Graph summarisation of web data: data-driven generation of structured representations

Campinas, Stéphane

2016-08-31

http://hdl.handle.net/10379/6495

Some rights reserved. For more information, please see the item record link above.
Graph Summarisation of Web Data: Data-driven Generation of Structured Representations

Stéphane Campinas

Under the supervision of Prof. Dr. Stefan Decker
and co-supervision of Prof. Dr. Dietrich Rebholz-Schuhmann
of Insight Centre for Data Analytics, Galway

May 4, 2017
Abstract

The advent of the Internet enabled the sharing of information between people all around the world. Projects like Wikipedia have made human knowledge accessible to anybody with a simple mouse click. The Linked Data movement has made a considerable leap in the amount of data now available on the Web. Data about science, social interactions, governments, entertainment and more is now available to anybody.

That data is described at a very fine granularity, allowing to describe precisely entities (people, films, monuments, ...) and their relationships. This marks a shift in data management on the Web: instead of a graph of web documents, we witness now a graph of entities with links carrying semantic; we call this Web Data.

Web Data is characterized by the use of the Resource Description Framework data model, which enables a dynamic management of information, allowing anyone to easily create and modify data. That flexible model facilitated the creation of data that grew organically: the structure of the data is not necessarily maintained over time, and some data may be created by integrating several existing datasources, impacting on the structure consistency of the resulting integration. This data is referred to as semi-structured.

Web Data is therefore a large collection of semi-structured and heterogeneous data. It is then difficult for a user to understand what information is contained in a particular dataset within that collection or how to access it. In relational database systems, the schema answers to that need; which, in the context of Web Data, is generally missing.

In this thesis, we propose the use of graph summarisation for highlighting the structure of a dataset, which output is referred to as the graph summary. The graph summary is a graph, generated from the data itself, which shares the same structure as the original dataset. Unlike the original graph, the summary abstracts itself from details about entities and is focused on the structure of the graph. Therefore, the graph summary is akin to a schema, which we assume is at the core of many applications such as query optimisation, data exploration, data integration, ... The generation of a graph summary is then intended to be at the core of such applications within the context of Web Data.

We present a formal model for graph summarisation and how to generate a summary. We use that model to compute precise summaries using the bisimulation equivalence relation; and we argue for the case of approximate graph summaries as the most viable option for many applications. Because graph summarisation is a technique applied directly on the data, approximate graph summaries are susceptible to the quality of the data. Hence, we introduce a model for assessing the precision of a summary with regards to the data. Finally, we develop applications that leverage summaries to demonstrate their usefulness.
Acknowledgements

This thesis would not have come to fruit without the help of many people.

I thank my parents for supporting me along this path of research and study. I wish to thank my brother and sister for their support overseas. I would like also to thank my girlfriend, now wife, Kasia for being present throughout this journey. The support of friends was priceless, with Diego Ceccarelli being noteworthy. This thesis is not only the product of my own efforts, but also of everybody I was in contact with during that time. In addition, I am thankful to Grishka for always being happy.

I wish also to thanks my supervisors Giovanni and Stefan for guiding me throughout the years, and to Dietrich for providing me with valuable feedback and supporting me in the last moments. Finally, my years of research would not have been the same without the support, guidance, and patience of my mentor Renaud.
Contents

I. Background 1

Chapter 1. Introduction 3
  1.1. Motivation 4
  1.2. Data Management 6
  1.3. Ranking of Web Data 7
  1.4. Challenges 8
  1.5. Scope of the Research 9
  1.6. Applications 10
  1.7. Scientific Methodology 10
  1.8. Outline 11
  1.9. Contributions 12

Chapter 2. Semi-Structured Data 13
  2.1. Semi-Structured Data Models 13
  2.2. Abstract Model 14
  2.3. Formal Model 15

II. Methods 21

Chapter 3. Graph Summarisation: Related Work 23
  3.1. Dimensions of Graph Summarisation Challenges 23
  3.2. Graph Summarisation Approaches 25
  3.3. Graph Exploration 26
  3.4. Data Profiling 28
  3.5. Data Analytics 30
  3.6. Query Optimisation 33
  3.7. Summary 34

Chapter 4. Graph Summary: a Novel Schema for Web Data 35
  4.1. Graph Summarisation 35
  4.2. Graph Summary Generation 52
  4.3. Conclusion and Future Work 66
Contents

Chapter 5. Graph Summary Precision 69
  5.1. Graph Summarisation Lattice 69
  5.2. Summary Error 72
  5.3. Evaluation 77
  5.4. Conclusion and Future Work 90

Chapter 6. Directed Acyclic Graphs Ranking 91
  6.1. Related Work 92
  6.2. Ranking Model Foundations 92
  6.3. Field-Based Ranking Models 93
  6.4. MF Ranking Model 97
  6.5. Weights 103
  6.6. Experiments 105
  6.7. Query-dependent Ranking of the Graph Summary 112
  6.8. Conclusion and Future Work 121

III. Putting the Pieces Together 125

Chapter 7. Graph Summary Applications 127
  7.1. Graph Summary for Web Data Management 127
  7.2. SPARQLed: Query Recommendation 132
  7.3. Web Data Inspector 141
  7.4. Conclusion 149

Chapter 8. Conclusions 151
  8.1. Summary of the Thesis 151
  8.2. Directions for Future Research 152
  8.3. Current Limitations 153

IV. Appendices 155

Appendix A. Data Tables of the Graph Summarisation Evaluation 157

Appendix B. Data Tables of the DAG Ranking Evaluation 161
  B.1. Entity Ranking 161
  B.2. Graph Summary Ranking 161

Glossary 179
List of Figures

1.1. A graph modelled as a bag of words ......................................... 9
2.1. The graph data abstract model ................................................. 15
2.2. An entity graph describing people, places, and their relationships. 17
2.3. Interpretation of the entity description dependent on the graph structure. 18
3.1. Graph summarisation dimensions ............................................. 24
3.2. Graph heterogeneity where the dashed graphs represent possible summaries of the graph ......................................................... 25
3.3. Time-line of the research on graph summarisation ......................... 26
3.4. Graph summarisation based on equivalence relation as in [ZLXH11] ... 31
4.1. Graph homomorphism ............................................................. 37
4.2. A graph and a possible summary .............................................. 39
4.3. Types summary of a graph ......................................................... 40
4.4. Illustration of the conditions for a node $v'_0$ to substitute a node $v_0$, as per the simulation $R$. If the reverse is also true, i.e., that the node $v_0$ can substitute $v'_0$, then the nodes $v_0$ and $v'_0$ are said to be bisimilar. ........ 42
4.5. Summary of the graph in Figure 2.2 with bisimulation $R_{fbt}$ as the summarisation relation. The table indicates the mappings by $R_{fbt}$. The content in $G$ is abstracted by the sumnode $\emptyset$ ......................... 45
4.6. Ontology probability distribution .............................................. 46
4.7. Summary of the graph in Figure 2.2 with $R_a$. The table indicates the mappings by $R_a$. The content in $G$ is abstracted by the sumnode $\emptyset$........ 49
4.8. Example of the Types summarisation $R_t$ in the case of a single dataset. 55
4.9. Challenge in summarising heterogeneous data sources with the Types summarisation relation $R_t$ ............................................. 56
4.10. Edge authority ...................................................................... 58
4.11. Example of the Types summarisation $R_t$ in the case of inter-linked datasets. 60
4.12. SPARQL-based graph summarisation over a single dataset ............ 63
4.13. Graph summarisation using the Cascading framework .................. 67
5.1. Summarisation relations $R_1$ and $R_2$ ordered with the binary relation $\sqsubseteq$ in Definition (5.1.1) ............................................. 70
5.2. Lattice of the summarisation relations set according to the partial order $\sqsubseteq$ ................................................................. 71
5.3. Ordering of graph summaries ..................................................... 72
5.4. The inferred graph $\mathcal{I}(R_t)$ created from the comparison of the Types and Bisimulation summaries ............................................. 74
List of Figures

5.5. Size of the Unique Type summary $S_{uf}$ ........................................ 81
5.6. Compression ratio $G : S$ of graph summaries. The ratio axis is reported in logarithmic scale. ................................................................. 83
5.7. Performance comparison of the graph summaries .......................... 84
5.8. Connectivity precision bar chart .................................................. 86
5.9. Schema precision bar chart ......................................................... 87
5.10. Efficiency and precision trade-offs of the candidate summarisation relations. The values are taken as the average across all dataset categories. .......................... 89

6.1. A graph divided into four entities identified by the root nodes $me$, $:_b1$, $:_b2$ and $paper/547$. ......................................................... 93
6.2. Depiction of the MF model ......................................................... 99
6.3. Effect of the control function over the LC coverage. The opacity of the curve gets stronger as the parameter increases. .............................. 105
6.4. Impact of the normalisation parameters on the MF ranking functions . 108
6.5. Comparison of state-of-the-art ranking functions against the MF generalizations ................................................................. 110
6.6. Evaluation of weights with the MF ranking extensions .................. 113
6.7. A solution to the SPARQL query of the running example. A number in parenthesis represents a statistic associated with that node. ..................... 119
6.8. MF normalisation parameters over simple and complex graphs ....... 121

7.1. Trade-offs between graph schema creation approaches. .................. 128
7.2. RDF reification of the graph summary ........................................ 131
7.3. Overview of possible recommendations in SPARQLed .................. 134
7.4. Web Data Inspector - home page ............................................. 142
7.5. Web Data Inspector - general view ........................................... 142
7.6. Web Data Inspector - inside this dataset ................................... 145
7.7. Web Data Inspector - to/from this dataset ................................ 146
7.8. Web Data Inspector - third party links view ............................... 148
## List of Tables

4.1. Summarisation relations for approximate graph summaries .......................... 52

5.1. Descriptive statistics of the datasets used in the evaluation of a graph summary precision ................................................................. 82

6.1. Normalisation parameters values, found through a constrained particle swarm optimization .............................................................. 109

6.2. Queries extracted from the USEWOD2013 query logs .............................. 118

6.3. Comparison of the basic and MF-based ranking approaches on the Types summary of DBpedia 3.3 ......................................................... 122

7.1. Full and lite reification of a *Types* summary $R_t$ ................................. 133

A.1. Size and order of graph summaries ........................................................... 158

A.2. Computational performance of graph summaries creation ......................... 158

A.3. Connectivity precision comparison. For each category of dataset complexity, we report the mean $\mu$ of the connectivity precision .................. 159

A.4. Schema precision comparison ................................................................. 159

B.1. Comparison of state-of-the-art functions against the MF generalizations .... 162

B.2. Evaluation of the weights effectiveness on PL2MF and BM25MF .............. 162


B.4. Queries of complexity 2 ................................................................. 165

B.5. Queries of complexity 3 ................................................................. 166

B.6. Queries of complexity 4, 5, 6, 9, and 10 .............................................. 167

B.7. Average MAP values over simple and complex graphs ............................ 168
<table>
<thead>
<tr>
<th>Section</th>
<th>Definition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1.</td>
<td>Definition (Graph)</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2.</td>
<td>Definition (Type Attributes Set $T$)</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3.</td>
<td>Definition (Entity Description)</td>
<td>18</td>
</tr>
<tr>
<td>2.3.4.</td>
<td>Definition (Dataset)</td>
<td>19</td>
</tr>
<tr>
<td>4.1.1.</td>
<td>Definition (Graph Homomorphism)</td>
<td>36</td>
</tr>
<tr>
<td>4.1.2.</td>
<td>Definition (Undefined Sumnode $\Omega$)</td>
<td>38</td>
</tr>
<tr>
<td>4.1.3.</td>
<td>Definition (Sink Sumnode $\emptyset$)</td>
<td>38</td>
</tr>
<tr>
<td>4.1.4.</td>
<td>Definition (Graph Summary)</td>
<td>39</td>
</tr>
<tr>
<td>4.1.5.</td>
<td>Definition (Summary Path Instance)</td>
<td>40</td>
</tr>
<tr>
<td>4.1.6.</td>
<td>Definition (Precise Graph Summary)</td>
<td>41</td>
</tr>
<tr>
<td>4.1.7.</td>
<td>Definition (Bisimulation)</td>
<td>41</td>
</tr>
<tr>
<td>4.1.8.</td>
<td>Definition (FBT-Bisimulation)</td>
<td>42</td>
</tr>
<tr>
<td>4.1.9.</td>
<td>Definition (FT-Bisimulation)</td>
<td>43</td>
</tr>
<tr>
<td>4.1.10.</td>
<td>Definition (Bisimulation Summary)</td>
<td>44</td>
</tr>
<tr>
<td>4.1.11.</td>
<td>Definition (FT-Bisimulation Summary)</td>
<td>45</td>
</tr>
<tr>
<td>4.1.12.</td>
<td>Definition (Unique Type Summary)</td>
<td>47</td>
</tr>
<tr>
<td>4.1.13.</td>
<td>Definition (Types Summary)</td>
<td>48</td>
</tr>
<tr>
<td>4.1.14.</td>
<td>Definition (Attributes Summary)</td>
<td>48</td>
</tr>
<tr>
<td>4.1.15.</td>
<td>Definition (Attributes &amp; Types Summary)</td>
<td>49</td>
</tr>
<tr>
<td>4.1.16.</td>
<td>Definition (IO Attributes Summary)</td>
<td>49</td>
</tr>
<tr>
<td>4.1.17.</td>
<td>Definition (Incoming Attributes Summary)</td>
<td>50</td>
</tr>
<tr>
<td>4.1.18.</td>
<td>Definition (Incoming Attributes &amp; Types Summary)</td>
<td>50</td>
</tr>
<tr>
<td>4.1.19.</td>
<td>Definition (IO Attributes &amp; Types Summary)</td>
<td>51</td>
</tr>
<tr>
<td>4.2.1.</td>
<td>Definition (Edge Provenance)</td>
<td>57</td>
</tr>
<tr>
<td>4.2.2.</td>
<td>Definition (Node Ownership)</td>
<td>57</td>
</tr>
<tr>
<td>4.2.3.</td>
<td>Definition (Edge Authority)</td>
<td>57</td>
</tr>
<tr>
<td>5.1.1.</td>
<td>Definition (Relation $\sqsubseteq$)</td>
<td>70</td>
</tr>
<tr>
<td>5.2.1.</td>
<td>Definition (Summary Error $\text{Err}(R)$)</td>
<td>73</td>
</tr>
<tr>
<td>5.2.2.</td>
<td>Definition (Inferred Graph)</td>
<td>73</td>
</tr>
<tr>
<td>5.2.3.</td>
<td>Definition (Connectivity Error)</td>
<td>75</td>
</tr>
<tr>
<td>5.2.4.</td>
<td>Definition (Attribute Error)</td>
<td>75</td>
</tr>
<tr>
<td>5.2.5.</td>
<td>Definition (Type Error)</td>
<td>76</td>
</tr>
<tr>
<td>5.2.6.</td>
<td>Definition (True Positive Sumedges)</td>
<td>76</td>
</tr>
<tr>
<td>5.2.7.</td>
<td>Definition (False Positive Sumedges)</td>
<td>77</td>
</tr>
<tr>
<td>Section</td>
<td>Definition Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2.8.</td>
<td>Definition (Sumedge Precision)</td>
<td>77</td>
</tr>
<tr>
<td>6.2.1.</td>
<td>Definition (Rooted Entity)</td>
<td>93</td>
</tr>
<tr>
<td>6.7.1.</td>
<td>Definition (Query Extraction Function)</td>
<td>114</td>
</tr>
<tr>
<td>6.7.2.</td>
<td>Definition (Triple Pattern)</td>
<td>115</td>
</tr>
<tr>
<td>7.2.1.</td>
<td>Definition (Summary Pattern)</td>
<td>135</td>
</tr>
<tr>
<td>7.2.2.</td>
<td>Definition (Class Triple Pattern)</td>
<td>135</td>
</tr>
</tbody>
</table>
Part I.

Background
Chapter 1.

Introduction

Since the advent of the Internet, people and companies exchange information on the Web. There is now an abundance of information about a wide range of domains such as science, commerce, leisure, etc. Although the Web browsed by Humans mainly contains unstructured data such as text, a significant amount of structured data exists.

- hyperlinks from a web page to another in order to point to additional or related information;
- structural information encoded into the HTML pages using table or list tags for instance;
- metadata encoded into web pages that provides some meaning about their content.

The Semantic Web vision is to enrich this data with machine-readable semantic information, transforming the Web into a global database that we call the Web of Data [BHBL09]. For example, the description of a table within a web page can be improved by using a semantic markup, e.g., using the RDFa\(^1\) markup language, so that the columns of the table have a sense: for instance is it concerned with time data, people’s name, geographic places, etc.

