adjusted with this refinement step. Considering that `ont1:Book` and `ont3:Person` are more commonly ranked first, the scores of triples with their input domain d and range r are refined to boost these candidates in other triples with the same

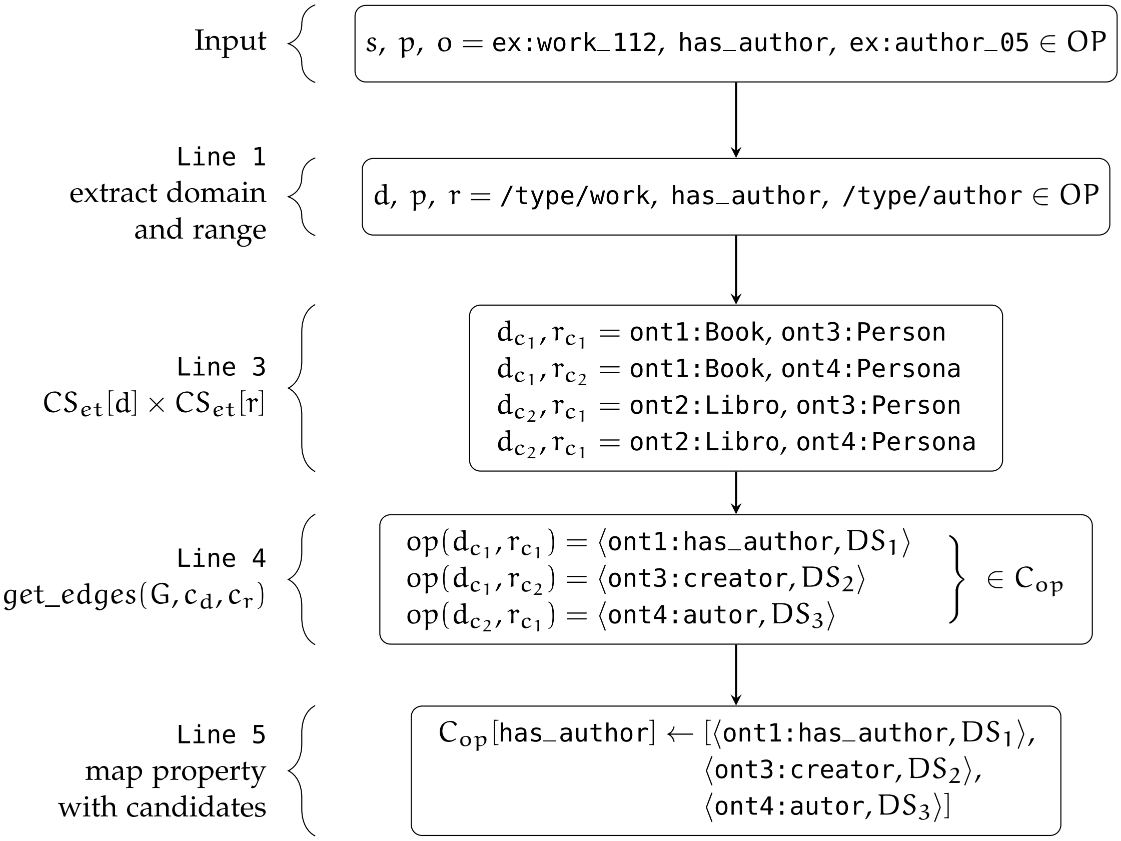

Input
s, p, o = ex:work_112, has_author, ex:author_05 ∈ OP
Line 1
extract domain and range
d, p, r = /type/work, has_author, /type/author ∈ OP
Line 3
$CS_{et}[d] \times CS_{et}[r]$
d_{c_1}, r_{c_1} = ont1:Book, ont3:Person
d_{c_1}, r_{c_2} = ont1:Book, ont4:Persona
d_{c_2}, r_{c_1} = ont2:Libro, ont3:Person
d_{c_2}, r_{c_1} = ont2:Libro, ont4:Persona
Line 4
$get\_edges(G, c_d, c_r)$
$op(d_{c_1}, r_{c_1}) = \langle$ont1:has_author, $DS_1\rangle$
$op(d_{c_1}, r_{c_2}) = \langle$ont3:creator, $DS_2\rangle$
$op(d_{c_2}, r_{c_1}) = \langle$ont4:autor, $DS_3\rangle$
$\in C_{op}$
Line 5
map property with candidates
C_{op}[has_author] ← [⟨ont1:has_author, DS_1⟩, ⟨ont3:creator, DS_2⟩, ⟨ont4:autor, DS_3⟩]

Figure C.5: Overview of the object property candidate generation process with an example.

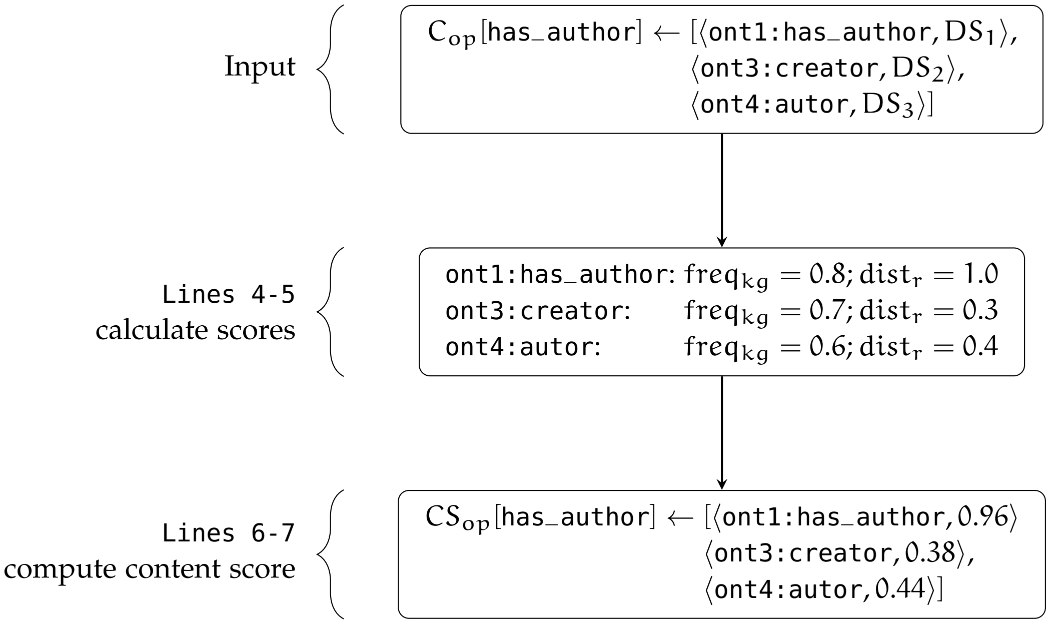

Input
C_{op}[has_author] ← [⟨ont1:has_author, DS_1⟩, ⟨ont3:creator, DS_2⟩, ⟨ont4:autor, DS_3⟩]
Lines 4-5
calculate scores
ont1:has_author: $freq_{kg} = 0.8; dist_r = 1.0$
ont3:creator: $freq_{kg} = 0.7; dist_r = 0.3$
ont4:autor: $freq_{kg} = 0.6; dist_r = 0.4$
Lines 6-7
compute content score
CS_{op}[has_author] ← [⟨ont1:has_author, 0.96⟩ ⟨ont3:creator, 0.38⟩, ⟨ont4:autor, 0.44⟩]

Figure C.6: Overview of the object property candidate content scoring with an example.

domain and range. At the end of this step, the data model DM proposed is more consistent in the entity types it suggests for the overall model.

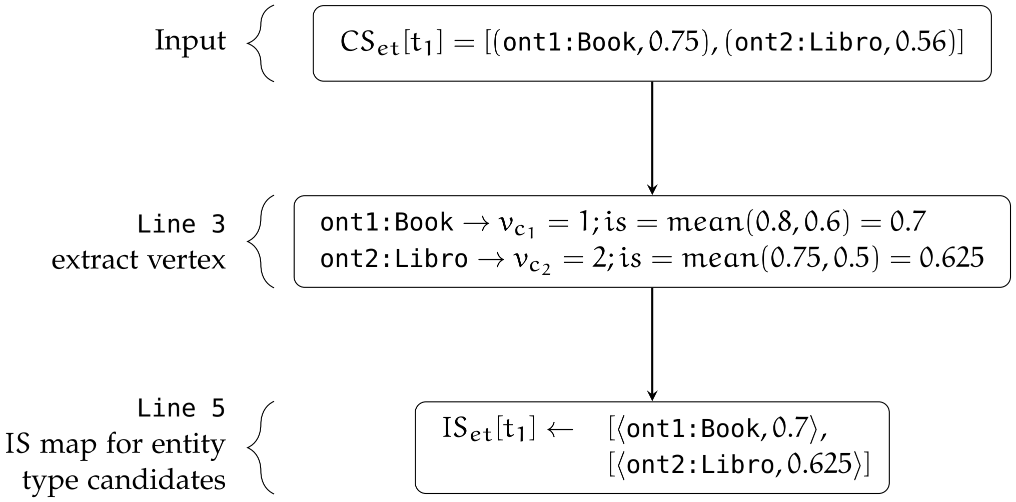

Input
$CS_{et}[t_1] = [($ont1:Book$, 0.75), ($ont2:Libro$, 0.56)]$
Line 3
extract vertex
ont1:Book $\rightarrow v_{c_1} = 1; is = mean(0.8, 0.6) = 0.7$
ont2:Libro $\rightarrow v_{c_2} = 2; is = mean(0.75, 0.5) = 0.625$
Line 5
IS map for entity
type candidates
$IS_{et}[t_1] \leftarrow [\langle$ont1:Book$, 0.7\rangle,$
$[\langle$ont2:Libro$, 0.625\rangle]$

Figure C.7: Overview of the interoperability scoring with an example.