The Semantic Web movement amplified the growth of, now semantically enriched, information on the Web. The Linking Open Data\(^2\) (LOD) project is a successful example, which provides a collection of inter-linked datasets about diverse domains.

A core point of the Semantic Web is the use of “Uniform Resource Identifiers” (URIs) for denoting things, e.g., objects, people, places, . . . . This allows to uniquely name something and describe it, so that it may be reused in other places. Once the data is described using URIs, and also that existing URIs are reused, we reach a state where the data published on the Web is inter-connected. The data thus forms a “graph”, with nodes representing the described objects, and edges detailing the relationship between two objects (nodes).

The inter-connected data is at the “scale of the Web”, consisting of thousands, if not millions, of different pieces of information about some topics. This scale opens up interesting ways of searching, browsing, and exploring the data.

\(^1\)RDFa: http://www.w3.org/TR/xhtml-rdfa-primer/
\(^2\)http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
Chapter 1. Introduction

In order to account for the variety of data about a subject, a “Data Mashup” technique is needed as a mean for reading through all different things being said about that subject. Also, within the context of a data mashup, one may be more interested in aggregated measures rather than specific details.

Data exploration ultimately requires some knowledge about the structure of the (graph) data. There is a need to know what attributes a piece of information contains along with their semantics, so to enable one to search for relevant data. Such knowledge about the structure requires techniques for managing the data. In Section 1.2, we present two data management approaches that are suited to different kinds of data.

The data exploration endeavour is challenged by the scale, heterogeneity, and quality of the data itself. The Web of Data is composed of at least a thousand\(^3\) of datasets, spanning over a variety of topics. Web Data presents a structure that changes from a dataset to the other, evolving over time. The amount of information about a topic varies from data sources, some being more complete in one aspect and less in another. To overcome these challenges, there is a need for ranking the data so that relevant results to an information need are accessed first. This calls for a novel ranking mechanism that leverage the rich structured information of the Web Data; we ponder this in Section 1.3.

Throughout this thesis, we present methodologies designed at shedding light on the structure of Web Data. We propose methods that take into account the shifting nature of the Web Data. We develop then an approach which highlights the “skeleton” (structure) of datasets. We leverage the rich structure of the Web of Data in a novel ranking model for (semi-)structured data. We finally present how these techniques can be combined for a more effective exploration of datasets.

In this thesis, we use Web of Data to refer to the infrastructure that makes possible the inter-connected (semi-)structured data; and Web Data to refer to that (semi-)structured data [DCT12].

1.1. Motivation

A user with an information need has at his disposition several ways to find relevant data. A search engine may be used so that documents embedding relevant, structured data, can be retrieved: for example, thanks to services such as Swoogle [DFJ+04] or Sindice [ODC+08]. Such engines provide search access using Information Retrieval indices, on which boolean queries are evaluated.

However, it is better to rely on graph query languages such as the standardised SPARQL\(^4\) language, if the user’s quest requires more expressivity than what is possible with boolean queries. In such a case, a direct access to the data is needed. Platforms

---

\(^3\)Datahub portal: http://datahub.io/dataset?tags=lod
\(^4\)SPARQL: http://www.w3.org/TR/sparql11-overview/
1.1. Motivation

such as DataHub\(^5\) or Publicdata\(^6\) present available data accesses, e.g., download of data dumps or a SPARQL endpoint. Once a user has found a possibly relevant dataset, he has the actual daunting task of querying that dataset: not because of the language itself, but due to the lack of information about the dataset’s structure.

As an illustration, we consider the Eunis\(^7\) dataset which contains information about species and their habitats.

**Query formulation.** Although the Eunis dataset can be considered as a small dataset, a user is nonetheless challenged in effectively and efficiently querying the dataset. In order to formulate a query, predicates and/or types occurring in the dataset are needed. However, such information is in general not available. Here, a VoID [ACHZ09] description is present, but that specific file does not provide sufficient information beyond what is needed for simple queries.

A simple way to retrieve such information is to query the dataset directly. However, we then have to choose among 500 types and at least 1500 predicates. The Eunis dataset exhibits as well the use of over 40 different vocabularies. A possible reason of such heterogeneity can be that the Eunis dataset is the end result of several integrated data sources. Also, as a consequence of this heterogeneity, a single ontology cannot help in learning about the dataset’s structure.

As the query construction progresses, the user needs to know which predicates/types can be used in conjunction with the already selected ones. At this point, a user is faced with the issue of data inconsistency: the resources in a datasets do not all share the same set of predicates/types. Thus, the query must be carefully built so to retrieve actual results. The complexity of writing the query increases since the user needs to account for missing predicates (or types). *How can the user learn about the data structure so to formulate intelligent queries?*

**Ranking.** With the Eunis dataset having at least 500 types and 1500 predicates, the user is left with the challenge of deciding which one to use. Are the ones most used the more relevant for the user? Does the predicate label, e.g., http://xmlns.com/foaf/0.1/name, give some indication about the semantic of that predicate? One may dereference it and read its description, but this demands yet additional work for the user. *How can a user decide which type/predicate to use?*

---

\(^5\)DataHub: [http://datahub.io/](http://datahub.io/)

\(^6\)PublicData.eu: [https://publicdata.eu/](https://publicdata.eu/)

\(^7\)Eunis on DataHub: [http://datahub.io/dataset/eunis](http://datahub.io/dataset/eunis)


1.2. Data Management

Web Data is composed of a variety of data about many entities, for example which can be a person, a country, an event, etc. The decentralised nature of the Web has the effect that information about a same entity is available on several web pages. Data sources can be complementary in some aspects, and redundant in others. Furthermore, each source has its own take on data modelling. Sources may label an attribute of an entity differently although what is described is the same. Sources may also represent an attribute differently, e.g., the name of a person as two properties with first and last name, versus a single property which is a concatenation of the two. These facts highlight the heterogeneity of the data on the Web, which stands as a challenge when managing data.

In order to fulfill an information need, a user generally uses a web search engine. This allows him to browse only the most relevant pages with regards to his need out of the billions of web pages that the Web consists of. To achieve this, search engines rely on the structure of the data so to rank documents appropriately. The knowledge of the structure of the data, which can sometimes be seen as a schema, is a necessary medium for exploration and, ultimately, finding the answer to one’s need.

To ease this process, many vocabularies have been built over the years for representing one’s data; for example the following:

- **FOAF** defines terms for describing a person and the relationships between people;
- **SKOS** is used for representing knowledge organisation systems such as taxonomies; or
- **GEO** represents geospatial coordinates.

However, vocabularies tend to be loosely followed in datasets — this is reflected by the disparate use of vocabulary terms and ontologies across datasets in [CCP+11]. One of the reasons is that many different design choices are possible while modelling one’s data. The structure of the data and use of vocabularies evolve over time as new requirements come: new terms are added, some are deleted, and the semantic of some may be modified. Furthermore, changes in the vocabulary upstream may not be reflected in the datasets in which it is used. As a consequence, we refer to Web Data as being semi-structured [Abi97].

The vocabulary heterogeneity, combined with the scale of datasets, actually complicates the effective use of the data. This calls for novel techniques designed at “understanding” the data while taking into account its structural richness.

---

13. [http://www.w3.org/2003/01/geo/wgs84_pos](http://www.w3.org/2003/01/geo/wgs84_pos)
1.2.1. Schema-based Data Management

In the relational database community, a *logical schema* provides an overview of a database. The schema describes the content of the database, how it is structured and what attributes it has. In the Web of Data, such a description is generally not available, thus preventing an effective exploration of the data.

Web Data is in general not bundled with a schema, or a description akin to a schema. This is due to the very nature of Web Data: it is dynamic, diverse, and ever-evolving. The growth of Web Data is in essence similar to that of the Web: it evolves organically, constrained by nothing specifically. In such an environment, a schema-based data management is extremely difficult to put into place, if not impossible.

1.2.2. Decentralised Data Management

Web Data can be created without the need to follow specific rules. The data does not need to adhere to some kind of pre-defined schema. Inter-linking with others’ data is promoted in the Web of Data by the use of URIs. The distributed aspect of the data is emphasised by Halevy et al. [HIST03] where a peer-to-peer data management system is proposed. However, data sources are assumed to have a schema, and the existence of a global schema that *mediates* the sharing of information between sources is needed.

In the Web of Data, the existence of a central place where the schema of data sources is kept is unreasonable due to the scale and dynamicity of the data. Nonetheless, a schema enables two benefits (1) it ensures that new data adheres to that schema; and (2) it provides an overview over existing data. Given that the first point cannot be achieved in the Web of Data context, we propose the graph summary as a way to provide the second point: it is a novel system for managing distributed, heterogeneous data sources on the Web.

1.3. Ranking of Web Data

With the growth of the Web in the early days of The Internet, the use of web search engines to sift through the mass of documents marked a significant milestone in the interaction of people with regards to data. Then, the ability of search engines not only to retrieve but also to *rank* the data by relevance to a query marked another milestone: the possibility to find a needle in the haystack.

Traditionally, the documents ranked contain mostly textual information, with little structure that can be leveraged by a ranking algorithm. Web documents contain HTML markup for display purposes, but those can be cues for the importance of a piece of text within the document. Web Data instead is structured beyond what can be extrapolated from the markup cues. Therefore, this calls for a novel ranking mechanism that supports this rich structure.
Chapter 1. Introduction

1.3.1. Traditional Ranking on The Web

Documents on the Web are mainly textual, containing information only for Humans to process. In order to rank such documents, a straight-forward approach is to consider a document as a set of words and using word frequencies to estimate its importance within the document. Thus, with such a modelling a document is referred to as a \textit{bag of words}. The ranking of a document can be improved by, for example, adding some structural information into the algorithm. Some specific parts can be extracted so to apply a weight, for instance the title. Such approaches are referred to as \textit{Field-based} ranking. However, the structure considered in those are too simple and so they cannot leverage the complex (graph) structure that is exhibited in Web Data. This marks a shift from the ranking of (Field-based) documents to graph data.

1.3.2. Ranking of Graph Data

In the Web of Data, we use URIs to name things, e.g., people, monuments, places, . . ., which allows one to lookup an URI and learn about it, but also for others to reference that URI in their own dataset. The Web Data takes then the form of a graph, where a node represent a concept and the edge a labelled relationship that links two nodes. While Web Data follows the standard RDF format that opens many opportunities to tackle graph ranking, we consider in this thesis to improve the ranking model.

In Web Data, the structure of the graph carries some information about the data. However, most of it is discarded by Field-based ranking approaches due to the bag of words modelling. The Figure 1.1a depicts a graph about “Ireland” and its neighbouring country; the Figure 1.1b is a depiction of its bag of words. We observe that with the bag of words modelling, we lose some semantic that is carried by the structure: the population number is not closely associated with the “United Kingdom” anymore.

The ranking of a graph requires a novel way of modelling the data that does not discard the structure. Hence, we propose the “MF” ranking model that leverage the graph structure in order to better compute the relevance of a graph with regards to a query.

1.4. Challenges

The purpose of the summary is to describe the structure of a graph. Therefore, it can hold the function of a schema for that graph. A schema is the basis of many algorithms over the data. A schema describes the content thus facilitating users to search for a particular resource. From the description a schema provides, one may draw out statistical information about the data that can be useful in some scenarios, e.g., query optimisation. Schemas help in the integration process of several data sources. As a consequence, a major challenge is to determine what features of the data can be used in order to effectively generate a schema. We study this aspect in Chapter 4.
1.5. Scope of the Research

The construction of a graph summary does not ensure that the entity graph can be re-built from its summary. This means that some details about the structure of the graph was discarded during the summarisation process. Although some errors in the summary can be acceptable for a user exploring the data, it can cause a significant drop of performance for a query optimizer. Hence, it is necessary to measure how accurate a summary is with regards to the data. We introduce a model for measuring the precision of a summary in Chapter 5.

In the recent years, the amount of data has been growing increasingly. A glimpse of this growth over the years can be visually appreciated through the many diagrams of the Linking Open Data cloud. It is common to see data with millions of entities and billions of statements, to which the “Open Data” movement contributed a significant part. However, the ease of sharing data has the downside of that data not following a strict schema, if any.

As a consequence, there is a need for techniques that can provide a structure akin to a schema, e.g., a summary, without restraining the “evolution” of the data, while at the same time that can scale to large volumes of data.

1.5. Scope of the Research

The scope of this thesis is to study the creation of summaries on large scale semi-structured information sources. By semi-structured, we mean data that presents some structure but that is not followed strictly by all records in the dataset. By large, scale we mean data which contains millions of entities with billions of statements describing said entities.

The main challenge is to present a summary creation process that is able to cope with

---

14 The Linking Open Data cloud diagram: [http://lod-cloud.net/](http://lod-cloud.net/)
Chapter 1. Introduction

the scale of the data, as well as with the variability of the data due to the heterogeneous and loose use of vocabularies. To this end, we investigate graph summarisation approaches which generate summaries via graph homomorphism. Due to the heterogeneity of the data, the generation of approximate summaries is the only viable option for a large number of use cases.

Different applications have varying interactivity with respect to the summary. We investigate ranking techniques for retrieving first the parts of the summary an application need. We evaluate how ranking can be used for improving the exploration of data.

1.6. Applications

Graph summarisation produces a summary of the structure of the graph, providing information akin to a schema in relational databases. We can then leverage the summary in several ways, of which some are presented in this thesis.

*Graph schema.* We present in Section 7.1 that the graph summary can serve the function of a schema for the graph. The generated summary of a graph highlights its structure, unveiling the underlying schema; ultimately this allows one to query for the desired information.

*Querying assistance.* In Section 7.2 we use the structural information from a summary in order to assist a user in writing a query over the graph. This allows users having little to no knowledge about the data to build meaningful queries.

*Browsing of a dataset’s structure.* The information gathered during the graph summarisation process of a dataset is made available for browsing in Section 7.3. This enables a user to learn about the internals of a dataset: its structure, the kind of information it carries, the relationships between concepts, or its connections to other datasets. With such knowledge, the user may identify problems with the data, e.g., missing attributes.

1.7. Scientific Methodology

We deployed several scientific methods from different fields from Computer Science, especially theoretical Computer Science and experimental Computer Science.

Specifically, we investigated several graph summaries useful for different aspects. We then generalised these to define a general formal framework, described in Chapter 4, to cover approximate graph summaries as well. We determined the complexity of the resulting algorithms and evaluated their behaviour in experimental settings in Chapter 5.
We also experimented with the resulting graphs to investigate the behaviour in a practical setting within the Sindice [DCT12] project and evaluated the results.

The Sindice [DCT12] project aimed to create an Information Retrieval search engine for Web Data. That engine had already collected a significant amount of data, but then with this came the challenge of writing meaningful queries. To help with this need, a good ranking is needed which was investigated in [DTC+10] and continued with the MF ranking model (Chapter 6). Difficulty to understand what was inside the collected data, but also the challenge with writing queries motivated the research into graph summarisation (Chapter 4).

1.8. Outline

The thesis is divided in three parts. This first part Background contains the introduction to the thesis, and presents fundamental concepts and models that will be reused throughout the rest of the thesis. We introduce in the second part Methods new techniques for generating and evaluating graph summaries, as well as a new ranking model for semi-structured data. In the final part Putting the Pieces Together, we describe applications of the presented methods.

Part II: Foundations
Chapter 2: We present here a graph data model that encompasses the wide variety of formats used for modelling structured data. The model is layered so to represent the concepts of entity and dataset.

Part III: Methods
Chapter 3: We present in this chapter several approaches at summarising graph data and for what scenario they were built for. We emphasize their shortcomings and how these are answered thanks to the graph summary.

Chapter 4: In this chapter, we introduce the formal model of a graph summary. Then, we present a technique for generating graph summaries for Web Data at scale.

Chapter 5: The summary of a graph is built in a bottom-up fashion, through the aggregation of nodes of the entity graph. The graph summary being a graph homomorphism, it is not assured that entity graph can be re-built from its summary. Therefore, some details about the structure of the graph are missing from the graph summary. In order to account for this loss, we developed a model for measuring the precision of a graph summary.

Chapter 6: In this chapter, we introduce the “MF” ranking model for semi-structured data. The model considers the structural heterogeneity of the data for a better scoring of entities.
Chapter 1. Introduction

Part IV: Putting the Pieces Together
Chapter 7: In this chapter, we present applications of the introduced methods. We develop a system based on a generated graph summary that assists users in writing graph-shaped queries without a priori knowledge of the data structure. We present an RDF model of the graph summary, which allows the summary to serve the function of a schema. In addition, we present another application of the graph summary that enables one to inspect a dataset so that problems with the data may be identified.

1.9. Contributions

Web Data is a large collection of (semi-)structured data about a variety of domains. The data comes from a wide panel of sources, ranging from database dumps to publicly-edited content. In order to use this information effectively, it is necessary to know how the data is structured, which is essential in cases such as data integration or querying for example. In the context of Web Data, the availability of such a description of the data structure is extremely rare.

The main contribution of this work is we believe a solution to this basic challenge when interacting with Web Data: a method for extracting the structure of large and distributed semi-structured datasets through graph summarisation. Since this method needs to scale to large datasets, we investigate extracting an approximation of the data structure. As a consequence, we also investigate the use of a ranking data model in order to reduce the loss of precision due to the approximation.

Specific contributions include:

- a formal model for an entity graph in Chapter 2;
- a review of literature regarding the summarisation of graphs in Chapter 3;
- a method for generating graph summaries of large and distributed semi-structured data in Chapter 4, based on the work in [CPC+12, CDT13];
- a model for measuring the precision of a graph summary in Chapter 5, based on the work in [CDT13];
- a novel ranking model specifically designed for dealing with the structural and content heterogeneity of Web Data in Chapter 6, based on the work in [CDR+11, CDT12]; and
- a set of tools for improving the exploitation of semi-structured data in Chapter 7, based on the work in [ABT14].
Chapter 2.

Semi-Structured Data

Despite the fact that the Web is best known as a large collection of textual documents, it also provides an increasing amount of structured data sources in a wide range of formats, from HTML tables to Deep Web databases, XML documents, documents embedding semantic markups, e.g., Microformats, Microdata, RDF, RDFa. Structured data on the Web covers a large range of domains, e.g., e-commerce, e-government, social network, science, editorial world, ..., and can describe any kind of entities, e.g., people, organisations, products, locations, etc. We face the challenge of handling the heterogeneity of the data representation.

In some areas, information is commonly represented with graphs, since it is expressive enough for modelling data with complex relationships. Graphs are used in social science for describing interactions between people. Graphs such as semantic networks [Sow06] have been used in artificial intelligence and machine translation, with the Semantic Web being a large scale instance.

Therefore, we base this thesis work on a graph data model in order to account for the variety of data.

2.1. Semi-Structured Data Models

Information on the Web is available in variety of formats and design, e.g., from HTML tables to documents embedding semantic markups. Each data source has its own schema, ranging from loosely to strictly defined. The data evolves over time, e.g., new attributes are required or new entities have been integrated from an external source. As such, data that does not follow a strict schema is called semi-structured [Abi97].

OEM –- Object Exchange Model

A seminal work in modelling semi-structured is the OEM model [PGMW95]. OEM served as the basic data model in projects such as Tsimmis [CGMH+94], Lore [QWG+96] and Lorel [AQM+97]. In the Object Exchange Model, the data is a collection of objects, where each object can be either atomic or complex. An atomic object represents some base data types, while a complex object is a (attribute, object) pair. OEM is a graph-based data model, where the nodes are the objects and the edges are labelled with attribute names.
Chapter 2. Semi-Structured Data

XML -- Extensible Markup Language

XML [BPSM+98] is a W3C standard for exchanging data on the Web. Nested, tagged elements are the building blocks of XML. A tagged element can have attribute value pairs and sub-elements. A sub-element may be either a tagged element or an atomic value such as text. XML is well suited for semi-structured data since there is no restriction regarding the naming of tags or the relationships between tags.

RDF -- Resource Description Format

RDF [KC04a] is a generic data model proposed for exchanging data on the Web. A resource description in RDF is composed of statements about the resource. A statement asserts that a resource has a value associated with a property. It is modelled as a triple consisting of a subject, a predicate, and an object. A set of RDF statements forms a directed labelled graph. While it is similar to the OEM model, RDF proposes some additional features such as the use of URIs as global identifiers which helps to avoid ambiguities and enables the creation of a global data graph.

Any kind of data can generally be modelled using a graph, since it is versatile enough to accommodate different specifications. As such, we build the work in this thesis upon a generic graph model.

2.2. Abstract Model

We consider the data as being a labelled directed graph and build methods based on this model. Therefore, the findings presented in this work can be applied to any domain where a graph is used as data representation.

We describe in this Section a graph data model based on the work of [DCT12]. The graph is represented with two layers that depict the data at two extremes: the dataset and the entity layers. We formally introduce the concepts of entity and dataset in Sections 2.3.2 and 2.3.3, respectively. The Figure 2.1 depicts the layers of the model, where the dataset layer is depicted at the top, and the entity layer at the bottom.

Dataset Layer: A dataset represents a collection of entities; the dataset layer is the ensemble of entities’ collections. This layer provides coarse information about the data: the focus of this layer is the relationships between datasets. In the figure, the dataset \( D_1 \) connects to the dataset \( D_2 \) via the link \( p_1 \), and \( D_2 \) is connected to \( D_1 \) via two links, i.e., \( p_2 \) and \( p_3 \).

Entity Layer: The entity layer contains information about entities. Of the two layers, this one provides the finer details about the data: attributes about an entity, and also relationships between entities. In the figure, the entity layer provides additional information about the dataset layer, showing which entities are actually connected within — and between — datasets.

The dataset layer shows the links between datasets, i.e., inter-dataset links, while the entity layer provides in addition information on the links within a dataset, i.e., intra-dataset links.
The dataset layer is blind to the information the entity layer provides: its focus is on the relationships between datasets, and not between the entities that compose the datasets.

Regardless of the layer, we represent the information using a node and edge labelled directed graph, referred simply as a graph from this point on. Also, we will call the graph on the dataset layer the dataset graph, and the graph on the entity layer the entity graph.

### 2.3. Formal Model

In the following we formally define a directed labelled graph that we call simply a graph henceforth, and introduce concepts used in the rest of the thesis. We use a graph as a generic representation of (semi-)structured and unstructured data on the Web of Data. A graph is defined as a tuple consisting of a set of nodes $V$ and a set $A$ of directed edges labelled $a$. We consider edge and node labels to come from a set that we denote with $\mathcal{L}$. Figure 2.2 depicts a graph of people and places along with their connections.

**Definition 2.3.1 (Graph)**

Let $V$ be a set of nodes and $\mathcal{L}$ a set of labels. The set of directed labelled edges is defined as $A \subseteq \{(x, \alpha, y) \mid (x, y) \in V^2, \alpha \in \mathcal{L}\}$.

A graph $G$ is a tuple $G = (V, A, l_V)$ where $l_V : V \mapsto \mathcal{L}$ is a node-labelling function.
Chapter 2. Semi-Structured Data

Given a graph $G = \langle V, A, l \rangle$, we introduce below terms used throughout the thesis.

**Edge** An edge labelled $\alpha$ from the node $x$ to the node $y$ is written as $(x, \alpha, y)$. We say that $x$ is the *source*, and $y$ is the *target*.

**Path** We call a *path* a sequence of edges label $\alpha_1, \cdots, \alpha_n$. A path exists in the graph $G$ if there is a sequence of nodes $\{v_1, \cdots, v_{n+1}\} \subseteq V$ connected with such labelled edges.

In Figure 2.2 the path *author.lives* exists in that graph as it connects the nodes $\{v_0, v_2, v_3\}$.

**Attribute** We call *Attribute* the label of an edge.

We define as $\textit{Attributes} (x \in V) = \{\alpha \in L | \exists y \in V, (x, \alpha, y) \in A\}$ the set of attributes associated as a source with a node $x$ in $G$. In the figure, we have $\textit{attributes}(v_1) = \{\text{lives}, \text{works}, \text{name}, \tau\}$.

Similarly, we define the set of *Incoming attributes* as

$$\textit{attributes}^{-1}(x \in V) = \{\alpha \in L | \exists w \in V, (w, \alpha, x) \in A\}$$

**Value** We call *value* the label of a node which is the target of an edge.

**$k$-hop** We say that two nodes are $k$-hop away if there is a path composed of $k$ edges label connecting the two. For example in the figure, the nodes $v_1$ and $v_6$ are 2-hops away because of the path *lives, location*.

**Size** The size of a graph is defined as the number of edges $|A|$ that the graph is composed of.

### 2.3.1. Type Attribute

Attributes and node labels can be used in a graph for describing the content. For example in Figure 2.2, we can expect that the value of the edge labelled “name” represents a textual identification of the node $v_1$. In addition, a set of nodes may represent a same concept, e.g., the nodes $v_1$ and $v_2$ represent a person. We call, e.g., “person”, the *type* of a node, and that node is an *instance* of that type.

We model this in a graph thanks to an edge where (1) the source is an instance; (2) the target is the type; and (3) the attribute is a certain pre-defined label that we denote as $\tau$. That certain attribute is used to indicate that the target of the edge represents a type; in RDF, it is commonly the *rdf:type*\(^1\) attribute.

Since there may be several attributes that can hold this function, we refer to them as the *type attributes*. For example in Figure 2.2, the edge $(v_1, \tau, \text{Person})$ indicates that the node $v_1$ is an instance of type *Person*. We define $T$ as the set of *type attributes*, and we denote an element of that set with $\tau$.

\(^1\)RDF type attribute: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Figure 2.2.: An entity graph describing people, places, and their relationships.
Chapter 2. Semi-Structured Data

(a) 1-hop entity description

(b) 2-hops entity description

Figure 2.3.: Interpretation of the entity description dependent on the graph structure.

**Definition 2.3.2 (Type Attributes Set $T$)**

Let $T \subset \mathcal{L}$ be a subset of labels. We define $T$ as the set of type attributes, and we call $\tau \in T$ a type attribute.

For a given node $x \in V$, we consider $\text{Types}(x)$ as the set of values for which the attribute is a type attribute.

$$
types(x \in V) = \{l_V(c) \mid \exists c \in V \exists \tau \in T : (x, \tau, c) \in A\}
$$

For instance, we have $\text{types}(v_6) = \{\text{Country}, \text{Place}\}$ in Figure 2.2 as the set of type values for the node $v_6$.

### 2.3.2. Entity description

The Web Data represents a graph of interconnected entities, where an entity may be a person, a place, or an organisation, .... We call an entity description the pieces of informations in the graph that are about that entity. For instance, we can consider “Galway” to be part of the entity description of $v_3$ in Figure 2.2, but not “Rome”.

There exists several ways to define which parts of the graph describe an entity. One may consider all the edges, incoming and outgoing, to form the entity description. For some graphs, edges that are two-hops away might be part of it too.

We illustrate this in Figure 2.3 where we might consider the entity description to consist of edges 1-hop away from the node given the graph in Figure 2.3a, although it is more adapted to extend it to 2-hops with the graph in Figure 2.3b. The reason for the design in the latter graph might be to add more information about a person’s label, e.g., the honorific.

**Definition 2.3.3 (Entity Description)**

Let $G = \langle V, A, l_V \rangle$ be a graph. We define the entity description of a node $u \in V$ as
2.3. Formal Model

the set of edges $E(u) \subseteq A$ that describe that node.

For example, if we consider the outgoing edges to form the entity description, we have for the node $v_6$ the following set:

$$E(v_6) = \{(v_6, \text{label, Ireland}), (v_6, \tau, \text{Country}), (v_6, \tau, \text{Place})\}$$

2.3.3. Dataset

In the Web of Data, a dataset [ACHZ09] represents a collection of edges which, e.g., are located in a same website, or share a common topic. A dataset can be a dump in RDF, or the embedded metadata of pages in a website.

Definition 2.3.4 (Dataset)

Let $G_1, \cdots, G_i, \cdots, G_n$ be $n$ graphs such that $G_i = (V_i, A_i, l_{V_i})$ is a graph. Let $l_G : G \mapsto \mathcal{L}$ be a function that assigns a label to a graph. We define a dataset as the tuple $\langle D, \alpha \rangle$, where $D = \{G_1, \cdots, G_n\}$ and $\alpha \in \mathcal{L}$ is the dataset label such that $\forall 0 < i \leq n l_G(G_i) = \alpha$.

We extend without loss of generality the definitions presented in this chapter to a dataset, which we indicate by adding a subscript $d$. We shall omit this subscript if it is clear from the context that only one dataset is considered. Therefore, we have the following definitions:

- $G_d = (V_i, A_i, l_{V_i})$: The label of the graph $G$ is $d$, i.e., $l_G(G) = d$.
- $E_d$: The edges from the entity description belong to a graph which dataset label is $d$.
Part II.

Methods
Chapter 3.

Graph Summarisation: Related Work

In many Computer Science areas, the information is represented as and is analysed through graphs. Graphs are used to capture the social relationship between people, to represent processes and their transition in concurrency, or to structure data in a flexible way, e.g., using the Object Exchange Model [PGMW95] or, more recently, the Resource Description Framework [KC04b].

A common challenge encountered when analysing a graph is its size. A large number of nodes or edges increases the running time of algorithms applied on the data. A solution is to map the graph into another, smaller graph while preserving its structure. That process is a graph homomorphism, which is commonly called graph summarisation and the resulting graph is then considered as a summary of the entity graph.

In addition to the challenge of the graph size, we consider also the other challenge of the graph structure being partially or totally unknown. As discussed in Section 1.2, although a dataset might adhere to an ontology, in general it cannot be used to understand the structure of that dataset, a possible reason being either (a) the dataset is a mashup of several data sources, each following a different ontology — therefore, there is no ontology that describes the combined dataset; or (b) the dataset is originally built from a variety of ontologies in order to fulfill modelling requirements. Therefore, a graph summarisation can then be used for shedding light on the structure of the graph, which can hold a function similar to a schema in databases.

Depending on the kind of interaction with the summary, a different level of details is needed. Intuitively, it is conceivable that a graphic application may require less information about the data compared to a query optimisation application. We present in this chapter the scientific literature on graph summarisation. The research works are ordered by the increasing level of details needed in the summary by the respective use case applications.

3.1. Dimensions of Graph Summarisation Challenges

In the computer science literature, we see the graph summarisation technique being employed to face a performance challenge. Some graphs are too large to be processed in a reasonable time or to fit into the main memory. A solution is to substitute that graph
with another that is significantly smaller, thus improving the application’s performance. Questions that can be raised are (a) in which aspect is the summary smaller than the graph; and (b) what properties of the graph influence the summary. In this section, we present three dimensions through which we discuss the literature on graph summarisation in the rest of this chapter:

1. the size of the graph;
2. the size of its summary; and
3. the impact of the graph’s heterogeneity on the summary.

The Figure 3.1 depicts the dimensions which form a three-dimensional space, within which a graph summarisation approach is located. For example, an approach might produce a summary which size increases as the graph gets more heterogeneous.

**Size of the graph.** It is desirable for a summary to be smaller than the entity graph, since it is generally its substitute for performance reason. We measure the “complexity” of a graph by its number of edges. The rationale is that the more a graph has edges, the more difficult it is to process since it requires more computational power and memory.

**Graph heterogeneity.** We consider a graph to be heterogeneous when it is not consistent with regards to (a) its structure; and (b) its labelling. Our rationale is that the more heterogeneous an entity graph is, the more difficult it is to create a precise and concise summary of that graph. By precise, we mean a summary which description of the entity structure of the graph is accurate. By concise, we mean a summary of small size.

For example, we can imagine a dataset describing people in which the attributes are not consistent from a person to another: in one the phone number is missing, while in another it is the birth place. The Figure 3.2 illustrates the two scenari of graph heterogeneity, where we depict possibles graph summarisation with a dashed graph.

**Heterogeneity of the graph structure.** Figure 3.2a depicts a graph in which the structure is inconsistent. The node $a_0$ has two incoming edges and a single outgoing edge, while for the node $b_0$ it is the opposite: a single incoming edge and two outgoing. Although the depicted summary in dashed lines may be considered sufficient, it loses some
3.2. Graph Summarisation Approaches

The concept of graph summarisation has been explored for diverse use cases in the literature. In Section 3.3, we present some research where it has been employed for aiding a user in exploring an unknown graph. In Section 3.4, a summary of the graph is used for profiling the data, i.e., gathering some statistics about the graph so to analyse it with regards to upstream applications such as data integration. Graph summarisation is also used for analysing the content of the graph by computing aggregates, which research is discussed in Section 3.5. As discussed in Section 3.6, graph summarisation has been used for the purpose of optimising the query evaluation by outlining what paths exist in the graph and creating indices over the summary.
Chapter 3. Graph Summarisation: Related Work

Figure 3.3.: Time-line of the research on graph summarisation

Figure 3.3 depicts a time-line of the literature presented in this chapter. The main use case of each work is indicated with a trapeze having its own colour and stripes; a trapeze reflects the sections outlined in the previous paragraph. From the time-line, we observe that graph summarisation was at first used for optimising query evaluation through indices; a new direction later appeared where the summary is used for analysing and exploring a graph. This new direction might be linked with the emergence of Web Data, since users generally interact with graphs which structure is (partially) unknown.