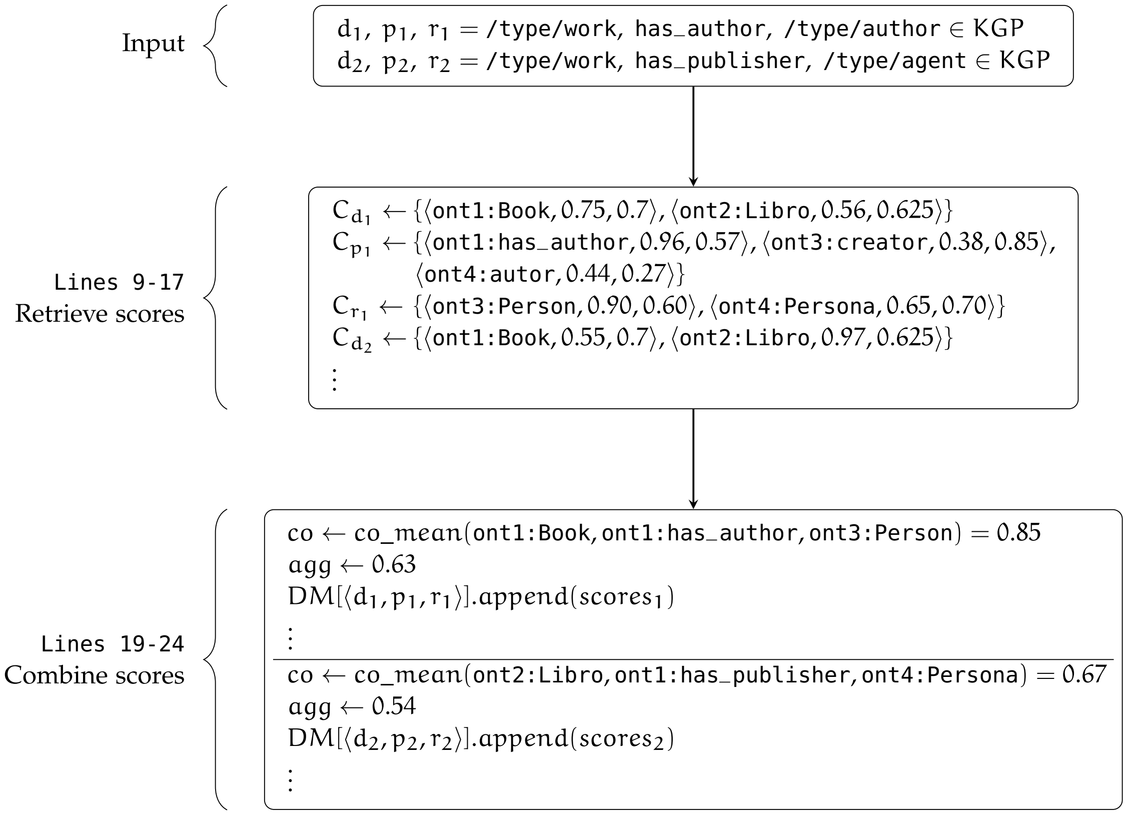

Input
$d_1, p_1, r_1 =$ /type/work, has_author, /type/author $\in$ KGP
$d_2, p_2, r_2 =$ /type/work, has_publisher, /type/agent $\in$ KGP
Lines 9-17
Retrieve scores
$C_{d_1} \leftarrow \{\langle$ont1:Book$, 0.75, 0.7\rangle, \langle$ont2:Libro$, 0.56, 0.625\rangle\}$
$C_{p_1} \leftarrow \{\langle$ont1:has_author$, 0.96, 0.57\rangle, \langle$ont3:creator$, 0.38, 0.85\rangle,$
$\langle$ont4:autor$, 0.44, 0.27\rangle\}$
$C_{r_1} \leftarrow \{\langle$ont3:Person$, 0.90, 0.60\rangle, \langle$ont4:Persona$, 0.65, 0.70\rangle\}$
$C_{d_2} \leftarrow \{\langle$ont1:Book$, 0.55, 0.7\rangle, \langle$ont2:Libro$, 0.97, 0.625\rangle\}$
$\vdots$
Lines 19-24
Combine scores
$co \leftarrow co_mean($ont1:Book, ont1:has_author, ont3:Person$) = 0.85$
$agg \leftarrow 0.63$
$DM[\langle d_1, p_1, r_1\rangle].append(scores_1)$
$\vdots$
$co \leftarrow co_mean($ont2:Libro, ont1:has_publisher, ont4:Persona$) = 0.67$
$agg \leftarrow 0.54$
$DM[\langle d_2, p_2, r_2\rangle].append(scores_2)$
$\vdots$

Figure C.8: Overview of the aggregation phase of the consistency scoring with an example.

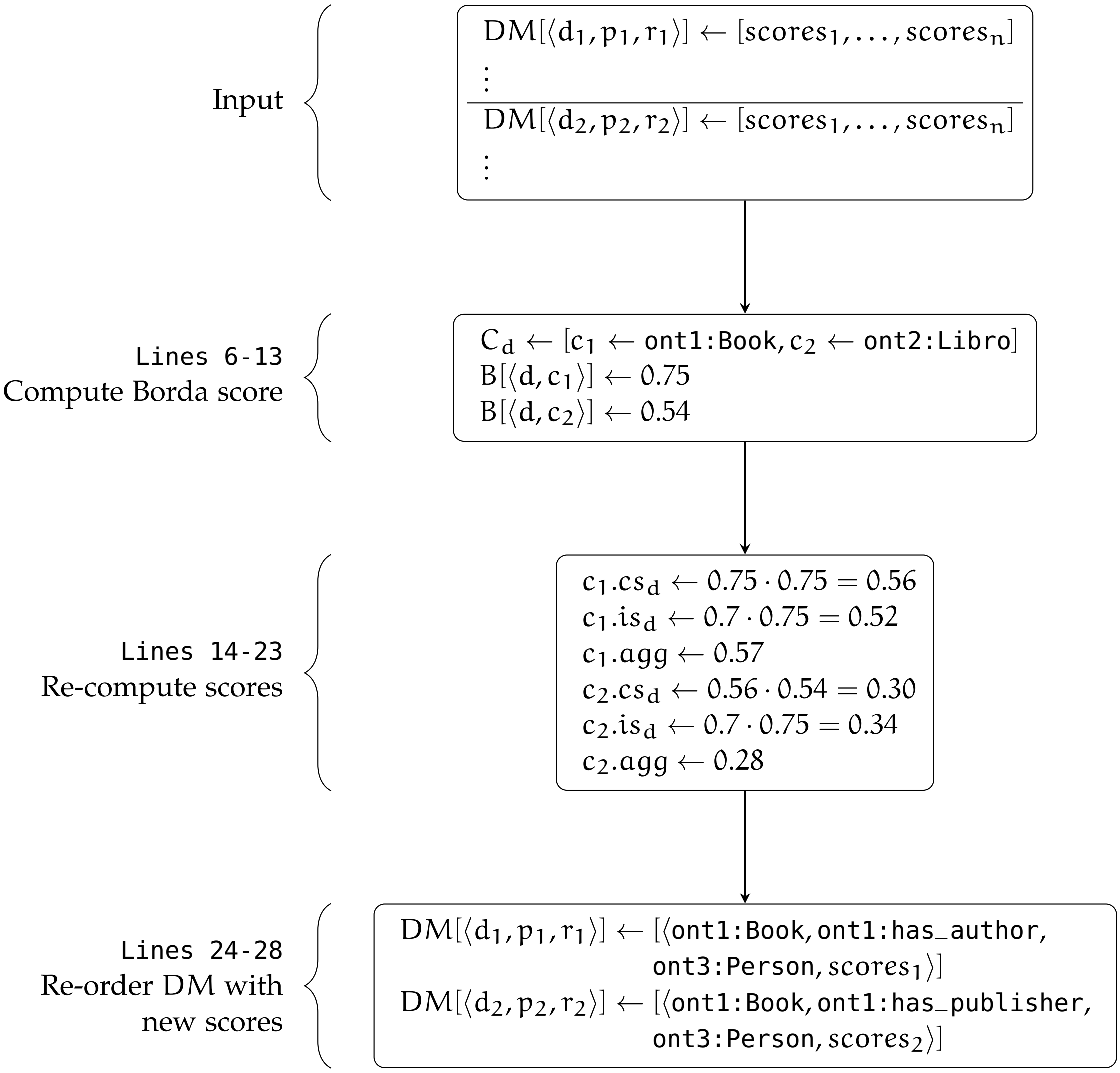

Input
$\mathrm{DM}[\langle d_1, p_1, r_1 \rangle] \leftarrow [\mathit{scores}_1, \ldots, \mathit{scores}_n]$
$\vdots$
$\mathrm{DM}[\langle d_2, p_2, r_2 \rangle] \leftarrow [\mathit{scores}_1, \ldots, \mathit{scores}_n]$
$\vdots$
Lines 6-13
Compute Borda score
$C_d \leftarrow [c_1 \leftarrow$ ont1:Book, $c_2 \leftarrow$ ont2:Libro]
$B[\langle d, c_1 \rangle] \leftarrow 0.75$
$B[\langle d, c_2 \rangle] \leftarrow 0.54$
Lines 14-23
Re-compute scores
$c_1.cs_d \leftarrow 0.75 \cdot 0.75 = 0.56$
$c_1.is_d \leftarrow 0.7 \cdot 0.75 = 0.52$
$c_1.agg \leftarrow 0.57$
$c_2.cs_d \leftarrow 0.56 \cdot 0.54 = 0.30$
$c_2.is_d \leftarrow 0.7 \cdot 0.75 = 0.34$
$c_2.agg \leftarrow 0.28$
Lines 24-28
Re-order DM with new scores
$\mathrm{DM}[\langle d_1, p_1, r_1 \rangle] \leftarrow [\langle$ ont1:Book, ont1:has_author, ont3:Person, $\mathit{scores}_1 \rangle]$
$\mathrm{DM}[\langle d_2, p_2, r_2 \rangle] \leftarrow [\langle$ ont1:Book, ont1:has_publisher, ont3:Person, $\mathit{scores}_2 \rangle]$

Figure C.9: Overview of the refinement phase of the consistency scoring with an example.