Graph clustering versus graph summarisation. Both techniques are similar in the sense that they partition the graph. Clustering is the process of finding the underlying structure of a dataset by grouping elements that are similar to each other. A group is then called a cluster. In the context of a graph, a clustering technique in general seeks clusters of nodes that have many connections to each other within, but few between the clusters. Schaeffer presents in [Sch07] a survey of graph clustering approaches. Instead, with graph summarisation techniques we aim to partition the graph in such a way that the original structure of the graph is preserved.

Graph summarisation vocabulary. We explain in this paragraph several terms that we use henceforth when discussing about graph summarisation. Given a graph $G = \langle V, A, l_V \rangle$, the process of graph summarisation creates another graph, called the summary of $G$. Each node of $G$ is mapped to one or more nodes of the summary, which we call sumnodes. Similarly, edges of the graph $G$ are also mapped to edges in the summary, which we call sumedges. The goal of the graph summarisation is to create a summary where the structure of the entity graph $G$ is preserved.

3.3. Graph Exploration

Graph summarisation techniques can be used for exploring a graph. In this scenario, the structure of the graph is (partially) unknown; it is necessary however to know it in order to compose queries over the graph. The user leverage the summary to learn what at-
3.3. Graph Exploration

Attributes are used, what types, and how they relate with each other. In the time-line, the research works for this application are depicted with a red horizontally-striped trapeze in Figure 3.3.

Tian et. al [THP08] propose the use of graph summarisation for exploring a graph: by grouping nodes that share some pre-defined criteria, the “general” structure of the graph is brought out. The coarseness of the grouping can also be tuned so to retain more or less details about the graph. This work is a bridge to the OLAP (online analytical processing) world, since it allows OLAP-style exploration of the graph by providing drill-down and rollup operations. The user can drill-down, i.e., view more details about the structure of the graph, and rollup, i.e., browse a more abstract view of the structure of the graph. This is made possible by showing a summary of the graph at each level of detail in the drill-down/rollup exploration.

The proposed approach creates a summary graph by mapping each node of the graph into exactly one sumnode; several nodes can map to a same sumnode in the summary. Two nodes are mapped to a same sumnode if they share the same (a) attributes’ value; and (b) incident sumnodes of a node maps to. The algorithm first partition the graph in order to achieve (a), then iteratively splits each block (sumnode) in the partition until each node in a sumnode fulfils (b). The latter step relies on the incident matrix between nodes in the graph and sumnodes in the summary to decide whether to split a sumnode or not; there is one such matrix per attribute. A cell $a_{i,j}$ of the matrix is equal to 1 if a node on row $i$ maps to a sumnode which is a neighbour of the sumnode on column $j$, otherwise it is equal to 0.

The algorithm requires several passes over the graph in order to fulfill (b). For large graphs that do not fit in memory, this becomes a challenge in itself. Indeed, if some part of the graph is not in memory, it needs to be retrieved, e.g., from disk, which would impact negatively on the algorithm performance.

The authors extend this work in [ZTP10] where a measure for the “interestingness” of a graph summary is proposed. The aim of this measure is to quantify how easy it is for a person to visualise and understand the high-level structural characteristics of the graph.

Our contribution. In Chapter 5, we argue that the interestingness of a summary actually depends on the application it is used for. In [ZTP10] the authors also investigate how to summarise numeric data; while this is an important aspect of graph summarisation in some application, we focus in this thesis on summarising the structure of the graph.

Liu et. al [ZJH11] propose a graph summarisation that approximates the information in the entity graph, i.e., the structure of the graph and the attributes’ value associated with a node. The authors aim to construct a summary that is homogeneous, i.e., each node mapped to a sumnode of the summary are consistent with one another. Three criteria that measure different aspect of the summary homogeneity are introduced:

- all nodes mapped to a sumnode have the same attributes’ value;
Chapter 3. Graph Summarisation: Related Work

- a sumedge \((a, \alpha, b)\) implies that all nodes in the sumnode \(a\) are connected to at least one node in \(b\); and
- all nodes mapped to a sumnode must have the same degree\(^1\).

The quality of a criterion is measured using the entropy. A summary built with all three criteria fulfilled can be approximated — so to achieve a smaller size — by relaxing a criterion.

Two algorithms are presented for creating an approximate summary. The first one is an agglomerative approach which starts from the exact graph partition according to the three criteria, then proceeds to merge sumnodes until reaching a pre-defined number of sumnodes. At each iteration, the merged sumnodes are those that would result in minimal entropy. The second algorithm instead relies on a K-Means clustering of the graph and split sumnode until a pre-defined number of sumnodes is reached.

Limitations. The first algorithm presents the challenge of starting from the exact partition, which is computationally expensive as it requires several iterations over the graph. Both algorithms use a pre-defined number of sumnodes, although a user probably has no idea what number to choose.

The authors in [KGSS12] propose a three-layer index on top of the Linking Open Data (LOD). Each layer provides a view on the data with varying details. Ultimately, the index allows to select data sources that are relevant for a SPARQL query. The selection is driven by the three layers of the index, where a layer is more suited to a query of specific complexity. Each layer of the proposed index provides views of the LOD graph with varying granularity: the lower the layer, the more detailed it is.

Our contribution. A layer can be considered as a possible summarisation of the LOD graph, although connections between sumnodes within a layer is not provided. The proposed index model is then orthogonal to our graph summarisation model, each layer being a summary itself.

We note that the third layer which is the most detailed one is based on bisimulation; however, the definition used in that work is not the commonly used one, which we present in Section 4.1.2.

3.4. Data Profiling

A graph summarisation algorithm can be used for providing insights into a graph by accumulating some statistics during the summary construction process. In the time-line of Figure 3.3, literature pertaining to this use case are depicted with a red dotted-trapeze.

\(^1\) Degree is the number of nodes adjacent to a node.
3.4. Data Profiling

Khatchadourian et. al [KC10] describe the structure of the graph along with statistics thanks to summaries. The proposed summary is built using the bisimulation equivalence, which is described in Section 4.1.2.

**Our contribution.** The presented approach is tailored towards graphs using specifically the RDF data modelling; instead, we present a graph summarisation framework that is independent of the graph modelling.

The authors consider a directed graph where only nodes are labelled: each label in our graph model presented in Section 2.3 forms a node. We use instead the same graph model for both the entity graph and for the summary graph, thus simplifying the framework.

The labelling scheme proposed in [KC10] of sumnodes and sumedges is limiting, since it considers only parts of the resources’ URI in the entity graph, e.g., the localname. The set of labels thus constructed is too small for finely controlling how much information from the graph is retained. To add more flexibility, we decouple the definition of a node from the creation of its label, where the former is discussed in Section 4.1 and the latter in Section 4.2.

The authors discuss the summarisation of inter-linked datasets through the use of specific labelled edges, i.e., owl:sameAs edges. Instead, we consider in this scenario the dataset label, which we detail in Section 7.

Navlakha et. al [NRS08] propose a graph summarisation approach based on Information Theory. The result of the approach is the creation of two data structures, i.e., the summary graph and a set of corrections. The set of corrections allows to rebuild the entity graph by correcting any error in the summary graph, e.g., an instance of a sumedge links two disconnected nodes in the entity graph. The algorithm is based on the minimum description length [Grü07] principle which aims to minimise the cost of the summary representation of the graph, i.e., the sum of the summary’s size and of the corrections’ set size. The size of the summary can be further reduced by allowing some errors; there is then a trade-off between the precision and the size of the summary.

**Our contribution.** In contrary, we do not attempt to rebuild the entity graph from the summary and so we focus on summarising the structure of the graph.

Liu et al. [LTH+14] present a graph summarisation approach based on the Message Passing paradigm. The algorithm iteratively merges nodes of the graph until a predefined number of nodes are left; the merged nodes constitute then the sumnodes of the summary. Several solutions for selecting which nodes to merge are discussed, and a method based on Locality Sensitive Hashing [GIM99] is introduced so to improve the merging process.

A sumedge indicates in this approach that all nodes that are mapped to the adjacent sumnodes are connected to each other as well. A subset $A \subset V$ of nodes in the graph are mapped to a sumnode $a$, and similarly, a subset $B \subset V$ of nodes are mapped to
a sumnode $b$; then, the existence of a sumedge $(a, \alpha, b)$ in the summary means in this approach that every node in $A$ is also connected to a node in $B$. In order to reduce the size of the summary, this requirement can be relaxed when merging: the merge that incurs the less error is chosen.

**Our contribution.** In comparison, our framework simplifies the graph summarisation process thanks to its summary definition. Since it is based on graph homomorphism, we know exactly to which sumnode a node of the graph is mapped to; therefore, our graph summarisation is not an iterative process, hence simplifying the overall execution.

### 3.5. Data Analytics

OLAP (online analytical processing) is a technique for answering analytical queries over multi-dimensional data. Through different kind of operations it allows a user to analyse the data from different perspectives. Each perspective is called a *cube*, obtained by aggregating the values of some dimensions. In the time-line of Figure 3.3, the research works that pertain to this use case are depicted with a plain grey trapeze.

Several works investigate the use of OLAP over graphs: data records analysed through OLAP are now related to each other. The main difference between traditional OLAP and graph-OLAP is the *measure* which traditionally is a number becomes a *graph*. Compared to [THP08] discussed in Section 3.3, the works presented here are anchored in a OLAP background, rather than being a bridge to OLAP from the graph exploration domain.

**Observation.** Among the possible OLAP operations, *rollup* and *drill-down* control the level of details presented to the user about the graph. With regards to our graph summarisation presented in Chapter 4, each level is actually a summary. Therefore, the approaches of OLAP over graphs are orthogonal to ours.

Chen et. al propose in [CYZ+08] two OLAP frameworks *I-OLAP* and *T-OLAP* for analysing graphs. The I-OLAP aggregation is aimed at summarising several graphs representing a same entity. It is then a merge of the graphs that keeps the entity layer, while we rather consider the summarisation of the structure of the graph which is a different view on the graph. The T-OLAP framework, which is further investigated in [QZY+11], modifies the *topology* of the graph. T-OLAP maps nodes into sumnodes according to their attributes’ value, and computes a statistic for the sumnode using an aggregate function, e.g., *count*.

**Our contribution.** Compared to our graph summarisation, the authors consider the attributes’ value while we do not require those.
Zhao et al. [ZLXH11] present a framework for analysing multi-dimensional graphs. Rooted in the OLAP domain, the framework retains the relationships between nodes, information which is normally discarded in traditional OLAP. The work presents the summary as the output of a query over the graph, which aggregates nodes sharing a fixed set of attributes’ value.

**Our contribution.** The proposed framework considers a graph with a fixed set of attributes. However, we consider a graph which vocabulary and structure is heterogeneous. Therefore, it is unlikely the graph can be effectively collapsed over a pre-defined set of labels; instead we propose a model that can handle the heterogeneity of the graph.

The summary in [ZLXH11] is defined based on an equivalence relation; two nodes of the graph are equivalent if they share the same attributes’ value. The equivalence relation forms classes, which stand for the sumnodes of the summary; two sumnodes are connected if any node they contain are connected in the graph.

**Our contribution.** We propose instead a framework based on graph homomorphism which is more expressive in the definition of the summary.

Figure 3.4 depicts the summarisation of a graph as per [ZLXH11]; it presents three sumnodes $S_1$, $S_2$, and $S_3$, in which nodes of the graph they are associated with are represented as a cross. Every node in a sumnode is equivalent to one another; however, this does not affect the connectivity between the sumnodes as $S_2$ and $S_3$ are linked although only two of their nodes out of five actually are.

**Our contribution.** In our framework, conditions affecting the connectivity in the summary can be defined explicitly if needed. Furthermore, the graph homomorphism is also a more natural definition of the summary, since homomorphism is an operation that retain the structure of the graph.
Colazzo et al. [CGMR14] describe a framework for using OLAP over RDF data. Each cube represents a summary and is built based on SPARQL queries which define sumnodes and sumedges. This analytical framework is flexible enough to accommodate many kinds of analysis, since the aggregate function is also based on a SPARQL query. Although the analysis of the data is done through its cubes by a user, the computation of a cube is always done over the original data.

**Observation.** The need of writing SPARQL queries suggests that a user possesses some knowledge about the structure of the graph. This work is orthogonal to the one proposed in this thesis: our approach highlights the graph structure, thanks to which a deeper analysis of the graph can be done.

Wang et al. generalise the concept of OLAP over graphs in [WFW+14] by proposing a property graph model and allowing to summarise over the properties associated with edges and/or nodes; a property graph model is a graph where a node and an edge is associated with a list of property-value pairs.

**Our contribution.** This approach focuses on summarising the graph based on the values of a pre-defined set of attributes; instead, we consider graphs that are heterogeneous and also possibly with multi-valued attributes.

The generalised graph OLAP approach [WFW+14] is implemented over a MapReduce [DG08] framework. It leverage the lattice graph of all the possible summaries in order to generate several summaries in a single task in the MapReduce framework. A best execution plan for the series of MapReduce jobs is found using a cost model.

**Our contribution.** In this thesis, we show how the Lattice graph can also be used for the purpose of evaluating the precision of a summary in Section 5.

Rudolf et al. [RPBL13] propose a framework for analysing graphs through templates. The graph is summarised given user-defined graph patterns, which are then matched over the graph. An aggregation function is finally applied over the set of matched sub-graphs, e.g., a count function.

**Our contribution.** This approach requires the user to have some knowledge about the structure of the graph for performing the summarisation; instead, our framework does not assume any prior knowledge.
3.6. Query Optimisation

The purpose of the graph summarisation is to create a graph that share the same structure as the original, but is significantly smaller. Hence, that summary can be used in place of the original in order to improve the speed of query optimisation. In the time-line of Figure 3.3, the literature that pertain to this application of the summary are depicted with a blue vertically-striped trapeze.

DataGuides [GW97] is the first work to improve the query execution through the use of an index on the summary of the structure of the graph. The summary is built with the requirement that every path in the summary is unique; this would allow to spend less time searching through each different path. In the presence of a graph with a complex structure, the size of the summary can be as large as the graph, or even larger. This issue is discussed in [GW99], where some constraints of DataGuides are relaxed, e.g., the existence of a path in the entity graph.

DataGuides relies on the notion of automata for the creation of the summary: converting a graph that is a non-deterministic finite automaton to a deterministic finite automaton. Instead, several works leverage the notion of bisimulation [Par81] used in concurrency systems. Bisimulation is a technique that aims to replicate all the paths of a graph in a summary, so that the summary and the original graph cannot be distinguished from a user’s perspective. A detailed explanation of this technique can be found in Section 4.1.2. Retaining all the paths of the original graph in the summary can reveal to be excessive for complex graphs. Therefore, some constraints of the bisimulation are changed so to reduce the size of the summary, or to better adapt it to a specific application. The size of a summary is then balanced against the level of details it stores.

Milo et. al [MS99] extend the work on DataGuides, using the notion of bisimulation to build an index of the structure of the graph. The authors propose to reduce further the size of a summary by defining the shape of query to answer. However, it assumes some knowledge about the structure of the graph in order to define the shape of the query.

The constraints of the bisimulation relation are relaxed so to achieve a smaller summary: Kaushik et. al [KSBG02] propose a graph summarisation approach where the bisimulation is simplified by limiting the length of a path to \( k \)-hops. In parallel, Kaushik et. al [KBNK02] create an index of paths expressions, based on a summary that has been built for a pre-defined query expression. Chen et. al [CLO03] propose instead to vary the maximum length of a path per node, depending on the query load of the system. Polyzotis et. al [PG06] propose a system that relies on the notion of bisimulation for estimating cardinalities necessary during query evaluation.

Tran et. al [TLR12] use a similar approach to [KBNK02] for improving the partitions of RDF data and the execution of queries. The data is partitioned into groups that have a similar structure so to reduce (a) the communication load during query evaluation; and (b) the number of joins.

Riondato et al. [RGSB14] present a graph summarisation approach that bridges to graph clustering. The summarisation is expressed as an optimisation problem, where
an error function is to be minimised. The function measure the error for reconstructing the entity graph from the summary. This function is expressed as the matrix distance between the adjacency matrices of the graph and its summary. The number of nodes in the summary is set as a parameter to the algorithm, which can be difficult for the user to decide which value to choose.

**Our contribution.** We propose in this thesis a generic graph summarisation framework in which the summarisation is defined based on different features of the graph, e.g., attributes or types. No a priori knowledge about the graph is required, neither is it necessary to specify the number of sumnodes to reach. The features used for defining the summarisation can be tuned so to achieve a smaller summary, sacrificing as little as possible the amount of details recorded about the entity graph.

### 3.7. Summary

In this Chapter, we reviewed the literature related to graph summarisation and compared it with our framework that we present in Chapter 4. We outline in this section the core differences between our and the presented approaches to graph summarisation.

**Expressiveness.** The presented graph summarisation algorithms are designed for specific applications, and can then be difficult to port. We present a flexible and expressive graph summarisation based on graph homomorphism. The definition of the summarisation simply needs the user’s specification of how a node in the graph is mapped to a sumnode. The mapping is simply based on features of a node. In addition, since our summarisation is based on homomorphism, the mapping process of a sumnode to a node is deterministic; there is no need to specify properties of the summary in advance, e.g., the number of sumnodes.

**Performance.** Graph summarisation algorithms require to visit each edge of the entity graph at least once. In order to improve the quality of the summary, some approaches [ZJH11] perform several iterations over the graph. Iterations are necessary when the summarisation of a node also depends on nodes that are not directly connected to it, and so there is a need to traverse the graph. The larger the graph is, the more expensive it is to iterate over the graph. While such algorithms can be expressed in our framework, it is not a necessity in order to summarise a graph.

**Generic.** We present a generic graph summarisation approach that can be used as a base in a variety of applications. For example, a summary is core to drill-down/rollup operations in OLAP over graphs.
Chapter 4.

Graph Summary: a Novel Schema for Web Data

The Web Data consists of a tremendous amount of structured datasets, coming from a variety of sources. An attractive aspect of the Web Data is the total freedom with regards to the production of “knowledge”; as stated by Berners-Lee in [BL97], “Anyone can say anything about anything”. Undoubtedly, the facility to publish data played a role in the considerable growth of the Web Data. However, this also means that there is nothing enforcing some quality of the data, e.g., what design structure or vocabulary terms to use, or checking data consistency.

Although there exists a number of ontologies for describing a wealth of data, datasets do not follow strictly the specifications. Enforcing the structure would stand against the design choice of the Web Data stated above. Another approach is to go with the flow and generate such specifications directly from the data itself.

Graph summarisation is a technique that maps an entity graph into another, smaller graph. That new graph keeps the same structure as the entity graph, but contains less nodes and less edges. As we aim to highlight the underlying structure of a graph, we consider already existing features for the summarisation, built alongside the graph. Such features are for example the Attributes and Types of a graph. In particular, data exhibiting graph heterogeneity with regards to the structure or the vocabulary can be summarised effectively with the presented methods. It is important to note that the notion of data quality indicates in general errors or mistakes in the data. This is not a concern within this thesis as data quality is orthogonal to the notion of heterogeneity.

In this chapter, we present a generic framework for summarising graphs in the context of Web Data, generating either precise or approximate summaries. In addition, we describe possible implementations of the framework.

4.1. Graph Summarisation

Web Data contains a large amount of structured data spanning over many different domains, from information on movies to the description of genes. There are billions of statements and millions of entities. There is a wealth of vocabulary terms for every kind of data, which are used more or less in the way a vocabulary was design for.
In order to make sense of this deluge of data, summarisation techniques are available for different levels of the data. For example, all the information related to an entity can be summarised so to highlight the important parts. In this section, we are interested in the summarisation of the structure of graphs. In the context of Web Data, the structure of a graph is defined by the use of predicates and classes.

We present first a model for graph summarisation in Section 4.1.1. Then in Section 4.1.2, we introduce a model for a precise summary. Finally, we emphasise the direction of approximate graph summarisation as the only viable summarisation for Web Data in many cases in Section 4.1.3.

4.1.1. Model

In this section, we present a formal model of graph summarisation.

Summarisation Requirements

Graph summarisation is an operation over the graph that abstracts from its content, in order to highlights its structure. From a structural point of view, traversing a graph or its summary is equivalent. The summary of a graph exhibits the following properties:

1. a path in the graph exists also in the summary;
2. both the graph and its summary share the same vocabulary; and
3. a graph may conform to several summaries.

These properties of a summary are fulfilled by defining the graph summarisation as a graph homomorphism.

Graph Homomorphism

A graph is homomorphic to another one if there exists a mapping of nodes that matches edges from the first to the second graph. This ensures that the structure of the first graph is kept. The Figure 4.1 depicts two graphs, where there exists a relation $R$ that maps nodes of the left graph to nodes of the right graph. The edges of the left graph are also kept in the right graph. Indeed, there is an edge from nodes 2 and 3 to 1, and as well there is an edge between their mapped nodes — nodes $a$ and $b$ respectively.

Definition 4.1.1 (Graph Homomorphism)

Let $G = \langle V, A, l_V \rangle$ and $S = \langle W, B, l_W \rangle$ be two graphs. Let $R \subseteq V \times W$ be a binary relation. In the context of the binary relation, $G$ is homomorphic to $S$ with regards to $R$ if every edge in $G$ is mapped to an edge in $S$:

$$\forall(u, \alpha, v) \in A \quad \forall(x, y) \in W \quad (u, x) \in R \land (v, y) \in R \implies (x, \alpha, y) \in B$$

We say that $G$ is homomorphic to $S$ with regards to $R$.  

36
4.1. Graph Summarisation

A graph homomorphism is defined by a many-to-many binary relation \( R \) from the nodes \( V \) to the nodes \( W \), and that for every edge in \( A \) there is a corresponding edge in \( B \). Whenever two nodes in the graph \( G \) are linked by an attribute \( \alpha \), then so are their corresponding nodes in the graph \( S \).

**Remark.** From this point on, we shall differentiate between the nodes and edges of the entity graph from those of the summary by calling a node of the summary a *sumnode*, and an edge a *sumedge*. Unless stated explicitly, the terms node and edge refer to the components of the entity graph.

**Features.** We denote as feature elements of the graph on which the relation of the graph homomorphism is based on. As an example, Attributes is a feature of the Attributes summary as defined in Definition (4.1.14). In this thesis, we focus on the following features but others may be used:

- attribute, whether incoming or outgoing of a node; and
- type.

**Source material.** A graph and its summary do not share the same set of nodes. The nodes of the summary are taken from a set that is distinct from the set of nodes in the graph. We refer to that set as the source material which we denote as \( S \). We define in the following paragraphs two nodes of the summary that are used in its creation, i.e., the undefined sumnode and the sink sumnode.

**Undefined sumnode.** Depending on the definition of the binary relation \( R \) used in a graph homomorphism, some nodes of the graph do not have any mapping. For example, the node \( v_0 \) in Figure 2.2 does not have a type edge; it has then no mapping for a...
glssummarisation-relation that considers the type of a node as depicted in the Figure 4.3. We map such a node of the graph to a sumnode $\Omega \in \mathcal{Z}$ that we say to be undefined for the binary relation $R$.

**Definition 4.1.2 (Undefined Sumnode $\Omega$)**

Let $G = \langle V, A, l_V \rangle$ and $S = \langle W \cup \{\Omega\}, B, l_W \rangle$ be two graphs such that $G$ is homomorphic to $S$ with regards to the binary relation $R \subseteq V \times W$. Let $Q \subseteq V \times \{\Omega\}$ be a binary relation. We define the sumnode $\Omega \in \mathcal{Z}$ with $\Omega \notin V$ as the sumnode to which are mapped the nodes of $V$ that do not have an image in $W$ as per the binary relation $R$:

$$(u, \Omega) \in Q \implies \exists x \in W \ (u, x) \in R$$

A graph $G$ that is homomorphic to a graph $S$ with regards to a binary relation $R$ is still the case if we add the undefined sumnode to the graph $S$: we simply need to add the sumedges to the graph $S$ that link to and from the undefined sumnode.

**Sink sumnode: content abstraction.** The purpose of a summary is to highlight the structure of the graph, which is defined by the properties and Types. Content information does not pertain to the structure of the data, but only to individual entities. Therefore, we abstract the summary from the content in the graph, which are in general stored in sink nodes, i.e., nodes that have no outgoing edges. To do so, we map such nodes to a sink sumnode that we identify as $\emptyset$. For example, the node labelled Ireland in the Figure 2.2 is mapped to $\emptyset$. Also, we note that the nodes $S_2$ and $S_3$ in Figure 4.2 are sink sumnodes.

**Remark.** Features of the graph summarisation are not subject to content abstraction, in order for the summary to retain the vocabulary of the original graph.

**Definition 4.1.3 (Sink Sumnode $\emptyset$)**

Let $G = \langle V, A, l_V \rangle$ and $S = \langle W \cup \{\emptyset\}, B, l_W \rangle$ be two graphs such that $G$ is homomorphic to $S$ with regards to the binary relation $R \subseteq V \times W$. Let $P \subseteq V \times \{\emptyset\}$ be a binary relation. We define the sink sumnode $\emptyset \in \mathcal{Z}$ with $\emptyset \notin V$ as the sumnode to which all sink nodes in $G$ are mapped to:

$$u \in V \ \forall \alpha \in \mathcal{L} \ \forall v \in V \ (u, \alpha, v) \notin A \implies (u, \emptyset) \in P$$

**Graph Summary**

Given two graphs $G$ and $S$, if $G$ is homomorphic to $S$ with regards to a binary relation $R$, then we call $S$ the summary of $G$ as per the binary relation $R$ [CPC+12]. Indeed, the summary being homomorph to $G$ retain by definition the paths in the graph described in the property (1). Both graph share the set of labels $\mathcal{L}$, thus meeting the property (2) of a summary.
4.1. Graph Summarisation

![Graph G and Summary S](image)

**Definition 4.1.4 (Graph Summary)**

Let $V \subseteq Z$ and $W \subseteq Z$ be two sets of nodes where $Z$ is the set of source material such that (a) $U \in V$; (b) $\emptyset \in W$; and (c) $V \cap W = \emptyset$.

Let $G = \langle V, A, l_V \rangle$ and $S = \langle W, B, l_W \rangle$ be two graphs where $l_V$ and $l_W$ are two labelling functions such that $l_V : V \mapsto L$ and $l_W : W \mapsto L$; and let $R \subseteq V \times W$ be a binary relation.

We call the graph $S$ a summary of $G$ with respect to $R$ — which we refer to as the summarisation relation — if $G$ is homomorphic to $S$ with respect to $R$.

Figure 4.2 depicts a possible summary for a graph. The dotted lines illustrate the summarisation relation $R$, where we have for example $(a, S_1) \in R$ and $(John, S_3) \in R$.

**Uniqueness.** Given a summarisation relation, each node in the graph is mapped to a set of sumnodes in the summary. If the mapping is deterministic, then there exists a unique graph summary for a graph within the context of that summarisation relation. The summarisation relations proposed in Section 4.1.3 generate a unique summary for a given entity graph. Indeed, the conditions on which a mapping is based is deterministic. For example, the summarisation relation $R_t$ introduced in that section bases its mapping on the set of Types associated to a node.

4.1.2. Precise Graph Summary

The purpose of a summary is to mirror the structure of the graph while being significantly smaller. Therefore, the summary can substitute the graph and improve the performance of applications. How well the summary mirrors the graph is then of high importance. From a structural perspective, a summary is precise if it is indiscernible from the entity graph, which we define in this section.

Although the structure of the graph is preserved in the summary by definition of graph
homomorphism, the structure of the summary may not exactly reflect the structure of the graph. Indeed, a summary may have paths that are not present in the entity graph $G$, as depicted in the Figure 4.1.

Indeed, we remark that although the structure of the graph $G$ is kept in the summary, the inverse may not hold true. The Figure 4.1 depicts a loop on the $b$ node; although there is an edge from 1 to 4, there is no edge in the opposite direction.

A summary is built from a graph based on a summarisation relation that maps each node of that graph to a sumnode, i.e., a node of the summary. An edge $(u, \alpha, v) \in A$ is mapped to a sumedge by definition of the graph homomorphism. Several paths in the graph may be mapped to a same path on the summary. We introduce the summary path instance as the set of nodes in the graph which mappings form a path in the summary. For example, if we consider the summary in Figure 4.3 built with the summarisation relation $R_t$, the set $\{v_1, v_3, v_6\}$ in an instance of the path $(S_1, \text{lives}, S_2) \in \mathcal{W} \land (S_2, \text{lives}, S_4) \in \mathcal{W}$ in that summary; indeed, we have $(v_1, S_1) \in R_t$, $(v_3, S_2) \in R_t$, and $(v_6, S_4) \in R_t$.

**Definition 4.1.5 (Summary Path Instance)**

Let $G = (V, A, l_V)$ be a graph, and $S = (\mathcal{W}, \mathcal{B}, l_{\mathcal{W}})$ be the summary of $G$ according to the summarisation relation $R \subseteq V \times \mathcal{W}$.

Let $p = (x_1, \alpha_1, x_2) \in \mathcal{B} \land \cdots \land (x_n, \alpha_n, x_{n+1}) \in \mathcal{B}$ be a path in the summary $S$ with $(x_1, \cdots, x_{n+1}) \in \mathcal{W}^{n+1}$. We call the set $\{u_1, \cdots, u_{n+1}\}$ an instance of the summary.
4.1. Graph Summarisation

If each node \( u_i \in V \) is mapped to a sumnode \( x_i \) of \( p \) by the relation \( R \), i.e.,:

\[
\forall i \in [1, n] \ (u_i, x_i) \in R
\]

In the previous example, we remark that the summary path instance \( \{v_1, v_3, v_6\} \) given for \((S_1, \text{lives}, S_2) \in W \land (S_2, \text{lives}, S_4) \in W\) in Figure 4.3 is not the only one. Indeed, there are three more instances possible, i.e., \( \{v_1, v_3, v_7\} \), \( \{v_2, v_3, v_6\} \), and \( \{v_2, v_3, v_7\} \). However, two out of those four do not form a path in the graph in Figure 2.2, i.e., the sets \( \{v_1, v_3, v_7\} \) and \( \{v_2, v_3, v_7\} \). We call a summary precise if there is no such case.

**Definition 4.1.6 (Precise Graph Summary)**

Let \( G = (V, A, l_V) \) be a graph, and \( S = (W, B, l_W) \) be the summary of \( G \) according to the summarisation relation \( R \subseteq V \times W \).

Let \( p = (x_1, \alpha_1, x_2) \land \cdots \land (x_n, \alpha_n, x_{n+1}) \in B \) be a path in the summary \( S \) with \((x_1, \cdots, x_{n+1}) \in W^{n+1}\). Let the set \( \{u_1, \cdots, u_{n+1}\} \) be a summary path instance with \( u_i \in V \).

We say that the summary \( S \) is precise if each instance of a summary path \( p \) forms a path that does exist in the graph \( G \) with regards to the Attributes in \( p \):

\[
\forall i \in [1, n] \ \exists (u_i, \alpha_i, u_{i+1}) \in A
\]

**Bisimulation**

The bisimulation [Par81] is a notion of concurrency that studies the equality of processes. A bisimulation is a binary relation on \( V \) that relates two nodes of the graph if itself and its inverse are simulations.

If we consider two nodes \((u, v) \in V\), a simulation \( R \) is a relation that states that for every edge \((u, \alpha, x) \in A\), there exists an edge \((v, \alpha, y) \in A\) such that there is also a simulation between the nodes \( x \) and \( y \). Intuitively, \((u, v) \in R\) communicates that the node \( v \) can substitute the node \( u \) since all the outgoing paths from \( u \) match those from \( v \). A bisimulation is stronger as it states that the relation must be symmetric as well, thus ensuring that either node may substitute the other.

Figure 4.4 depicts the simulation relation \( R \) between nodes \( v_0 \) and \( v'_0 \) such that \((v_0, v'_0) \in R\); this illustrates that:

1. for every outgoing edge from \((v_0, \alpha_i, v_i)\), there is also an edge with the same label from \( v'_0 \); and
2. there is a simulation between the nodes thus reached \( v_i \) and \( v'_i \); i.e., \((v_i, v'_i) \in R\).

If for the node \( v'_0 \) the previous two points hold as well, then the binary relation \( R \) is actually a bisimulation.