BIBLIOGRAPHY

[1] S. Brin and L. Page. 'The anatomy of a large-scale hypertextual Web search engine'. In: *Computer Networks and ISDN Systems*. Proceedings of the Seventh International World Wide Web Conference 30.1 (1st Apr. 1998), pp. 107–117.

[2] R. Blanco, P. Mika and S. Vigna. 'Effective and Efficient Entity Search in RDF Data'. In: *The Semantic Web – ISWC 2011*. Ed. by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy and E. Blomqvist. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 83–97.

[3] R. Guha, R. McCool and E. Miller. 'Semantic search'. In: *Proceedings of the 12th international conference on World Wide Web*. WWW '03. Budapest, Hungary: Association for Computing Machinery, 20th May 2003, pp. 700–709.

[4] J. Pound, P. Mika and H. Zaragoza. 'Ad-hoc object retrieval in the web of data'. In: *Proceedings of the 19th international conference on World wide web - WWW '10*. the 19th international conference. Raleigh, North Carolina, USA: ACM Press, 2010, p. 771.

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer. 'DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia'. In: *Semantic Web* 6.2 (2015), pp. 167–195.

[6] F. M. Suchanek, G. Kasneci and G. Weikum. 'YAGO: A Core of Semantic Knowledge'. In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. New York, NY, USA: ACM, 2007, pp. 697–706.

[7] D. Vrandečić and M. Krötzsch. 'Wikidata: a free collaborative knowledgebase'. In: *Communications of the ACM* 57.10 (23rd Sept. 2014), pp. 78–85.

[8] T. Berners-Lee and M. Fischetti. *Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor*. 1st. San Francisco: Harper, 1999. 226 pp.

[9] T. Berners-Lee. *Linked Data*. 27th July 2006. URL: `https://www.w3.org/DesignIssues/LinkedData.html` (visited on 15th May 2020).

[10] T. R. Gruber. 'A translation approach to portable ontology specifications'. In: *Knowledge Acquisition* 5.2 (1st June 1993), pp. 199–220.

[11] M. S. Marshall, R. Boyce, H. F. Deus, J. Zhao, E. L. Willighagen, M. Samwald, E. Pichler, J. Hajagos, E. Prud'hommeaux and S. Stephens. 'Emerging practices for mapping and linking life sciences data using RDF — A case series'. In: *Journal of Web Semantics*. Special Issue on Dealing with the Messiness of the Web of Data 14 (1st July 2012), pp. 2–13.

[12] W. Hu, H. Qiu and M. Dumontier. 'Link Analysis of Life Science Linked Data'. In: *International Semantic Web Conference*. Ed. by M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab. Vol. 9367. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 446–462.

[13] M. Krötzsch and V. Thost. 'Ontologies for Knowledge Graphs: Breaking the Rules'. In: *The Semantic Web – ISWC 2016*. ISWC 2016. Ed. by P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck and Y. Gil. Vol. 9981. Lecture Notes in Computer Science. Cham: Springer, 2016, pp. 376–392.

[14] K. Smith-Yoshimura. 'Analysis of International Linked Data Survey for Implementers'. In: *D-Lib Magazine* 22.7 (July 2016).

[15] K. Smith-Yoshimura. 'Analysis of 2018 International Linked Data Survey for Implementers'. In: *The Code4Lib Journal* 42 (8th Nov. 2018).

[16] L. McKenna, C. Debruyne and D. O'Sullivan. 'Understanding the Position of Information Professionals with regards to Linked Data: A Survey of Libraries, Archives and Museums'. In: *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*. JCDL '18. New York, NY, USA: Association for Computing Machinery, 23rd May 2018, pp. 7–16.

[17] M. d'Aquin, A. Adamou and S. Dietze. 'Assessing the educational linked data landscape'. In: *Proceedings of the 5th Annual ACM Web Science Conference*. WebSci '13. Paris, France: Association for Computing Machinery, 2nd May 2013, pp. 43–46.

[18] I. Ullah, S. Khusro, A. Ullah and M. Naeem. 'An Overview of the Current State of Linked and Open Data in Cataloging'. In: *Information Technology and Libraries* 37.4 (17th Dec. 2018), pp. 47–80.

[19] C. A. Knoblock and P. Szekely. 'Exploiting Semantics for Big Data Integration'. In: *AI Magazine* 36.1 (25th Mar. 2015), pp. 25–38.

[20] A. Sadeghi, C. Lange, M.-E. Vidal and S. Auer. 'Integration of Scholarly Communication Metadata Using Knowledge Graphs'. In: *Research and Advanced Technology for Digital Libraries*. Vol. 10450. Cham: Springer, 2017, pp. 328–341.

[21] M. d'Aquin and N. F. Noy. 'Where to Publish and Find Ontologies? A Survey of Ontology Libraries'. In: *Web semantics (Online)* 11 (1st Mar. 2012), pp. 96–111.

[22] D. Oliveira, A. S. Butt, A. Haller, D. Rebholz-Schuhmann and R. Sahay. 'Where to search top-K biomedical ontologies?' In: *Briefings in Bioinformatics* 20.4 (19th July 2019), pp. 1477–1491.

[23] D. Collarana, C. Lange and S. Auer. 'FuhSen: A Platform for Federated, RDF-based Hybrid Search'. In: *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*. the 25th International Conference Companion. Montréal, Québec, Canada: ACM Press, 2016, pp. 171–174.

[24] S. Neumaier and A. Polleres. 'Enabling Spatio-Temporal Search in Open Data'. In: *Journal of Web Semantics* 55 (1st Mar. 2019), pp. 21–36.

[25] M. Hallo, S. Luján-Mora, A. Maté and J. Trujillo. 'Current state of Linked Data in digital libraries'. In: *Journal of Information Science* 42.2 (1st Apr. 2016), pp. 117–127.

[26] H. Park and M. Kipp. 'Library Linked Data Models: Library Data in the Semantic Web'. In: *Cataloging & Classification Quarterly* 57.5 (4th July 2019), pp. 261–277.

[27] D. Oliveira, R. Sahay and M. d'Aquin. 'Leveraging Ontologies for Knowledge Graph Schemas'. In: *Proceedings of the 1st Workshop on Knowledge Graph Building co-located with ESWC 2019*. ESWC. Vol. 2489. CEUR Workshop Proceedings. Portoroz, Slovenia: CEUR-WS.org, 2019, pp. 24–36.

[28] D. Oliveira and M. D'Aquin. 'ADOG - Annotating Data with Ontologies and Graphs'. In: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference*. ISWC. Vol. 2553. CEUR Workshop Proceedings. CEUR-WS.org, 2019, p. 6.

[29] D. Oliveira and M. d'Aquin. 'RICDaM: Recommending Interoperable and Consistent Data Models'. In: *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020)*. International Semantic Web Conference. Vol. 2717. CEUR Workshop Proceedings. CEUR-WS.org, 2020, p. 5.

[30] R. Ackoff. 'From Data to Wisdom'. In: *Journal of Applied Systems Analysis* 16 (1989), pp. 3–9.

[31] J. Rowley. 'The wisdom hierarchy: representations of the DIKW hierarchy'. In: *Journal of Information Science* 33.2 (1st Apr. 2007). Publisher: SAGE Publications Ltd, pp. 163–180.

[32] M. Frické. 'The knowledge pyramid: a critique of the DIKW hierarchy'. In: *Journal of Information Science* 35.2 (1st Apr. 2009). Publisher: SAGE Publications Ltd, pp. 131–142.

[33] I. Tuomi. 'Data is More Than Knowledge: Implications of the Reversed Knowledge Hierarchy for Knowledge Management and Organizational Memory'. In: *Journal of Management Information Systems* 16 (1999), pp. 103–117.

[34] M. E. Jennex and S. E. Bartczak. 'A Revised Knowledge Pyramid'. In: *International Journal of Knowledge Management (IJKM)* 9.3 (2013). Publisher: IGI Global, pp. 19–30.

[35] D. C. Tsichritzis and F. H. Lochovsky. *Data models*. Prentice-Hall software series. Englewood Cliffs, N.J: Prentice-Hall, 1982. 381 pp.

[36] E. F. Codd. 'A relational model of data for large shared data banks'. In: *Communications of the ACM* 13.6 (1st June 1970), pp. 377–387.

[37] D. R. Howe. *Data analysis for database design*. 3rd ed. Oxford ; Boston: Butterworth Heinemann, 2001. 323 pp.

[38] C. J. Date. *An introduction to database systems*. 8th ed. Boston: Pearson/Addison Wesley, 2004. 983 pp.

[39] D. Tsichritzis and A. Klug. 'The ANSI/X3/SPARC DBMS framework report of the study group on database management systems'. In: *Information Systems* 3.3 (1st Jan. 1978), pp. 173–191.

[40] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta and D. Lin. 'Knowledge Base Completion via Search-Based Question Answering'. In: (2014).