**Definition 4.1.7 (Bisimulation)**

Let \( G = (V, A, l_V) \) be a graph and \( \sim \subseteq V \times V \) a binary relation on \( V \). The relation \( \sim \)
Chapter 4. Graph Summary: a Novel Schema for Web Data

Figure 4.4.: Illustration of the conditions for a node \( v'_0 \) to substitute a node \( v_0 \), as per the simulation \( R \). If the reverse is also true, i.e., that the node \( v_0 \) can substitute \( v'_0 \), then the nodes \( v_0 \) and \( v'_0 \) are said to be bisimilar.

**is a bisimulation** if \( \forall (x, y) \in \sim : \)

\[ \forall (x, \alpha, x') \in A \exists (y, \alpha, y') \in A \land (x', y') \in \sim \]

The converse must hold as well, i.e.:

\[ \forall (y, \alpha, y') \in A \exists (x, \alpha, x') \in A \land (x', y') \in \sim \]

**Remark.** Two nodes \( (u, v) \in V^2 \) are said **bisimilar** if \( (u, v) \in \sim \).

The coinductive aspect of the definition ensures that two nodes are bisimilar if their **outgoing** paths are the same.

**Forward/Backward bisimulation.** Kaushik et al. [KBNK02] propose the **forward-and-backward** (f&b)-bisimulation which extends the definition by considering incoming edges in addition to the outgoing ones. Since a summary deals with the structure of the graph given by the relationships of Attributes and Types, we introduce the type as an additional requirement.

In addition to the requirements of a f&b-bisimulation between two nodes, we consider their type information. We define the **fbt-bisimulation** as the f&b-bisimulation that ensures that nodes are equivalent with regards to their type as well.

**Definition 4.1.8 (FBT-Bisimulation)**

Let \( G = (V, A, l_V) \) be a graph and \( (\sim_f, \sim_b, \sim_t) \subseteq (V \times V)^3 \) be three equivalence
relations on \( V \). The relation \( \approx_{ft} \subseteq V \times V \) on \( V \) is a fbt-bisimulation if \( \forall (x, y) \in \approx_{ft}, \) we have \((x, y) \in \sim_f, (x, y) \in \sim_b \) and \((x, y) \in \sim_t \) such that:

1. \( \sim_f \) is a forward bisimulation such that:
   \[
   \forall (x, \alpha, x') \in A \exists (y, \alpha, y') \in A \land (x', y') \in \sim_f
   \]
   conversely, \( \forall (y, \alpha, y') \in A \exists (x, \alpha, x') \in A \land (x', y') \in \sim_f \)

2. \( \sim_b \) is a backward bisimulation such that:
   \[
   \forall (x^{-1}, \alpha, x) \in A \exists (y^{-1}, \alpha, y) \in A \land (x^{-1}, y^{-1}) \in \sim_b
   \]
   conversely, \( \forall (y^{-1}, \alpha, y) \in A \exists (x^{-1}, \alpha, x) \in A \land (x^{-1}, y^{-1}) \in \sim_b \)

3. \( \sim_t \) preserves the types such that:
   \[
   \forall (x, \tau, t) \in A \exists (y, \tau, t) \in A
   \]
   conversely, \( \forall (y, \tau, t) \in A \exists (x, \tau, t) \in A \)

**Forward bisimulation.** We define the \( ft \)-bisimulation as the summarisation relation based on the bisimulation relation and on the type feature. Therefore with this relation, we consider only the forward paths in the bisimulation.

**Definition 4.1.9 (FT-Bisimulation)**

Let \( G = (V, A, l_V) \) be a graph and \((\sim_f, \sim_t) \subseteq (V \times V)^2 \) be two equivalence relations on \( V \). The relation \( \approx_{ft} \subseteq V \times V \) on \( V \) is a ft-bisimulation if \( \forall (x, y) \in \approx_{ft}, \) we have \((x, y) \in \sim_f \) and \((x, y) \in \sim_t \) such that:

1. \( \sim_f \) is a forward bisimulation such that:
   \[
   \forall (x, \alpha, x') \in A \exists (y, \alpha, y') \in A \land (x', y') \in \sim_f
   \]
   conversely, \( \forall (y, \alpha, y') \in A \exists (x, \alpha, x') \in A \land (x', y') \in \sim_f \)

2. \( \sim_t \) preserves the types such that:
   \[
   \forall (x, \tau, t) \in A \exists (y, \tau, t) \in A
   \]
   conversely, \( \forall (y, \tau, t) \in A \exists (x, \tau, t) \in A \)

**Bisimulation Summary**

Since the bisimulation is an equivalence relation, we can create equivalence classes defined as \( [x] = \{ x \in V \mid y \in V, x \sim y \} \); a class is the set of nodes that are bisimilar. An
equivalence class is then exactly a sumnode of the summary. The summarisation relation is in this case a one-to-one mapping of the equivalence class to the sumnode.

**Definition 4.1.10 (Bisimulation Summary)**

Let $G = \langle V, A, l_V \rangle$ and $S_{fbt} = \langle W_{fbt}, B_{fbt}, l_{W_{fbt}} \rangle$ be two graphs. Let $\approx_{fbt} \subseteq V \times V$ be a fbt-bisimulation relation.

The set of nodes $W_{fbt} \subseteq \mathcal{Z}$ contains as many elements as there are equivalence classes as per the fbt-bisimulation relation:

$$|W_{fbt}| = |\{[x] | \exists y \in V \ x \approx_{fbt} y\}|$$

We call $S_{fbt}$ the bisimulation summary of $G$ according to the summarisation relation $R_{fbt} \subseteq V \times W_{fbt}$ defined as a one-to-one mapping between the set of equivalence classes and the set $W_{fbt}$:

$$R_{fbt} = \{(u, x) \in V \times W_{fbt} | \forall v \in [u] \ (v, x) \in V \times W_{fbt} \land \forall y \in W_{fbt} \ x \neq y \ (v, y) \not\in V \times W_{fbt}\}$$

The summary depicted on the Figure 4.5 with the bisimulation $R_{fbt}$ assigns an equivalence class to each node of the graph in Figure 2.2. In the running example, the only difference of the summary with the entity graph is the content abstraction, e.g., the node Ireland is represented by the node $\emptyset$.

In order to compute this summary, the algorithm proposed in [PT87] is generally used. It offers a $O(|A| \times \log(|V|))$ complexity for the computation of the bisimulation. We note that the algorithm starts from an existing partitioning of the graph. Then, the partitions are refined iteratively until all nodes in a partition are bisimilar.

**Forward bisimulation summary.** We define as $FT$-Bisimulation summary $S_{ft}$ the summary of $G$ according to the $ft$-bisimulation as the summarisation relation. Unlike the previous bisimulation summary, this graph summary considers only the forward paths.
4.1. Graph Summarisation

Figure 4.5.: Summary of the graph in Figure 2.2 with bisimulation $R_{ft}$ as the summarisation relation. The table indicates the mappings by $R_{ft}$. The content in $G$ is abstracted by the sumnode $∅$.

Definition 4.1.11 (FT-Bisimulation Summary)

Let $G = \langle V, A, l_V \rangle$ and $S_{ft} = \langle W_{ft}, B_{ft}, l_{W_{ft}} \rangle$ be two graphs. Let $\approx_{ft} \subseteq V \times V$ be a ft-bisimulation relation.

The set of nodes $W_{ft} \subseteq Z$ contains as many elements as there are equivalence classes as per the ft-bisimulation relation:

$$|W_{ft}| = |\{[x] | \exists y \in V \ x \approx_{ft} y\}|$$

We call $S_{ft}$ the ft-bisimulation summary of $G$ according to the summarisation relation $R_{ft} \subseteq V \times W_{ft}$ defined as a one-to-one mapping between the set of equivalence classes and the set $W_{ft}$:

$$R_{ft} = \{(u, x) \in V \times W_{ft} | \forall v \in [u] \ (v, x) \in V \times W_{ft} \land \forall y \in W_{ft} \ x \neq y \ (v, y) \not\in V \times W_{ft}\}$$

4.1.3. Approximate Graph Summary

In general, we find in Web Data a heterogeneous use of vocabularies, where several different ontologies can be used within the same dataset, in ways that might not have been intended for. A reasonable explanation is that each person has own way of modelling data, thus making it difficult to fit a pre-defined ontology. Web Data presents a complex graph structure that varies greatly from dataset to dataset.
In addition, Web Data is composed of user-created content which quality is not assessed, e.g., use of an appropriate ontology, typographic errors when editing, incorrect description of an entity, etc. This quality issue, combined with the heterogeneity of the graph, is an obstacle towards the generation of a precise summary. Indeed, attempting to record every single path would require a summary so large that its benefits would be nullified. Therefore, we present in this section approximate graph summaries.

**Heterogeneous Graph Structure**

We investigate the direction of approximate summarisation for the creation of a graph summary. DataGuides [GW97] were proposed to index the structure of data following the OEM [PGMW95] data model. The creation of a DataGuide is equivalent to the conversion of a non-deterministic finite automaton to a deterministic finite automaton. Due to heterogeneous structure of the data, the size of a DataGuide grows exponentially, becoming larger than the original data as noted by Goldman et al. [GW99].

We created in [CCP+11] a dataset based on Sindice [ODC+08] collection for the task of entity-oriented search. The Figure 4.6 depicts the distribution of the frequency of ontologies used across documents, i.e., the probability for an ontology to be used in exactly \( n \) documents. The distribution shows a power-law distribution following a Zipf function with a slope of \( \alpha = 2.27 \).

This investigation has showed that most ontologies (99%) are used only once. However, the distribution tail is sparse, suggesting that a few ontologies are used in a large proportion of documents. This result strengthens our belief that it is not a viable option in many cases for a graph summary to retain every path and combinations of paths that occur in an entity graph.
4.1. Graph Summarisation

Approximate Summarisation Relation

A graph summary can be constructed based on different features of the data. We present here some summarisation relations that consider the following features of the graph:

- the predicate URI;
- the type URI; and
- the direction of links, i.e., incoming or outgoing.

We report in Table 4.1 a summary of the presented relations along with the name of the summary that a relation generates.

Summarisation Relation $R_{ut}$

We define $R_{ut}$ as the summarisation relation that maps a node according to its type. We call the summary it generates the Unique Type summary. The nodes $v_4$, $v_5$, $v_6$, and $v_7$ in the Figure 2.2 are mapped to a same sumnode since they all have Place as a type. The $R_{ut}$ relation may map a node to multiple sumnodes since an entity may have several Types. For instance, the nodes $v_6$ and $v_7$ are also mapped to another sumnode since they both share the type Country.

**Definition 4.1.12 (Unique Type Summary)**

Let $G = (V, A, l_V)$ and $S_{ut} = (W_{ut}, B_{ut}, l_{W_{ut}})$ be two graphs.

The set of nodes $W_{ut}$ is a subset of the source material disjoint from $V$, i.e., $W_{ut} \subseteq Z$ with $V \cap W_{ut} = \emptyset$, and it contains as many elements as there are Types in the graph:

$$|W_{ut}| = |\{l_V(c) \mid \exists(u, \tau, c) \in A \land \tau \in T\}|$$

We call $S_{ut}$ the unique type summary of $G$ according to the summarisation relation $R_{ut} \subseteq V \times W_{ut}$ defined as:

$$R_{ut} = \{(u, x) \in V \times W_{ut} \mid \exists(v, y) \in V \times W_{ut}$$

$$(u, \tau, v) \in A \land (x, \tau, y) \land l_V(v) = l_{W_{ut}}(y) \land \tau \in T\}$$

Summarisation Relation $R_t$

We define $R_t$ the summarisation relation that maps a node based on its set of Types. We call the summary it generates the types summary. The Figure 4.3 depicts the $R_t$ summary of the graph in the Figure 2.2. The nodes $v_6$ and $v_7$ are mapped to the sumnode $S_4$ because they are both associated with the types Place and Country. The table indicates the mappings by $R_t$ of the nodes in $V$, e.g., the nodes $v_1$ and $v_2$ are mapped to the sumnode $S_1$ since both connect to Person via the type attribute. It differs from $R_{ut}$ since it considers the types of a node as set rather than individually. Indeed, the node $v_6$ is mapped to two sumnodes under the relation $R_{ut}$. 

47
Chapter 4. Graph Summary: a Novel Schema for Web Data

Definition 4.1.13 (Types Summary)

Let $G = \langle V, A, l \rangle$ and $S_t = \langle W_t, B_t, l_{W_t} \rangle$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_t$ is a subset of the source material disjoint from $V$, i.e., $W_t \subseteq Z$ with $V \cap W_t = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of types in the graph:

$$|W_t| = |\mathcal{P}\{l_V(c) \mid \exists (u, \tau, c) \in A \land \tau \in T\}|$$

We call $S_t$ the types summary of $G$ according to the summarisation relation $R_t \subseteq V \times W_t$ defined as:

$$R_t = \{(u, x) \in V \times W_t \mid \text{types}(u) = \text{types}(x)\}$$

**Summarisation Relation $R_a$**

We define $R_a$ the relation that maps a node based on its set of Attributes. We call the summary it generates the attributes summary. The nodes $v_3$, $v_4$, and $v_5$ are $\sim_a$-equivalent because they share the same set of attributes, i.e., $\{\text{label, location, type}\}$. The graph in Figure 4.7 is the Attributes summary of the graph in Figure 2.2.

Definition 4.1.14 (Attributes Summary)

Let $G = \langle V, A, l \rangle$ and $S_a = \langle W_a, B_a, l_{W_a} \rangle$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_a$ is a subset of the source material disjoint from $V$, i.e., $W_a \subseteq Z$ with $V \cap W_a = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes in the graph:

$$|W_a| = |\mathcal{P}\{\alpha \mid (u, v) \in V^2 \exists (u, \alpha, v) \in A\}|$$

We call $S_a$ the attributes summary of $G$ according to the summarisation relation $R_a \subseteq V \times W_a$ defined as:

$$R_a = \{(u, x) \in V \times W_a \mid \text{attributes}(u) = \text{attributes}(x)\}$$

**Remark.** We note that the nodes that do not have any outgoing attributes are mapped to the undefined sumnode $\Xi$.

Summarisation Relation $R_{at}$

We define $R_{at}$ as the relation that maps a node based on its set of Types and Attributes. We call the summary it generates the attributes $\&$ types summary. The nodes $v_1$ and $v_2$ are mapped to a same sumnode by $R_{at}$ because they are associated with the same type, i.e., $\text{Person}$, and they have the same attributes, i.e., $\{\text{lives, name, type, works}\}$.
4.1. Graph Summarisation

Figure 4.7.: Summary of the graph in Figure 2.2 with $R_a$. The table indicates the mappings by $R_a$. The content in $G$ is abstracted by the summnode $\varnothing$.

**Definition 4.1.15 (Attributes & Types Summary)**

Let $G = \langle V, A, l_V \rangle$ and $S_{at} = \langle W_{at}, B_{at}, l_{W_{at}} \rangle$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_{at}$ is a subset of the source material disjoint from $V$, i.e., $W_{at} \subseteq Z$ with $V \cap W_{at} = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes and types in the graph:

$$|W_{at}| = |\mathcal{P}\{\alpha \mid (u, v) \in V^2 \exists (u, \alpha, v) \in A\} \cup \{l_V(c) \mid \exists (u, \tau, c) \in A \land \tau \in T\}|$$

We call $S_{at}$ the attributes & types summary of $G$ according to the summarisation relation $R_{at} \subseteq V \times W_{at}$ defined as:

$$R_{at} = \{(u, x) \in V \times W_{at} \mid \text{Types}(u) = \text{Types}(x) \land \text{attributes}(u) = \text{attributes}(x)\}$$

**Summarisation Relation $R_{ioa}$**

We define $R_{ioa}$ the relation based on the set of incoming and outgoing Attributes. We call the summary it generates the IO attributes summary. All three nodes $v_3$, $v_4$, and $v_5$ map to the same summnode by the relation $R_a$. However with $R_{ioa}$, each is assigned to a separate summnode, since all three have different incoming set of attributes, i.e., $\{\text{lives}\}$, $\{\text{works}\}$, and $\emptyset$, respectively.