[41] P. P.-S. Chen. 'The entity-relationship model—toward a unified view of data'. In: *ACM Transactions on Database Systems* 1.1 (1st Mar. 1976), pp. 9–36.

[42] B.-H. Yoon, S.-K. Kim and S.-Y. Kim. 'Use of Graph Database for the Integration of Heterogeneous Biological Data'. In: *Genomics & Informatics* 15.1 (Mar. 2017), pp. 19–27.

[43] T. Berners-Lee. *Relational Databases and the Semantic Web*. Design Issues. Sept. 1998. URL: `https://www.w3.org/DesignIssues/RDB-RDF.html` (visited on 28th Apr. 2020).

[44] D. Klein and C. D. Manning. 'Natural language grammar induction with a generative constituent-context model'. In: *Pattern Recognition* 38.9 (1st Sept. 2005), pp. 1407–1419.

[45] C. Bizer, T. Heath and T. Berners-Lee. 'Linked Data - The Story So Far'. In: *International Journal on Semantic Web and Information Systems* 5.3 (July 2009), pp. 1–22.

[46] M. Schmachtenberg, C. Bizer and H. Paulheim. 'Adoption of the linked data best practices in different topical domains'. In: *International Semantic Web Conference*. Springer, 2014, pp. 245–260.

[47] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres and S. Decker. 'An empirical survey of Linked Data conformance'. In: *Journal of Web Semantics*. Special Issue on Dealing with the Messiness of the Web of Data 14 (1st July 2012), pp. 14–44.

[48] A. Haller, J. D. Fernández, M. R. Kamdar and A. Polleres. 'What Are Links in Linked Open Data? A Characterization and Evaluation of Links between Knowledge Graphs on the Web'. In: *Journal of Data and Information Quality* 12.2 (6th May 2020), pp. 1–34.

[49] J. Euzenat. 'Towards a principled approach to semantic interoperability'. In: *Proc. IJCAI 2001 workshop on ontology and information sharing*. IJCAI. Seattle, United States, 2001, p. 8.

[50] N. Noy. 'Semantic integration: a survey of ontology-based approaches'. In: *ACM SIGMOD Record* 33.4 (1st Dec. 2004), pp. 65–70.

[51] H. Wache, T. J. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner. 'Ontology-Based Integration of Information - A Survey of Existing Approaches'. In: *OIS@IJCAI*. 2001.

[52] W. W. W. C. (W3C). *Vocabularies*. World Wide Web Consortium (W3C). URL: `https://www.w3.org/standards/semanticweb/ontology` (visited on 20th May 2020).

[53] N. Guarino. 'Formal Ontology and Information Systems'. In: *Proceedings of FOIS'98*. FOIS. Trento, Italy: IOS Press, 1998, p. 13.

[54] D. Oberle. 'How ontologies benefit enterprise applications'. In: *Semantic Web* 5.6 (2014), pp. 473–491.

[55] R. Hoehndorf, P. N. Schofield and G. V. Gkoutos. 'The role of ontologies in biological and biomedical research: a functional perspective'. In: *Briefings in Bioinformatics* 16.6 (Nov. 2015), pp. 1069–1080.

[56] B. M. Konopka. 'Biomedical ontologies—A review'. In: *Biocybernetics and Biomedical Engineering* 35.2 (1st Jan. 2015), pp. 75–86.

[57] T. S. De Silva, D. MacDonald, G. Paterson, K. C. Sikdar and B. Cochrane. 'Systematized nomenclature of medicine clinical terms (SNOMED CT) to represent computed tomography procedures'. In: *Computer Methods and Programs in Biomedicine* 101.3 (1st Mar. 2011), pp. 324–329.

[58] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill et al. 'Gene Ontology: tool for the unification of biology'. In: *Nature genetics* 25.1 (May 2000), pp. 25–29.

[59] N. Guarino. 'Semantic matching: Formal ontological distinctions for information organization, extraction, and integration'. In: *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*. Ed. by M. T. Pazienza. Vol. 1299. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 139–170.

[60] A. Haller and A. Polleres. 'Are We Better Off With Just One Ontology on the Web?' In: *Semantic Web Journal* 11.1 (2020).

[61] R. Arp, B. Smith and A. D. Spear. *Building ontologies with Basic Formal Ontology*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 2015. 220 pp.

[62] D. Brickley and L. Miller. *FOAF Vocabulary Specification 0.99*. 14th Jan. 2014. URL: `http://xmlns.com/foaf/spec/` (visited on 17th Aug. 2020).

[63] A. Ghazvinian, N. F. Noy and M. A. Musen. 'How orthogonal are the OBO Foundry ontologies?' In: *Journal of Biomedical Semantics* 2 (Suppl 2 17th May 2011), S2.

[64] C. Ochs, Y. Perl, J. Geller, S. Arabandi, T. Tudorache and M. A. Musen. 'An empirical analysis of ontology reuse in BioPortal'. In: *Journal of Biomedical Informatics* 71 (July 2017), pp. 165–177.

[65] M. R. Kamdar, T. Tudorache and M. A. Musen. 'A systematic analysis of term reuse and term overlap across biomedical ontologies'. In: *Semantic Web* 8.6 (7th Aug. 2017). Ed. by G.-Q. Zhang, pp. 853–871.

[66] J. Euzenat. *Ontology matching*. OCLC: ocn124038270. Berlin ; New York: Springer, 2007. 333 pp.

[67] S. Hertling and H. Paulheim. 'The Knowledge Graph Track at OAEI: Gold Standards, Baselines, and the Golden Hammer Bias'. In: *The Semantic Web*. ESWC 2020. Ed. by A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase and M. Cochez. Vol. 12123. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 343–359.

[68] A. Algergawy, D. Faria, A. Ferrara, I. Fundulaki, I. Harrow, S. Hertling, E. Jimenez-Ruiz, N. Karam, A. Khiat, P. Lambrix, H. Li, S. Montanelli et al. 'Results of the Ontology Alignment Evaluation Initiative 2019'. In: *Proceedings of the 14th International Workshop on Ontology Matching co-located with ISWC 2019*. International Semantic Web Conference. Vol. 2536. Auckland, New Zealand: CEUR Workshop Proceedings, 2019, p. 40.

[69] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz and F. M. Couto. 'The AgreementMakerLight Ontology Matching System'. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. Springer, Berlin, Heidelberg, 9th Sept. 2013, pp. 527–541.

[70] M. Lefrançois, A. Zimmermann and N. Bakerally. 'A SPARQL Extension for Generating RDF from Heterogeneous Formats'. In: *The Semantic Web*. Ed. by E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler and O. Hartig. Vol. 10249. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 35–50.

[71] S. Das, S. Sundara and R. Cyganiak. *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. 27th Sept. 2012. URL: `https://www.w3.org/TR/r2rml/` (visited on 14th July 2020).

[72] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle. 'RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data.' In: *LDOW*. 2014.

[73] P. Heyvaert, B. De Meester, A. Dimou and R. Verborgh. 'Declarative Rules for Linked Data Generation at Your Fingertips!' In: *The Semantic Web: ESWC 2018 Satellite Events*. Ed. by A. Gangemi, A. L. Gentile, A. G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J. Z. Pan and M. Alam. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 213–217.

[74] L. F. de Medeiros, F. Priyatna and O. Corcho. 'MIRROR: Automatic R2RML Mapping Generation from Relational Databases'. In: *Engineering the Web in the Big Data Era*. Ed. by P. Cimiano, F. Frasincar, G.-J. Houben and D. Schwabe. Vol. 9114. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 326–343.

[75] F. Michel, L. Djimenou, C. Faron Zucker and J. Montagnat. 'Translation of Relational and Non-Relational Databases into RDF with xR2RML'. In: *11th International Confenrence on Web Information Systems and Technologies (WEBIST'15)*. Proceedings of the WebIST'15 Conference. Lisbon, Portugal, Oct. 2015, pp. 443–454.

[76] G. Haesendonck, W. Maroy, P. Heyvaert, R. Verborgh and A. Dimou. 'Parallel RDF generation from heterogeneous big data'. In: *Proceedings of the International Workshop on Semantic Big Data - SBD '19*. the International Workshop. Amsterdam, Netherlands: ACM Press, 2019, pp. 1–6.

[77] B.-L. Do, P. R. Aryan, T.-D. Trinh, P. Wetz, E. Kiesling and A. M. Tjoa. 'Toward a framework for statistical data integration'. In: *Proceedings of the 3rd International Workshop on Semantic Statistics*. ISWC. Vol. 1551. Bethlehem, U.S.A.: CEUR Workshop Proceedings, 2015, p. 12.

[78] A. Iglesias-Molina, D. Chaves-Fraga, F. Priyatna and O. Corcho. 'Enhancing the Maintainability of the Bio2RDF Project Using Declarative Mappings'. In: Semantic Web Applications and Tools for Healthcare and Life Sciences. Edinburgh, Scotland, 2019, p. 11.

[79] A. Schultz, A. Matteini, R. Isele, P. N. Mendes, C. Bizer and C. Becker. 'LDIF - A Framework for Large-Scale Linked Data Integration'. In: *21st International World Wide Web Conference (WWW2012), Developers Track*. Lyon, France, Apr. 2012, p. 3.

[80] F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, S. Villata, B. Bucher, F. Hamdi, L. Bihanic, G. Kepeklian, F. Cotton, J. Euzenat, Z. Fan et al. 'Enabling Linked Data Publication with the Datalift Platform'. In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Conference on Artificial Intelligence. Ontario, Canada, 2012, p. 6.

[81] P.-Y. Vandenbussche, G. A. Atemezing, M. Poveda-Villalón and B. Vatant. 'Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web'. In: *Semantic Web* 8.3 (6th Dec. 2016). Ed. by M. Dumontier, pp. 437–452.

[82] M. B. Ellefi, Z. Bellahsene and K. Todorov. 'Datavore: A Vocabulary Recommender Tool Assisting Linked Data Modeling'. In: ISWC: International Semantic Web Conference. Bethlehem, PA, United States, 2015, p. 5.

[83] P. L. Whetzel, N. F. Noy, N. H. Shah, P. R. Alexander, C. Nyulas, T. Tudorache and M. A. Musen. 'BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications'. In: *Nucleic Acids Research* 39 (Web Server issue 1st July 2011), W541–W545.

[84] S. Jupp, T. Burdett, C. Leroy and H. E. Parkinson. 'A new Ontology Lookup Service at EMBL-EBI.' In: *SWAT4LS*. 2015, pp. 118–119.

[85] M. Mountantonakis and Y. Tzitzikas. 'Large-scale Semantic Integration of Linked Data: A Survey'. In: *ACM Computing Surveys* 52.5 (13th Sept. 2019), pp. 1–40.

[86] H. Paulheim. 'Knowledge graph refinement: A survey of approaches and evaluation methods'. In: *Semantic web* 8.3 (2017), pp. 489–508.

[87] C. D. Manning, P. Raghavan and H. Schutze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

[88] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita and G. Attardi. 'Ranking very many typed entities on Wikipedia'. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. CIKM '07. Lisbon, Portugal: Association for Computing Machinery, 6th Nov. 2007, pp. 1015–1018.

[89] H. Halpin, D. M. Herzig, P. Mika, R. Blanco, J. Pound, H. S. Thompson and T. T. Duc. 'Evaluating Ad-Hoc Object Retrieval'. In: *Proceedings of the International Workshop on Evaluation of Semantic Technologies*. 2010, p. 12.

[90] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi and J. Sachs. 'Swoogle: a search and metadata engine for the semantic web'. In: *Proceedings of the Thirteenth ACM conference on Information and knowledge management - CIKM '04*. the Thirteenth ACM conference. Washington, D.C., USA: ACM Press, 2004, p. 652.

[91] G. Tummarello, R. Delbru and E. Oren. 'Sindice.com: Weaving the Open Linked Data'. In: *The Semantic Web*. Ed. by K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 4825. Series Title: Lecture Notes in

Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 552–565.

[92] M. d'Aquin and E. Motta. 'Watson, more than a Semantic Web search engine'. In: *Semantic Web* 2.1 (2011), pp. 55–63.

[93] A. Harth, J. Umbrich, A. Hogan and S. Decker. 'YARS2: A Federated Repository for Querying Graph Structured Data from the Web'. In: *The Semantic Web*. Ed. by K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 4825. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 211–224.

[94] Y. Qu and G. Cheng. 'Falcons Concept Search: A Practical Search Engine for Web Ontologies'. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41.4 (July 2011), pp. 810–816.

[95] M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss et al. 'Unicorn: a system for searching the social graph'. In: *Proceedings of the VLDB Endowment* 6.11 (27th Aug. 2013), pp. 1150–1161.

[96] A. Singhal. *Introducing the Knowledge Graph: things, not strings*. Official Google Blog. 16th May 2012. URL: `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html` (visited on 19th Apr. 2018).

[97] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor. 'Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge'. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1247–1250.

[98] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson and J. Taylor. 'Industry-scale Knowledge Graphs: Lessons and Challenges'. In: *ACM Queue* 17.2 (2019).

[99] L. Ehrlinger and W. Wöß. 'Towards a Definition of Knowledge Graphs'. In: *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems and the 1st International Workshop on Semantic Change & Evolving Semantics*. SEMANTiCS. Vol. 1695. Leipzig, Germany: CEUR, 2016.

[100] P. A. Bonatti, S. Decker, A. Polleres and V. Presutti. 'Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371)'. In: *Dagstuhl Reports* 8.9 (2018). In collab. with M. Wagner, pp. 29–111.

[101] M. K. Bergman. *A Common Sense View of Knowledge Graphs*. Adaptive Information, Adaptive Innovation, Adaptive Infrastructure. 1st July 2019. URL: `https://www.mkbergman.com/?p` (visited on 27th May 2020).

[102] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo et al. 'Knowledge Graphs'. In: *arXiv:2003.02320 [cs]* (4th Mar. 2020).

[103] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner and L. Pintscher. 'From Freebase to Wikidata: The Great Migration'. In: *Proceedings of the 25th International Conference on World Wide Web - WWW '16*. the 25th International Conference. Montr&#233;al, Qu&#233;bec, Canada: ACM Press, 2016, pp. 1419–1428.

[104] S. Malyshev, M. Krötzsch, L. González, J. Gonsior and A. Bielefeldt. 'Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph'. In: *The Semantic Web – ISWC 2018*. Ed. by D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee and E. Simperl. Vol. 11137. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 376–394.

[105] A. Callahan, J. Cruz-Toledo, P. Ansell and M. Dumontier. 'Bio2RDF Release 2: Improved Coverage, Interoperability and Provenance of Life Science Linked Data'. In: *The Semantic Web: Semantics and Big Data*. Ed. by P. Cimiano, O. Corcho, V. Presutti, L. Hollink and S. Rudolph. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 200–212.

[106] S. Peroni, D. Shotton and F. Vitali. 'One Year of the OpenCitations Corpus'. In: *The Semantic Web – ISWC 2017*. Ed. by C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange and J. Heflin. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 184–192.

[107] E. Hyvönen, E. Mäkelä, T. Kauppinen, O. Alm, J. Kurki, T. Ruotsalo, K. Seppälä, J. Takala, K. Puputti, H. Kuittinen, K. Viljanen, J. Tuominen et al. 'CultureSampo: A National Publication System of Cultural Heritage on the Semantic Web 2.0'. In: *The Semantic Web: Research and Applications*. Ed. by L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou and E. Simperl. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 851–856.

[108] S. Shrivastava. *Bring rich knowledge of people, places, things and local businesses to your apps*. Bing Blogs. Library Catalog: blogs.bing.com. 12th July 2017. URL: `https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps/` (visited on 5th June 2020).

[109] C. Bendtsen and S. Petrovski. *How data and AI are helping unlock the secrets of disease*. AstraZeneca Blog. Library Catalog: www.astrazeneca.com. 11th Jan. 2019. URL: `https://www.astrazeneca.com/what-science-can-do/labtalk-blog/uncategorized/how-data-and-ai-are-helping-unlock-the-secrets-of-disease.html` (visited on 5th June 2020).

[110] E. Meij. 'Understanding News Using the Bloomberg Knowledge Graph'. Invited talk at the Big Data Innovators Gathering (TheWebConf). 2019.

[111] C. Henson, S. Schmid, T. Tran and A. Karatzoglou. 'Using a Knowledge Graph of Scenes to Enable Search of Autonomous Driving Data'. In: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019)*. ISWC. Vol. 2456. Auckland, New Zealand: CEUR Workshop Proceedings, 2019, p. 2.

[112] A. Krishnan. *Making search easier*. Day One: The Amazon Blog. Library Catalog: blog.aboutamazon.com Section: Innovation. 17th Aug. 2018. URL: `https://blog.aboutamazon.com/innovation/making-search-easier` (visited on 5th June 2020).

[113] R. Pittman, A. Srivastava, S. Hewavitharana, A. Kale and S. Mansour. *Cracking the Code on Conversational Commerce*. eBay Blog. Library Catalog: www.ebayinc.com. 6th Apr. 2017. URL: `https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/` (visited on 5th June 2020).

[114] Eric Sun and Venky Iyer. *Under the Hood: The Entities Graph | Facebook*. Facebook Engineering. 6th June 2013. URL: `https://www.facebook.com/notes/facebook-engineering/under-the-hood-the-entities-graph/10151490531588920/` (visited on 17th Feb. 2020).

[115] Qi He, Bee-Chung Chen and Deepak Agarwal. *Building The LinkedIn Knowledge Graph*. LinkedIn Blog. Library Catalog: engineering.linkedin.com. 10th June 2016. URL: `https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph` (visited on 5th June 2020).

[116] V. Richardson. 'Teaching: Trends in Research'. In: *International Encyclopedia of the Social & Behavioral Sciences*. Ed. by N. J. Smelser and P. B. Baltes. Oxford: Pergamon, 1st Jan. 2001, pp. 15483–15487.

[117] G. A. Miller. 'WordNet: a lexical database for English'. In: *Communications of the ACM* 38.11 (1st Nov. 1995), pp. 39–41.

[118] Z. Aleksovski, M. Klein, W. ten Kate and F. van Harmelen. 'Matching Unstructured Vocabularies Using a Background Ontology'. In: *Managing Knowledge in a World of Networks*. Ed. by S. Staab and V. Svátek. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 4248. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 182–197.

[119] M. Sabou, M. d'Aquin and E. Motta. 'Exploring the Semantic Web as Background Knowledge for Ontology Matching'. In: *Journal on Data Semantics XI*. Ed. by S. Spaccapietra, J. Z. Pan, P. Thiran, T. Halpin, S. Staab, V. Svatek, P. Shvaiko and

J. Roddick. Vol. 5383. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 156–190.

[120] D. Faria, C. Pesquita, E. Santos, I. F. Cruz and F. M. Couto. 'Automatic Background Knowledge Selection for Matching Biomedical Ontologies'. In: *PLOS ONE* 9.11 (7th Nov. 2014). Publisher: Public Library of Science, e111226.

[121] A. Nikolov and E. Motta. 'Capturing Emerging Relations between Schema Ontologies on the Web of Data'. In: *Proceedings of the First International Workshop on Consuming Linked Data*. International Semantic Web Conference. Vol. 665. CEUR Workshop Proceedings. China: CEUR-WS.org, 2010, p. 13.