**Definition 4.1.16 (IO Attributes Summary)**

Let $G = \langle V, A, l_V \rangle$ and $S_{ioa} = \langle W_{ioa}, B_{ioa}, l_{W_{ioa}} \rangle$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

$$\begin{array}{c|c|c}
R_a (V, W) \\
V & W \\
v_0 & S_0 \\
v_1, v_2 & S_1 \\
v_3, v_4, v_5 & S_2 \\
v_6 & S_3 \\
v_7 & S_4 \\
\end{array}$$
Chapter 4. Graph Summary: a Novel Schema for Web Data

The set of nodes $W_{\text{ioa}}$ is a subset of the source material disjoint from $V$, i.e., $W_{\text{ioa}} \subseteq Z$ with $V \cap W_{\text{ioa}} = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes in the graph:

$$|W_{\text{ioa}}| = |\mathcal{P}\left(\{\alpha : (u, v) \in V^2 \exists (u, \alpha, v) \in A\}\right)|$$

We call $S_{\text{ioa}}$ the IO attributes summary of $G$ according to the summarisation relation $R_{\text{ioa}} \subseteq V \times W_{\text{ioa}}$ defined as:

$$R_{\text{ioa}} = \{(u, x) \in V \times W_{\text{ioa}} | \text{attributes}(u) = \text{attributes}(x) \land \text{attributes}^{-1}(u) = \text{attributes}^{-1}(x)\}$$

**Summarisation Relation $R_{\text{ia}}$**

We define $R_{\text{ia}}$ as the relation based on the set of incoming Attributes. We call the summary it generates the *incoming attributes* summary.

**Definition 4.1.17 (Incoming Attributes Summary)**

Let $G = (V, A, l_V)$ and $S_{\text{ia}} = (W_{\text{ia}}, B_{\text{ia}}, l_{W_{\text{ia}}})$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_{\text{ia}}$ is a subset of the source material disjoint from $V$, i.e., $W_{\text{ia}} \subseteq Z$ with $V \cap W_{\text{ia}} = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes in the graph:

$$|W_{\text{ia}}| = |\mathcal{P}\left(\{\alpha : (u, v) \in V^2 \exists (u, \alpha, v) \in A\}\right)|$$

We call $S_{\text{ia}}$ the incoming attributes summary of $G$ according to the summarisation relation $R_{\text{ia}} \subseteq V \times W_{\text{ia}}$ defined as:

$$R_{\text{ia}} = \{(u, x) \in V \times W_{\text{ia}} | \text{attributes}^{-1}(u) = \text{attributes}^{-1}(x)\}$$

**Remark.** We note that the nodes that do not have any incoming attributes are mapped to the undefined sumnode $\mathcal{U}$.

**Summarisation Relation $R_{\text{iat}}$**

We define $R_{\text{iat}}$ as the relation based on the set of incoming Attributes & Types. We call the summary this relation generates the *incoming attributes & types* summary.

**Definition 4.1.18 (Incoming Attributes & Types Summary)**

Let $G = (V, A, l_V)$ and $S_{\text{iat}} = (W_{\text{iat}}, B_{\text{iat}}, l_{W_{\text{iat}}})$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_{\text{iat}}$ is a subset of the source material disjoint from $V$, i.e., $W_{\text{iat}} \subseteq Z$
with $V \cap W_{iat} = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes and types in the graph:

$$|W_{iat}| = |\mathcal{P}\{\alpha \mid (u, v) \in V^2 \exists (u, \alpha, v) \in A\} \cup \{l_v(c) \mid \exists (u, \tau, c) \in A \land \tau \in T\}|$$

We call $S_{iat}$ the incoming attributes & types summary of $G$ according to the summarisation relation $R_{iat} \subseteq V \times W_{iat}$ defined as:

$$R_{iat} = \{(u, x) \in V \times W_{iat} \mid Types(u) = Types(x) \land attributes^{-1}(u) = attributes^{-1}(x)\}$$

Summarisation Relation $R_{ioat}$

The relation $R_{ioat}$ maps a node based on the set of Types, and the incoming and outgoing Attributes sets. We call the summary this relation generates the IO attributes & types summary. Although the nodes $v_1$ and $v_2$ are mapped to a same sumnode with $R_{iat}$, that is not the case with the relation $R_{ioat}$, because only the node $v_2$ has the incoming attribute creator.

Definition 4.1.19 (IO Attributes & Types Summary)

Let $G = \langle V, A, l_v \rangle$ and $S_{ioat} = \langle W_{ioat}, B_{ioat}, l_{W_{ioat}} \rangle$ be two graphs. Let $\mathcal{P}(S)$ be the powerset of the set $S$, i.e., it is the set of all subsets of $S$, including the empty set and $S$ itself.

The set of nodes $W_{ioat}$ is a subset of the source material disjoint from $V$, i.e., $W_{ioat} \subseteq Z$ with $V \cap W_{ioat} = \emptyset$, and it contains as many elements as the powerset $\mathcal{P}$ of attributes and types in the graph:

$$|W_{ioat}| = |\mathcal{P}\{\alpha \mid (u, v) \in V^2 \exists (u, \alpha, v) \in A\} \cup \{l_v(c) \mid \exists (u, \tau, c) \in A \land \tau \in T\}|$$

We call $S_{ioat}$ the IO attributes & types summary of $G$ according to the summarisation relation $R_{ioat} \subseteq V \times W_{ioat}$ defined as:

$$R_{ioat} = \{(u, x) \in V \times W_{ioat} \mid Types(u) = Types(x) \land attributes(u) = attributes(x) \land attributes^{-1}(u) = attributes^{-1}(x)\}$$

We note that the presented approximate summarisation relations are indeed not precise, because they do not fulfill the Definition 4.1.6. Indeed, it is possible to find a path in the approximate summary for which an instance as per Definition 4.1.5 does not form a path in the graph. For instance, the set $\{v_1, v_5, v_6\}$ is an instance of the summary path
Chapter 4. Graph Summary: a Novel Schema for Web Data

Notation Summarisation Relation

| Unique Type      | \( S_{ut} = \langle W_{ut}, B_{ut}, l_{W_{ut}} \rangle \) | Definition (4.1.12) |
| Types            | \( S_t = \langle W_t, B_t, l_{W_t} \rangle \)           | Definition (4.1.13) |
| Attributes       | \( S_a = \langle W_a, B_a, l_{W_a} \rangle \)           | Definition (4.1.14) |
| Attributes & Types | \( S_{at} = \langle W_{at}, B_{at}, l_{W_{at}} \rangle \) | Definition (4.1.15) |
| IO Attributes    | \( S_{ioa} = \langle W_{ioa}, B_{ioa}, l_{W_{ioa}} \rangle \) | Definition (4.1.16) |
| IO Attributes & Types | \( S_{ioat} = \langle W_{ioat}, B_{ioat}, l_{W_{ioat}} \rangle \) | Definition (4.1.19) |
| Incoming Attributes | \( S_{iat} = \langle W_{iat}, B_{iat}, l_{W_{iat}} \rangle \) | Definition (4.1.17) |
| Incoming Attributes & Types | \( S_{iat} = \langle W_{iat}, B_{iat}, l_{W_{iat}} \rangle \) | Definition (4.1.18) |

Table 4.1.: Summarisation relations for approximate graph summaries

works · location · capital; however, there is no path that connects all three nodes \( v_1, v_5, \) and \( v_6 \) with the attributes works, location, and capital.

4.2. Graph Summary Generation

In this section, we present an algorithm for generating a graph summary. Some optimisations are possible depending on the relation. However, the proposed approach is independent of the summarisation relation used. The algorithm takes a graph as input and outputs a graph summary, which is distinct from the input graph. We first present a version of the algorithm for a single dataset and then for a collection of several datasets in Section 4.2.1. Next, we describe in Section 4.2.2 two implementations of the algorithm, one based on MapReduce [DG04], and the other on SPARQL.

4.2.1. Algorithms

The creation of a graph summary is decomposed into four steps. These steps are common to both versions of the algorithm, i.e., the graph summarisation of a single dataset and of a collection of datasets. In a first step, we aggregate all the information about a node needed for the summarisation relation. In a second step, we create a table that associates a node of \( G \) to a sumnode. In a third step, we materialise the sumedges by joining the previous table with the entity graph. The fourth step is optional and consists in gathering statistics about the summarised graph.

Step 1: Entity description.

In order to create a graph summary, the basic information we manipulate is an entity. The computation of the entity description \( E \) requires a pass over the edges, with a worst case complexity of \( O(|A|) \). The entity description provides the necessary contextual information needed for the summarisation relation.
4.2. Graph Summary Generation

**Step 2: Node to sumnode mapping.** A sumnode unique identifier is defined based on the entity description $E$, with regards to the summarisation relation. For example, a unique identifier can be created from the Types of an entity for the relation $R_t$. The worst case complexity of this operation is $O(|V|)$, since we need to visit each node of the graph.

**Step 3: Sumedge materialisation.** We need to join the previous mapping of a node to its sumnode with the entity graph in order to materialise an sumedge. We do this by two joins, i.e., on the source and target nodes. In this step, we need to visit each edge in the graph, so in the worst case it has a $O(|A|)$ complexity.

**Step 4: Statistics gathering.** During the previous steps, it is possible to gather statistics about the graph. This step is optional, since depending on the application in which the summary is used statistics might be unnecessary. However, gathering statistics comes at no cost, since we can re-use computations of the previous steps. Indeed in Step 2, we can keep the count of nodes that are mapped to a sumnode. In addition in Step 3, we can count the occurrences of an edge between two sumnodes. Such statistics provide insight into the graph structure, complementing the graph summary.

**Graph Summarisation of a Single Dataset**

We present here the graph summarisation algorithm that is aimed at processing a single dataset. The Algorithm 1 outlines the flow of the graph summarisation which takes a graph as input and outputs its summary for a given summarisation relation.

The summarisation relation is first materialised into a table where each row report two elements: a node and its corresponding sumnode as per the summarisation relation. For each edge of the graph, the algorithm first retrieves the sumnodes corresponding to the source and target nodes thanks to the GetSumnode function. This function performs a lookup on the previous materialised table. We note that since the summary of a graph is unique for a given summarisation relation, the sumnode can be retrieved deterministically.

Then, the corresponding sumedge is added to the summary. The lines 2-5 represent the first two steps of the graph summary creation, and the line 7 corresponds to the materialisation of the sumedge.
Remark. On line 5 we retrieve the sumnode of \( v \) only if it is not a type, since we consider the type as a feature of the summarisation relation.

Algorithm 1: Graph summarisation of a single dataset

**Input:** A graph \( G = \langle V, A, l_V \rangle \)

**Output:** A summary \( S = \langle W, B, l_W \rangle \) of \( G \)

1. **foreach** \((u, \alpha, v) \in A\) **do**
   - \( S_u \leftarrow \text{GetSumnode}(u) \)
   - \( S_v \leftarrow \emptyset \)
   - **if** \( \alpha \notin L_T \) **then**
     - \( S_v \leftarrow \text{GetSumnode}(v) \)
   - **// Build the graph summary**
     - \( W \leftarrow W \cup \{S_u, S_v\} \)
     - \( B \leftarrow B \cup \{(S_u, \alpha, S_v)\} \)

Example. In the Figure 4.8a, we depict the \( Types \) summarisation \( R_t \) of a graph showing people and documents. We group the edges describing a same node in Step 1. In Step 2 we assign the sumnode \( h1 \) to the nodes (i.e., \( :e1 \) and \( :e3 \)) of type \( :Person \), and \( h2 \) to the node (i.e., \( :e2 \)) of type \( :Document \). In Step 3, we join the edges of the input graph with the table created in Step 2 in order to retrieve the sumnodes. It is possible to gather statistics about the summarisation in Step 4 by, e.g., grouping over the sumnodes and counting the number of rows in a group. Figure 4.8b depicts the \( Types \) summary that is the output of the algorithm which execution is depicted in Figure 4.8a.

Graph Summarisation of Inter-Linked Datasets

The Web Data is a collection of semi-structured data that hail from a variety of sources. Sources may provide overlapping information and reference each other. In such an environment, questions of trust about the legality of information arise, e.g., whether some description associated to a node in the graph is accurate or not. Since the graph summary is built from (inter-linked) data, it may present a structure of the graph that is not expected for a given dataset. We discuss in the next paragraph challenges presented in summarising inter-linked datasets.

Challenge in summarising heterogeneous inter-linked datasets. The information about an entity, e.g., a person, a place, an organisation, . . . , can be spread across several datasets. In the Web of Data this happens when a same entity URI is reused across datasets. The Step 2 of the graph summarisation relies on the entity description \( E \) in
4.2. Graph Summary Generation

(a) The input data is a set of edges, one per row. The columns $u_\mathcal{W}$ and $v_\mathcal{W}$ represents the sumnodes associated with the node $u$ and $v$, respectively. The Step 2 assigns the sumnode $h1$ to the nodes $e1$ and $e3$; and the sumnode $h2$ to the node $e3$. These sumnodes are joined with the input edges in Step 3 in order to materialise the sumedges.

(b) Depiction of the Types summary created in Figure 4.8a.

Figure 4.8.: Example of the Types summarisation $R_t$ in the case of a single dataset.
order to create the corresponding sumnode. Some edges of the description might be erroneous, which would then impact negatively on the summary. A solution is to keep track of the origin of an edge and act accordingly during the summarisation process.

In Figure 4.9 we depict the summarisation of a graph which edges are spread over the datasets $D_1$ and $D_2$. The former contains the edges $\{(v_1, a, Person), (v_1, name, John)\}$, while the latter contains $\{(v_1, a, Book), (v_1, title, Rama)\}$. Since the same node $v_1$ is used in $D_1$ and $D_2$, all four edges contribute to the summarisation relation. That node is then mapped to the sumnode $T_1$ according to the $Types$ summarisation relation $R_t$. Due to the overlap over $D_1$ and $D_2$ of $v_1$’s entity description, the sumnode reports that it is of both types $Person$ and $Book$, as well as both Attributes name and title.

Therefore, information about an entity within a dataset can be erroneous since it is unlikely that a same node is typed both a $Person$ and a $Book$. It is likely that either the graph in $D_1$ or in $D_2$ is incorrect; this leads to a misleading graph summary. In order to prevent this issue, we need to differentiate edges within $D_1$ from those in $D_2$.

**Edge authority.** In this section, we introduce the concept of edge authority which is needed for the summarisation of inter-linked datasets. The proposed definition may not fit every application, as it was crafted to fit our view of inter-linked datasets.

**Edge Provenance:** Due to the principle of Linked Data which states that anything can be said about anything, there is a need to know the provenance of an edge.
This necessity goes in accordance with the N-Quads\textsuperscript{1} serialisation format. The provenance of an edge indicates the dataset it is originally from.

**Node Ownership:** To assess the information given by an edge, we need to know the ownership of a node in addition to the edge’s provenance. This concept differs from the provenance of an edge in the sense that a dataset may contain edges about a node, but not necessarily owns that node.

\textbf{Definition 4.2.1 (Edge Provenance)}
\[
\text{Let } G = (V, A, l_V) \text{ be a graph. The provenance of an edge in } G \text{ is the dataset label of } G, \text{i.e., } l_G(G).
\]

\textbf{Definition 4.2.2 (Node Ownership)}
\[
\text{Let } G = (V, A, l_V) \text{ be a graph. The ownership of a node is the function } d : V \mapsto \mathcal{L} \text{ which maps a node to a dataset label.}
\]

\textbf{Remark.} In the case of inter-linked datasets, the provenance of edges that describe an entity may differ from the ownership of that entity.

The authority of an edge combines the concepts of provenance and ownership, in order to determine whether that edge is “legal” or not. An edge is legal if the provenance of the edge is the same as the ownership of the source node.

\textbf{Definition 4.2.3 (Edge Authority)}
\[
\text{Let } G = (V, A, l_V) \text{ be a graph. An edge } (u, \alpha, v) \in A \text{ of the graph } G \text{ has authority if } d(u) = l_{G}(G).
\]

\textbf{Remark.} In RDF, an edge which source node is a blank node has authority implicitly, since it is local to the dataset it originates from.

The Figure 4.10 depicts three cases of authority. A dataset is represented by a colour and a line type, i.e., there are three datasets A, B, and C which are dotted red, solid blue, and dashed green, respectively. The provenance of an edge determines its colour and line type. The ownership of a node is determined visually by the dataset it is in. We refer to the graphs in datasets A, B, C as $G_A$, $G_B$, and $G_C$, respectively.

The ownership of the nodes $a_1$ and $a_2$ is dataset A, and it is dataset B for the node $b_1$. Both edges $(a_1, \alpha, a_2)$ and $(a_1, \beta, b_1)$ have authority since (1) the ownership of the source node $a_1$ is the dataset A, i.e., $d(a_1) = \text{A}$; and (2) the provenance of both edges is also dataset A, i.e., $l_{G}(G_A) = \text{A}$. Since $d(a_1) = l_{G}(G_A)$, both edges aforementioned have

\textsuperscript{1}http://www.w3.org/TR/n-quads/
Figure 4.10.: Edge authority. A dataset is represented by a colour and a line type. The provenance of an edge determines its colour and line type. The ownership of a node is determined visually by the dataset it is in.

authority.