[122] S. Neumaier, J. Umbrich, J. X. Parreira and A. Polleres. 'Multi-level Semantic Labelling of Numerical Values'. In: *The Semantic Web – ISWC 2016*. Ed. by P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck and Y. Gil. Vol. 9981. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 428–445.

[123] Y. Yi, Z. Chen, J. Heflin and B. D. Davison. 'Recognizing Quantity Names for Tabular Data'. In: *Joint Proceedings of the First International Workshop on Professional Search (ProfS2018); the Second Workshop on Knowledge Graphs and Semantics for Text Retrieval, Analysis, and Understanding (KG4IR); and the International Workshop on Data Search (DATA:SEARCH'18)*. ACM SIGIR. Vol. 2127. CEUR Workshop Proceedings. Ann Arbor, Michigan, USA: CEUR-WS.org, 2018, p. 6.

[124] Z. Chen, H. Jia, J. Heflin and B. D. Davison. 'Generating Schema Labels through Dataset Content Analysis'. In: *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*. Companion of the The Web Conference 2018. Lyon, France: ACM Press, 2018, pp. 1515–1522.

[125] R. Tennant. 'A bibliographic metadata infrastructure for the twenty-first century'. In: *Library Hi Tech* 22.2 (June 2004), pp. 175–181.

[126] L. Andresen. 'After MARC – what then?' In: *Library Hi Tech* 22.1 (1st Jan. 2004), pp. 40–51.

[127] T. L. o. Congress. *MARC Code List for Relators Scheme*. Library of Congress. URL: `https://id.loc.gov/vocabulary/relators.html` (visited on 14th Aug. 2020).

[128] S. Kawashima, T. Katayama, H. Hatanaka, T. Kushida and T. Takagi. 'NBDC RDF portal: a comprehensive repository for semantic data in life sciences'. In: *Database* 2018 (1st Jan. 2018). Publisher: Oxford Academic.

[129] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault and J. Morissette. 'Bio2RDF: towards a mashup to build bioinformatics knowledge systems'. In: *Journal of Biomedical Informatics* 41.5 (Oct. 2008), pp. 706–716.

[130] L. A. P. P. Leme, G. R. Lopes, B. P. Nunes, M. A. Casanova and S. Dietze. 'Identifying Candidate Datasets for Data Interlinking'. In: *Web Engineering*. Ed. by F. Daniel, P. Dolog and Q. Li. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 354–366.

[131] M. Ben Ellefi, Z. Bellahsene, S. Dietze and K. Todorov. 'Dataset Recommendation for Data Linking: An Intensional Approach'. In: *The Semantic Web. Latest Advances and New Domains*. Ed. by H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto and C. Lange. Vol. 9678. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 36–51.

[132] D. Faria, C. Pesquita, T. Tervo, F. M. Couto and I. F. Cruz. 'AML and AMLC Results for OAEI 2019'. In: *Proceedings of the 14th International Workshop on Ontology Matching co-located with ISWC 2019*. International Semantic Web Conference. Vol. 2536. CEUR, 2019, p. 6.

[133] O. Zamazal and V. Svátek. 'The Ten-Year OntoFarm and its Fertilization within the Onto-Sphere'. In: *Journal of Web Semantics* 43 (1st Mar. 2017), pp. 46–53.

[134] A. Vennesland. 'Matcher composition for identification of subsumption relations in ontology matching'. In: *Proceedings of the International Conference on Web Intelligence - WI '17*. the International Conference. Leipzig, Germany: ACM Press, 2017, pp. 154–161.

[135] C. J. Mungall, M. Bada, T. Z. Berardini, J. Deegan, A. Ireland, M. A. Harris, D. P. Hill and J. Lomax. 'Cross-product extensions of the Gene Ontology'. In: *Journal of Biomedical Informatics* 44.1 (Feb. 2011), pp. 80–86.

[136] J. Bard, S. Y. Rhee and M. Ashburner. 'An ontology for cell types'. In: *Genome Biology* 6.2 (Jan. 2005), R21.

[137] D. J. Watts and S. H. Strogatz. 'Collective dynamics of 'small-world' networks'. In: *Nature* 393.6684 (June 1998), pp. 440–442.

[138] J. Yang and J. Leskovec. 'Defining and evaluating network communities based on ground-truth'. In: *Knowledge and Information Systems* 42.1 (1st Jan. 2015), pp. 181–213.

[139] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, The OBI Consortium, N. Leontis et al. 'The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration'. In: *Nature Biotechnology* 25.11 (Nov. 2007), pp. 1251–1255.

[140] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and Regression Trees*. 1st ed. Google-Books-ID: MGlQDwAAQBAJ. Chapman and Hall/CRC, 1984. 369 pp.

[141] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Elsevier, 1993. 313 pp.

[142] R. Stevens, P. Lord, J. Malone and N. Matentzoglu. 'Measuring expert performance at manually classifying domain entities under upper ontology classes'. In: *Journal of Web Semantics* 57 (1st Aug. 2019), p. 100469.

[143] B. D'Arcus and F. Giasson. *Bibliographic Ontology (BIBO) in RDF*. Dublin Core Metadata Initiave. 11th May 2016. URL: `https://www.dublincore.org/specifications/bibo/bibo/` (visited on 17th Aug. 2020).

[144] *SKOS Simple Knowledge Organization System Reference*. W3C Recommendation. In collab. with A. Miles and Sean Bechhofer. 18th Aug. 2009. URL: `https://www.w3.org/TR/2009/REC-skos-reference-20090818/` (visited on 17th Aug. 2020).

[145] R. J. Miller, L. M. Haas and M. A. Hernández. 'Schema Mapping as Query Discovery'. In: *Proceedings of the 26th International Conference on Very Large Data Bases*. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 10th Sept. 2000, pp. 77–88.

[146] L. Popa, Y. Velegrakis, M. A. Hernández, R. J. Miller and R. Fagin. 'Translating web data'. In: *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB '02. Hong Kong, China: VLDB Endowment, 20th Aug. 2002, pp. 598–609.

[147] H. Elmeleegy, A. Elmagarmid and J. Lee. 'Leveraging query logs for schema mapping generation in U-MAP'. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. SIGMOD '11. New York, NY, USA: Association for Computing Machinery, 12th June 2011, pp. 121–132.

[148] A. Kimmig, A. Memory, R. J. Miller and L. Getoor. 'A Collective, Probabilistic Approach to Schema Mapping'. In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017 IEEE 33rd International Conference on Data Engineering (ICDE). Apr. 2017, pp. 921–932.

[149] E. Rahm and P. A. Bernstein. 'A survey of approaches to automatic schema matching'. In: *The VLDB Journal* 10.4 (Dec. 2001), pp. 334–350.

[150] J. Pei, J. Hong and D. Bell. 'A Novel Clustering-Based Approach to Schema Matching'. In: *Advances in Information Systems*. Ed. by T. Yakhno and E. J. Neuhold. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 60–69.

[151] A. Oliveira, G. Tessarolli, G. Ghiotto, B. Pinto, F. Campello, M. Marques, C. Oliveira, I. Rodrigues, M. Kalinowski, U. Souza, L. Murta and V. Braganholo. 'An efficient similarity-based approach for comparing XML documents'. In: *Information Systems* 78 (1st Nov. 2018), pp. 40–57.

[152] G. Limaye, S. Sarawagi and S. Chakrabarti. 'Annotating and searching web tables using entities, types and relationships'. In: *Proceedings of the VLDB Endowment* 3.1 (Sept. 2010), pp. 1338–1347.

[153] V. Mulwad, T. Finin and A. Joshi. 'Semantic Message Passing for Generating Linked Data from Tables'. In: *Advanced Information Systems Engineering*. Ed. by C. Salinesi, M. C. Norrie and O. Pastor. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 7908. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 363–378.

[154] I. Ermilov and A.-C. N. Ngomo. 'TAIPAN: Automatic Property Mapping for Tabular Data'. In: *Knowledge Engineering and Knowledge Management*. EKAW. Ed. by E. Blomqvist, P. Ciancarini, F. Poggi and F. Vitali. Vol. 10024. Cham: Springer International Publishing, 2016, pp. 163–179.

[155] A. Alserafi, A. Abelló, O. Romero and T. Calders. 'Keeping the Data Lake in Form: Proximity Mining for Pre-Filtering Schema Matching'. In: *ACM Transactions on Information Systems* 38.3 (13th May 2020), 26:1–26:30.

[156] J. P. McCrae and P. Buitelaar. 'Linking Datasets Using Semantic Textual Similarity'. In: *Cybernetics and Information Technologies* 18.1 (1st Mar. 2018), pp. 109–123.

[157] N. Bikakis, C. Tsinaraki, N. Gioldasis, I. Stavrakantonakis and S. Christodoulakis. 'The XML and Semantic Web Worlds: Technologies, Interoperability and Integration: A Survey of the State of the Art'. In: *Semantic Hyper/Multimedia Adaptation: Schemes and Applications*. Ed. by I. E. Anagnostopoulos, M. Bieliková, P. Mylonas and N. Tsapatsoulis. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2013, pp. 319–360.

[158] S. Hertling and H. Paulheim. 'DBkWik: extracting and integrating knowledge from thousands of Wikis'. In: *Knowledge and Information Systems* 62.6 (1st June 2020), pp. 2169–2190.

[159] D. Vallet and H. Zaragoza. 'Inferring the Most Important Types of a Query: a Semantic Approach'. In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'08. Singapore, 2008.

[160] K. Balog and R. Neumayer. 'Hierarchical target type identification for entity-oriented queries'. In: *Proceedings of the 21st ACM International Conference on Information and knowledge management*. CIKM '12. Maui, Hawaii, USA: Association for Computing Machinery, 29th Oct. 2012, pp. 2391–2394.

[161] A. Tonon, M. Catasta, R. Prokofyev, G. Demartini, K. Aberer and P. Cudré-Mauroux. 'Contextualized ranking of entity types based on knowledge graphs'. In: *Journal of Web Semantics* 37-38 (Mar. 2016), pp. 170–183.

[162] H. Paulheim and C. Bizer. ‘Type Inference on Noisy RDF Data’. In: *Advanced Information Systems Engineering*. Ed. by C. Salinesi, M. C. Norrie and O. Pastor. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 7908. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 510–525.

[163] J. Sleeman, T. Finin and A. Joshi. ‘Entity Type Recognition for Heterogeneous Semantic Graphs’. In: *AI Magazine* 36.1 (25th Mar. 2015). Number: 1, pp. 75–86.

[164] O. Udrea, L. Getoor and R. J. Miller. ‘Leveraging data and structure in ontology integration’. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. SIGMOD ’07. New York, NY, USA: Association for Computing Machinery, 11th June 2007, pp. 449–460.

[165] F. M. Suchanek, S. Abiteboul and P. Senellart. ‘PARIS: probabilistic alignment of relations, instances, and schema’. In: *Proceedings of the VLDB Endowment* 5.3 (1st Nov. 2011), pp. 157–168.

[166] J. Lehmann, J. Schuppel and S. Auer. ‘Discovering Unknown Connections – the DBpedia Relationship Finder’. In: *Proceedings of the 1st Conference on Social Semantic Web (CSSW)*. The Social Semantic Web. Leipzig, Germany, 2007, p. 11.

[167] M. Sabou, M. d’Aquin and E. Motta. ‘SCARLET: SemantiC RelAtion DiscoveRy by Harvesting OnLinE OnTologies’. In: *The Semantic Web: Research and Applications*. Ed. by S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 854–858.

[168] D. Seo, H. K. Koo, S. Lee, P. Kim, H. Jung and W.-K. Sung. ‘Efficient Finding Relationship between Individuals in a Mass Ontology Database’. In: *U- and E-Service, Science and Technology*. Ed. by T.-h. Kim, H. Adeli, J. Ma, W.-c. Fang, B.-H. Kang, B. Park, F. E. Sandnes and K. C. Lee. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2011, pp. 281–286.

[169] B. Pereira Nunes, S. Dietze, M. A. Casanova, R. Kawase, B. Fetahu and W. Nejdl. ‘Combining a Co-occurrence-Based and a Semantic Measure for Entity Linking’. In: *The Semantic Web: Semantics and Big Data*. Ed. by P. Cimiano, O. Corcho, V. Presutti, L. Hollink and S. Rudolph. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 7882. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.

[170] L. A. Galárraga, N. Preda and F. M. Suchanek. ‘Mining rules to align knowledge bases’. In: *Proceedings of the 2013 workshop on Automated knowledge base construction*. AKBC ’13. New York, NY, USA: Association for Computing Machinery, 27th Oct. 2013, pp. 43–48.

[171] M. Koutraki, N. Preda and D. Vodislav. 'Online Relation Alignment for Linked Datasets'. In: *The Semantic Web*. Ed. by E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler and O. Hartig. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 152–168.

[172] K. Gunaratna, K. Thirunarayan, P. Jain, A. Sheth and S. Wijeratne. 'A statistical and schema independent approach to identify equivalent properties on linked data'. In: *Proceedings of the 9th International Conference on Semantic Systems - I-SEMANTICS '13*. the 9th International Conference. Graz, Austria: ACM Press, 2013, p. 33.

[173] B. Pereira Nunes, A. Mera, M. A. Casanova, B. Fetahu, L. A. P. Paes Leme and S. Dietze. 'Complex Matching of RDF Datatype Properties'. In: *Database and Expert Systems Applications*. DEXA. Ed. by H. Decker, L. Lhotská, S. Link, J. Basl and A. M. Tjoa. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos et al. Vol. 8055. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 195–208.

[174] Z. Syed, T. Finin, V. Mulwad and A. Joshi. 'Exploiting a Web of Semantic Data for Interpreting Tables'. In: *Proceedings of the Second Web Science Conference*. Web Science Conference. Raleigh, NC, USA, 2010.

[175] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao and C. Wu. 'Recovering semantics of tables on the web'. In: *Proceedings of the VLDB Endowment* 4.9 (June 2011), pp. 528–538.

[176] M. D. Adelfio and H. Samet. 'Schema extraction for tabular data on the web'. In: *Proceedings of the VLDB Endowment* 6.6 (Apr. 2013), pp. 421–432.

[177] T. Mikolov, K. Chen, G. Corrado and J. Dean. 'Efficient Estimation of Word Representations in Vector Space'. In: *arXiv:1301.3781 [cs]* (6th Sept. 2013).

[178] J. Pennington, R. Socher and C. Manning. 'GloVe: Global Vectors for Word Representation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[179] D. G. Saari. *Geometry of Voting*. OCLC: 903196450. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994.

[180] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen and K. Srinivas. 'SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems'. In: *The Semantic Web*. Ed. by A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase and M. Cochez. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 514–530.

[181] O. Hassanzadeh, V. Efthymiou, J. Chen, E. Jiménez-Ruiz and K. Srinivas. *SemTab2019: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching - 2019 Data Sets*. type: dataset. 25th Oct. 2019. URL: `https://zenodo.org/record/3518539` (visited on 3rd Sept. 2020).

[182] O. Bodenreider. 'Biomedical ontologies in action: role in knowledge management, data integration and decision support.' In: *Yearbook of medical informatics* (2008), pp. 67–79.

[183] O. Corcho, M. Fernández-López and A. Gómez-Pérez. 'Methodologies, tools and languages for building ontologies. Where is their meeting point?' In: *Data & Knowledge Engineering* 46.1 (1st July 2003), pp. 41–64.

[184] A. C. Yu. 'Methods in biomedical ontology'. In: *Journal of Biomedical Informatics*. Biomedical Ontologies 39.3 (1st June 2006), pp. 252–266.

[185] E. Simperl. 'Reusing ontologies on the Semantic Web: A feasibility study'. In: *Data & Knowledge Engineering* 68.10 (1st Oct. 2009), pp. 905–925.

[186] M. Fernández-López, M. Poveda-Villalón, M. C. Suárez-Figueroa and A. Gómez-Pérez. 'Why are ontologies not reused across the same domain?' In: *Journal of Web Semantics* 57 (1st Aug. 2019), p. 100492.

[187] M. Poveda-Villalón, M. C. Suárez-Figueroa and A. Gómez-Pérez. 'The Landscape of Ontology Reuse in Linked Data'. In: *Proceedings Ontology Engineering in a Data-driven World*. OEDW. 2012, p. 11.

[188] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. L. Rector and C. Rosse. 'Relations in biomedical ontologies'. In: *Genome Biology* 6 (28th Apr. 2005), R46.

[189] P. Szolovits. *Artificial Intelligence In Medicine*. Google-Books-ID: 8tmiDwAAQBAJ. Routledge, 13th Mar. 2019. 255 pp.

[190] N. Guarino and C. Welty. 'Evaluating ontological decisions with OntoClean'. In: *Communications of the ACM* 45.2 (1st Feb. 2002), pp. 61–65.

[191] A. Gangemi, C. Catenacci, M. Ciaramita and J. Lehmann. 'A theoretical framework for ontology evaluation and validation'. In: *Proceedings of the 2nd Italian Semantic Web Workshop*. SWAP. Vol. 166. Trento, Italy: CEUR Workshop Proceedings, 2005, p. 16.

[192] H. Alani, C. Brewster and N. Shadbolt. 'Ranking Ontologies with AKTiveRank'. In: *The Semantic Web - ISWC 2006*. International Semantic Web Conference. Springer, Berlin, Heidelberg, 5th Nov. 2006, pp. 1–15.

[193] C. Patel, K. Supekar, Y. Lee and E. K. Park. 'OntoKhoj: a semantic web portal for ontology searching, ranking and classification'. In: *Proceedings of the 5th ACM international workshop on Web information and data management*. WIDM '03. New Orleans, Louisiana, USA: Association for Computing Machinery, 7th Nov. 2003, pp. 58–61.

[194] E. Thomas, J. Z. Pan and D. H. Sleeman. 'ONTOSEARCH2: Searching Ontologies Semantically.' In: *OWLED*. 2007.

[195] P. Buitelaar. 'OntoSelect: Towards the Integration of an Ontology Library, Ontology Selection and Knowledge Markup'. In: *Proceedings of the 4th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2004) located at the 3rd International Semantic Web Conference ISWC 2004*. ISWC. Vol. 184. Hiroshima, Japan: CEUR Workshop Proceedings, 2004, p. 2.

[196] P. Buitelaar and T. Eigner. 'Evaluating Ontology Search.' In: *EON*. 2007, pp. 11–20.

[197] J. Schaible, T. Gottron, S. Scheglmann and A. Scherp. 'LOVER: Support for Modeling Data Using Linked Open Vocabularies'. In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. EDBT '13. event-place: Genoa, Italy. New York, NY, USA: ACM, 2013, pp. 89–92.

[198] T. Adamusiak, T. Burdett, N. Kurbatova, K. Joeri van der Velde, N. Abeygunawardena, D. Antonakaki, M. Kapushesky, H. Parkinson and M. A. Swertz. 'OntoCAT – simple ontology search and integration in Java, R and REST/JavaScript'. In: *BMC Bioinformatics* 12.1 (29th May 2011), p. 218.

[199] N. Kurbatova, T. Adamusiak, P. Kurnosov, M. A. Swertz and M. Kapushesky. 'ontoCAT: an R package for ontology traversal and search'. In: *Bioinformatics* 27.17 (1st Sept. 2011). Publisher: Oxford Academic, pp. 2468–2470.

[200] A. S. Butt, A. Haller and L. Xie. 'Ontology Search: An Empirical Evaluation'. In: *The Semantic Web – ISWC 2014*. International Semantic Web Conference. Springer, Cham, 19th Oct. 2014, pp. 130–147.

[201] N. Kolbe, P.-Y. Vandenbussche, S. Kubler and Y. Le Traon. 'LOVBench: Ontology Ranking Benchmark'. In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 20th Apr. 2020, pp. 1750–1760.

[202] R. Petryszak, T. Burdett, B. Fiorelli, N. A. Fonseca, M. Gonzalez-Porta, E. Hastings, W. Huber, S. Jupp, M. Keays, N. Kryvych, J. McMurry, J. C. Marioni et al. 'Expression Atlas update–a database of gene and transcript expression from microarray- and sequencing-based functional genomics experiments'. In: *Nucleic Acids Research* 42 (Database issue Jan. 2014), pp. D926–932.

[203] J. MacArthur, E. Bowler, M. Cerezo, L. Gil, P. Hall, E. Hastings, H. Junkins, A. McMahon, A. Milano, J. Morales, Z. M. Pendlington, D. Welter et al. 'The new NHGRI-EBI Catalog of published genome-wide association studies (GWAS Catalog)'. In: *Nucleic Acids Research* 45 (D1 4th Jan. 2017), pp. D896–D901.

[204] G. Salton and C. Buckley. 'Term-weighting Approaches in Automatic Text Retrieval'. In: *Inf. Process. Manage.* 24.5 (Aug. 1988), pp. 513–523.

[205] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu and M. Gatford. 'Okapi at TREC–3'. In: *Microsoft Research* (1st Jan. 1995).

[206] G. Salton, A. Wong and C. S. Yang. 'A Vector Space Model for Automatic Indexing'. In: *Commun. ACM* 18.11 (Nov. 1975), pp. 613–620.

[207] L. Page, S. Brin, R. Motwani and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* 11th Nov. 1999. URL: `http://ilpubs.stanford.edu:8090/422/` (visited on 14th Mar. 2017).

[208] M. Courtot, F. Gibson, A. L. Lister, J. Malone, D. Schober, R. R. Brinkman and A. Ruttenberg. 'MIREOT: the Minimum Information to Reference an External Ontology Term'. In: *Nature Precedings* (7th Aug. 2009).

[209] M. Horridge and S. Bechhofer. 'The OWL API: A Java API for OWL Ontologies'. In: *Semantic Web* 2.1 (2011), pp. 11–21.

[210] J. Hastings, P. de Matos, A. Dekker, M. Ennis, B. Harsha, N. Kale, V. Muthukrishnan, G. Owen, S. Turner, M. Williams and C. Steinbeck. 'The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013'. In: *Nucleic Acids Research* 41 (D1 1st Jan. 2013), pp. D456–D463.

[211] L. M. Schriml, C. Arze, S. Nadendla, Y.-W. W. Chang, M. Mazaitis, V. Felix, G. Feng and W. A. Kibbe. 'Disease Ontology: a backbone for disease semantic integration'. In: *Nucleic Acids Research* 40 (Database issue Jan. 2012), pp. D940–946.

[212] J. Hanna, E. Joseph, M. Brochhausen and W. R. Hogan. 'Building a drug ontology based on RxNorm and other sources'. In: *Journal of Biomedical Semantics* 4 (18th Dec. 2013), p. 44.

[213] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer and P. Rice. 'EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats'. In: *Bioinformatics* 29.10 (15th May 2013), pp. 1325–1332.

[214] J. Malone, E. Holloway, T. Adamusiak, M. Kapushesky, J. Zheng, N. Kolesnikov, A. Zhukova, A. Brazma and H. Parkinson. 'Modeling sample variables with an Experimental Factor Ontology'. In: *Bioinformatics* 26.8 (15th Apr. 2010), pp. 1112–1118.

[215] C. Rosse and V. Mejino JL. 'A Reference Ontology for Bioinformatics: The Foundational Model of Anatomy'. In: *J Biomed Informat* 36 (2003).

[216] S. Köhler, S. C. Doelken, C. J. Mungall, S. Bauer, H. V. Firth, I. Bailleul-Forestier, G. C. M. Black, D. L. Brown, M. Brudno, J. Campbell, D. R. FitzPatrick, J. T. Eppig et al. 'The Human Phenotype Ontology project: linking molecular biology and disease through phenotype data'. In: *Nucleic Acids Research* 42 (D1 Jan. 2014), pp. D966–D974.

[217] T. F. Hayamizu, M. Mangan, J. P. Corradi, J. A. Kadin and M. Ringwald. 'The Adult Mouse Anatomical Dictionary: a tool for annotating and integrating data'. In: *Genome Biology* 6.3 (15th Feb. 2005), R29.

[218] C. L. Smith, C.-A. W. Goldsmith and J. T. Eppig. 'The Mammalian Phenotype Ontology as a tool for annotating, analyzing and comparing phenotypic information'. In: *Genome Biology* 6.1 (15th Dec. 2004), R7.

[219] P. N. Schofield, J. P. Sundberg, B. A. Sundberg, C. McKerlie and G. V. Gkoutos. 'The mouse pathology ontology, MPATH; structure and applications'. In: *Journal of Biomedical Semantics* 4 (2013), p. 18.

[220] G. V. Gkoutos, P. N. Schofield and R. Hoehndorf. 'The Neurobehavior Ontology: An Ontology for Annotation and Integration of Behavior and Behavioral Phenotypes'. In: *International Review of Neurobiology*. Ed. by E. J. Chesler and M. A. Haendel. Vol. 103. Bioinformatics of Behavior: Part 1. Academic Press, 1st Jan. 2012, pp. 69–87.

[221] N. Sioutos, S. d. Coronado, M. W. Haber, F. W. Hartel, W.-L. Shaiu and L. W. Wright. 'NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information'. In: *Journal of Biomedical Informatics*. Bio*Medical Informatics 40.1 (Feb. 2007), pp. 30–43.

[222] Y. He, S. Sarntivijai, Y. Lin, Z. Xiang, A. Guo, S. Zhang, D. Jagannathan, L. Toldo, C. Tao and B. Smith. 'OAE: The Ontology of Adverse Events'. In: *Journal of Biomedical Semantics* 5 (2014), p. 29.

[223] Y. He, Y. Liu and B. Zhao. 'OGG: a Biological Ontology for Representing Genes and Genomes in Specific Organisms.' In: *ICBO*. Citeseer, 2014, pp. 13–20.

[224] C. J. Mungall, G. V. Gkoutos, C. L. Smith, M. A. Haendel, S. E. Lewis and M. Ashburner. 'Integrating phenotype ontologies across multiple species'. In: *Genome Biology* 11.1 (2010), R2.

[225] L. Cooper, R. L. Walls, J. Elser, M. A. Gandolfo, D. W. Stevenson, B. Smith, J. Preece, B. Athreya, C. J. Mungall, S. Rensing, M. Hiss, D. Lang et al. 'The Plant Ontology as a Tool for Comparative Plant Anatomy and Genomic Analyses'. In: *Plant and Cell Physiology* 54.2 (1st Feb. 2013), e1–e1.

[226] M. A. Haendel, G. G. Gkoutos, S. E. Lewis and C. Mungall. 'Uberon: towards a comprehensive multi-species anatomy ontology'. In: *Nature Precedings* 713 (11th Aug. 2009).

[227] C. A. Park, S. M. Bello, C. L. Smith, Z.-L. Hu, D. H. Munzenmaier, R. Nigam, J. R. Smith, M. Shimoyama, J. T. Eppig and J. M. Reecy. 'The Vertebrate Trait Ontology: a controlled vocabulary for the annotation of trait data across species'. In: *Journal of Biomedical Semantics* 4 (2013), p. 13.

[228] G. Schindelman, J. S. Fernandes, C. A. Bastiani, K. Yook and P. W. Sternberg. 'Worm Phenotype Ontology: integrating phenotype data within and beyond the C. elegans community'. In: *BMC bioinformatics* 12 (24th Jan. 2011), p. 32.

[229] E. Segerdell, J. B. Bowes, N. Pollet and P. D. Vize. 'An ontology for Xenopus anatomy and development'. In: *BMC Developmental Biology* 8 (25th Sept. 2008), p. 92.

[230] C. E. Van Slyke, Y. M. Bradford, M. Westerfield and M. A. Haendel. 'The zebrafish anatomy and stage ontologies: representing the anatomy and development of Danio rerio'. In: *Journal of Biomedical Semantics* 5 (2014), p. 12.

[231] J. H. Kim. 'Chi-Square Goodness-of-Fit Tests for Randomly Censored Data'. In: *The Annals of Statistics* 21.3 (Sept. 1993), pp. 1621–1639.

[232] B. Lamiroy and T. Sun. 'Computing Precision and Recall with Missing or Uncertain Ground Truth'. In: *Proceedings of the 9th International Conference on Graphics Recognition: New Trends and Challenges*. GREC'11. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 149–162.

[233] Y. Wang, L. Wang and Y. Li. 'A Theoretical Analysis of NDCG Ranking Measures'. In: 2013.

[234] C. Gavankar, Y.-F. Li and G. Ramakrishnan. 'Explicit Query Interpretation and Diversification for Context-Driven Concept Search Across Ontologies'. In: *The Semantic Web – ISWC 2016*. ISWC. Ed. by P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck and Y. Gil. Vol. 9981. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 271–288.