The edge \((a_2, \gamma, b_1)\) does not have authority, since its provenance is \(B\) and the ownership of its source node \(a_2\) is \(A\). Similarly, the edge \((a_1, \delta, b_1)\) does not have authority either, since its provenance (dataset \(C\)) is not equal to the ownership of node \(a_1\) (dataset \(A\)).

Algorithm. In order to summarize a collection of inter-linked datasets, we extend the Algorithm 1 with the dataset label information. The Algorithm 2 modifies the function \(GetSumnode\) so to implement the edge authority. We name it as \(GetSumnode_i\) to mark this change, where the subscript \(i\) denotes the dataset label. We extract the features for the summarisation relation from the edges in the entity description \(E\) only if one of the following conditions is met:

1. the edge has authority; or
2. the provenance of the edge is \(G_i\).

By doing so, we retain the summarisation features from each dataset. This ensures we can summarise a dataset that uses a node which ownership is external to it, without corrupting the information about that node.

The Algorithm 2 takes a collections graphs from \(n\) datasets, and outputs a summary that describes the structure of graphs within and between datasets. This algorithm is the same as the Algorithm 1, but with the redefined \(GetSumnode_i\) function. Given a node \(u \in V\), that function considers the edges from its entity description and filters out those that do not have authority (lines 3-4).

Example. In Figure 4.11a we depict the summarisation \(R_t\) of two inter-linked datasets, \(A\) and \(B\), where a person in the former is related to a person in the latter. The resulting \(Types\) summary is depicted on the Figure 4.11b, where the edges follows the same
4.2. Graph Summary Generation

Algorithm 2: Graph summarisation of a inter-linked datasets

**Input:** A collection of $n$ graphs: $\forall 0 < i < n G_i = \langle V_i, A_i, l_{V_i} \rangle$. Let $G = \langle V, A, l_V \rangle$ be the union of all graphs, i.e., $A = A_1 \cup \cdots \cup A_n$ and $V = V_1 \cup \cdots \cup V_n$

**Output:** A sumnode of the summary $S = \langle W, B, l_W \rangle$ of $G$

```
// The GetSumnode function considers the features that have authority and those which provenance is $G_i$
Function GetSumnode(u ∈ V):
    foreach e ∈ E(u) do // For every edge in the entity description of u
        if $l_G(e) = d(u)$ then // The edge $e$ has authority
            // Extract features for the summarisation relation from $e$
        else if $l_G(e) = l_G(G_i)$ then // The provenance of edge $e$ is $G_i$
            // Extract features for the summarisation relation from $e$
```

representation of the datasets, i.e., those which provenance is $A$ are solid red, and those which provenance is $B$ are dashed blue.

The type features Student and Person of the node $e_2$ are shared between the datasets $A$ and $B$, respectively. Since that node ownership is dataset $B$, the feature which provenance is $B$ has authority, i.e., Person, while the other in $A$, i.e., Student, has not. In order to retain the information provided by dataset $A$ about that node without corrupting the information gathered by the summary, we map the node $e_2$ according to the provenance information:

- for edges which provenance is $B$, calling GetSumnode$_B(e_2)$ returns the sumnode $h_1$; and

- for edges which provenance is $A$, calling GetSumnode$_A(e_2)$ returns the sumnode $h_2$.

4.2.2. Implementations

We present in this section two implementations of the graph summarisation, a first one based on SPARQL and then a second based on MapReduce [DG04]. We present only the implementations of the Algorithm 1 for the summarisation of a single dataset, since they can be easily extended to the Algorithm 2. We outline below the required operators of the graph summarisation algorithm.

**Step 1:** In this step, we need a group operator in order to build the entity description $E$.

**Step 2:** In this step we create the sumnode associated to a node according to its entity description. Therefore, we need a projection operator in order to extract the features of the summarisation relation. We note that a feature can be multi-valued, which needs to be taken into account as well. The creation of the sumnode from the extracted features requires an object invention [HS89] operator.
(a) The input data is a set of edges, one per row. The column “Prov” indicates the provenance of the edge. The columns $u_W$ and $v_W$ represents the sumnodes associated with the node $u$ and $v$, respectively. The ownership of $e_1$ is A, and the dataset B owns $e_2$. The Step 2 assigns the sumnodes according to the function $\text{GetSumnode}_i$ in Algorithm 2.

(b) Depiction of the Types summary created in Figure 4.11a. The edges which provenance is A are solid red, and those which provenance is B are dashed blue.

Figure 4.11.: Example of the Types summarisation $R_t$ in the case of inter-linked datasets.
4.2. Graph Summary Generation

**Step 3:** The materialisation of the sumedges requires a join operator through which we can retrieve the sumnode(s) associated with the source and target nodes.

**Step 4:** As in Step 1 we need a group operator in order to aggregate sumnodes and sumedges, possibly computing statistics at the same time.

**SPARQL implementation**

In this section, we describe an implementation of the graph summarisation algorithm based on SPARQL. The SPARQL query language is rich enough for us to only rely on it for the summary generation. The Figure 4.12b shows the SPARQL query for generating a graph summary, and the query in Figure 4.12a is an example of a summarisation relation, i.e., Types summarisation relation $R_t$.

**Operators**

SPARQL provides the operator `GROUP BY` which can be used to group data over a set of variables. The projection operator is implemented using a `SELECT` query. In the case of a multi-valued variable, we use the `GROUP_CONCAT` operator which allows to concatenate all the values into a single literal. The object invention is performed thanks to built-in hash functions. In a SPARQL query, patterns which reuse some variables for either subject, predicate, or object components create implicitly a join. The computation of statistics can be performed using the built-in aggregate function in conjunction with the `GROUP BY` operator, e.g., `COUNT`. Thanks to the `CONSTRUCT` operator, we are able to create the graph summary directly from the query.

**Step 1 and Step 2**

In the Figure 4.12a, we extract the features relevant for the relation, e.g., the type values in this example. We note the use of the `ORDER BY` in the Figure 4.12a so to ensure that an unique “identifier” is created for the set of features thanks to the built-in hash function SHA1. We use the aggregate `GROUP_CONCAT` in order to pass a single literal to SHA1, since an entity can be associated with multiple types.

**Step 3**

In the Figure 4.12b, the lines 4-7 (resp., lines 14-17) represent the summarisation relation, of which the Figure 4.12a is an example. The first block associates a sumnode to the variable on the subject position $?s$, and the second block to the variable on the object position $?o$, changing the name of projected variables appropriately for the object in the query of Figure 4.12a.

The sumedge materialisation is done on the line 11: the sumnode of the source node is retrieved thanks to the variable $?s$ that is projected by the sub-query in line 6; similarly for the target node thanks to the variable $?o$ returned by the sub-query line 16. On lines 9 and 19, we create a URI for the source and target sumnodes, respectively.

On line 22, we consider the case when there is no sumnode associated with the object variable. This can happen if the object is a literal, or if no feature can be extracted for the resource. Since we are concerned with the graph structure with the graph summary, we need to keep the type value. Therefore, we add the condition on line 25 which corresponds
to the line 4 in Algorithm 1. The empty string in the else clause represents the sumnode ∅ in the graph summary.

Remark. This works only if the ontology is not inside the processed dataset, since the type value would be lost by the sumnode URI creation.

Step 4
It is possible to assign some statistics about the summarisation when generating the sumnodes and sumedges. This requires again additional GROUP BY operations that are not depicted in the figure. However, we need then to reify the statistics in RDF since we use the CONSTRUCT operator in the query. This is further discussed in the Section 7.1.3.

Remark. If we use CONSTRUCT as the query form and we want to keep statistics about the summarisation, we need first to wrap the query into another one in which the aggregation is done. Then, we apply the CONSTRUCT query form on top. The reason is that that query form does not allow for aggregation operators such as COUNT in its clause.

MapReduce Implementation
In this section, we describe an implementation of the graph summarisation algorithm based on MapReduce [DG04], which is a batch-processing framework that eases distributed operations over massive amount of data. The unit of work in MapReduce, called a job, is composed of two operations: a mapper and a reducer. The mapper emits key-value pairs, and the reducer receives all the pairs associated to a same key.

Cascading. We developed the solution using Cascading\(^2\), which is a feature-rich API for defining and executing complex, scale-free, and fault-tolerant data processing workflows on a MapReduce cluster (e.g., Hadoop\(^3\)). The API lets the developer to quickly assemble complex processes without having to worry about the MapReduce paradigm. The Cascading model is based on the processing of “tuples”, which can be seen as database records, thanks to operators that filter, join, aggregate, ... In the algorithms to follow, we represent a tuple using the set notation \{...\}.

A Cascading flow is defined as a pipeline of such operators that connects data “sources” to “sinks” outputs. A flow can be composed of several “pipes”, which represent a set of operations. The Cascading flow forms a directed acyclic graph, which is then converted into a sequence of MapReduce jobs that can be executed on the cluster.

The Figure 4.13 depicts the Cascading flow of the graph summarisation. A node represent an operation over the incoming tuples, which if superscripted with GB involves an aggregate operation. An aggregate function combines multiple tuples into a single

\(^2\)http://www.cascading.org/projects/cascading/

\(^3\)http://hadoop.apache.org/
4.2. Graph Summary Generation

(a) Summarisation relation $R_t$

```sql
SELECT ?s (SHA1(GROUP_CONCAT(?t; separator = ",")) AS ?sID) {
    SELECT DISTINCT ?s ?t {
        ?s a ?t
    }
    ORDER BY ?t
}
GROUP BY ?s
```

(b) SPARQL query that generates the graph summary in the CONSTRUCT clause

Figure 4.12.: SPARQL-based graph summarisation over a single dataset
one; the function may be to count the tuples that were aggregated in order to add statistics about the graph to the summary for example. The triangle-shaped nodes represent points in the flow where the data is written to or read from.

**Dictionaries**

Read and write (I/O) operations on disk are a major cause of performance degradation of MapReduce jobs. In order to avoid moving the data across the MapReduce cluster, we compute a set of dictionaries that map a unique number to a vocabulary term, i.e., either an attribute or a type. We then use this unique number throughout the graph summary computation, decreasing the amount of data copied across the cluster, but also improving operations such as joins.

The Figure 4.13a depicts the Cascading flow for creating the dictionaries. The edges \((u, \alpha, v) \in A\) of the graph \(G\) are parsed, then a fork is done with regards to the attribute. We use the HFile [Mat] data structure as the dictionary backend. In our experiments, this marked a significant performance improvement compared to the Hadoop’s MapFile [Jon]. We use the DistributedCache\(^4\) in order to make the dictionaries accessible to all nodes of the MapReduce cluster.

The unique number a dictionary uses is computed using the MurmurHash3 [The, Hor] hash function. To further improve the performance of the summarisation, we hash the identifier of the nodes, i.e., the URI in the case of RDF data. A majority of the computation involves comparisons against these identifiers; thus using numbers instead of plain text improves shuffling operations in the MapReduce framework. In order to reduce potential hash collisions, we use the 128bit version of MurmurHash3.

Thanks to the dictionaries, the computation load over the two expensive join operations is decreased. Indeed, we only need to join data composed of unique numbers, without copying the plain text data across the MapReduce cluster which would increase the I/O. The entity description is needed only for the sumnode creation step. Once created, the computation can be abstracted from the actual data and use unique numbers instead. We use the dictionaries in the last step of the summarisation, which consist of the reification of the graph summary in RDF, intended for human consumption.

**Step 1**

The computation of a sumnode identifier requires all the information about an entity. Since the input data consist of an edge per tuple, we need to compute the entity description \(E\) which is depicted by the Step 1 in the Figure 4.13b. After parsing the input data, we aggregate the tuples based on the entity \(u\). Cascading provides the GroupBy operator to do so.

Depending on the summarisation relation, more or less information about an entity is needed. In the relations presented in this thesis such as the IO Attributes relation \(R_{ioa}\), attributes of incoming edges are required. We retrieve such information with no extra MapReduce job compared to others. The difference is in the amount of data aggregated.

The Algorithm 3 describes the mapper function of the entity description creation to

\(^4\)https://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/filecache/DistributedCache.html
4.2. Graph Summary Generation

which we add the incoming edges. The mapper function takes the edge as input, and emits a tuple containing three elements. The underlined element represents the key of the tuple. In the reducer function, we collect all the edges having the same hash key. If necessary, we can differentiate between incoming and outgoing edge thanks to the emitted flag.

Algorithm 3: Entity description expanded with incoming edges

Input: a tuple containing an edge \((u, \alpha, v) \in A\).
Output: a tuple containing (1) the hash value of the entity; (2) a flag indicating whether the edge is incoming; and (3) the edge.

// Underlined tuple elements define the key within the MapReduce framework
1 Function map\(\{ (u, \alpha, v) \}\):
2 emit\(\{ \text{hash}(u), 0, (u, \alpha, v) \}\)
3 emit\(\{ \text{hash}(v), 1, (v, \alpha, u) \}\)

Step 2
In order to generate unique identifiers for sumnodes, we extract features relevant to the summarisation relation from the entity description. For instance, we need the type values for the relation \(R_t\), and for \(R_a\) we are interested in the attributes. The projection operator as well as the handling of multi-valued feature are then performed programmatically, using a mapper.

The Algorithm 4 describes the implementation of the \(R_t\) relation, which stands as an example of Step 2 in the figure. We retrieve the type values associated with the node \(u\) on lines 2–5. The object invention is performed on line 6, where we emit the input tuple to which we add the identifier of the sumnode, i.e., the hash of the set of types.

Algorithm 4: Types summarisation relation \(R_t\)

Input: a tuple containing the hash value of a node \(u\), and the entity description \(\mathcal{E}(u)\) of that node.
Output: a tuple containing (1) the sumnode identifier; (2) the hash value of \(u\); and (3) the entity description \(\mathcal{E}(u)\) of that node.

1 Function map\(\{ \text{hash}(u), \mathcal{E}(u) \}\):
2 types ← []
3 foreach \((u, \alpha_i, v_i) \in \mathcal{E}(u)\) do
4   if \(\alpha_i = \tau\) then // \(\alpha_i\) is a type attribute
5     types[i] ← v_i
6   emit\(\{ \text{hash}(\text{types}), \text{hash}(u), \mathcal{E}(u) \}\)
Chapter 4. Graph Summary: a Novel Schema for Web Data

Step 3
We materialise the sumedges in two distinct processes, depicted by the Step 3a and Step 3b in the figure. The reason is to decrease the amount of data joined with the sumnode mappings from Step 2. To do so, we filter out in Step 3b the edges which target node is a sink — in RDF, this means to remove the edges which object is a literal. Also, we filter edges which have the type attribute. The reason is that the target node, which here is the type value, does not represent an entity. This is represented by the line 4 of the Algorithm 1.

In Step 3a, we create the sumedges which target sumnode is $\emptyset$. In Step 3b instead, we consider the sumedges which target might be defined — it may not be if the target node is not mapped to any sumnode (Step 2 in the figure). We remark that the Step 3a requires no join. Indeed, we keep the entity description along with the sumnode identifier as a result of the Step 2 processing.

Step 4
We end both Step 3a and Step 3b with a grouping operation. In Step 3a, we group all the tuples sharing the same sumnode identifier into a single tuple, computing some statistics if necessary, e.g., number of occurrences of an attribute, or the number of nodes mapped to the sumnode. In Step 3b, we group the tuples sharing both the source and target sumnodes as well as the attribute, thus forming the sumedges. Similarly, we may compute statistics about the sumedge in this step, e.g., the number of occurrences of a labelled edges connecting any nodes that were mapped to the adjacent sumnodes.

Discussion
Apart from the use of machine expensive operators such as ORDER BY or GROUP_CONCAT, we remark that the SPARQL query of the summarisation relation is duplicated two times. Therefore, this is an inefficient part of the query depicted in Figure 4.12b. Depending on the query planner of the SPARQL engine, this duplication might be spotted so to avoid computing twice the same pattern.

In comparison, the MapReduce-based one allows the summarisation to scale to much larger graphs. Indeed, in our experiments the SPARQL-based implementation is able to summarise graphs up to 20M edges only. A major obstacle to scale to larger graphs are the constraints imposed by SPARQL endpoints, e.g., limiting the number of rows returned by a SELECT query form, or query execution time-outs. Instead, the MapReduce implementation is able to scale to billions of edges, e.g., 2B with Freebase\textsuperscript{5} or even 30B with the Sindice dataset.

4.3. Conclusion and Future Work
In this chapter, we presented a generic framework for summarising graphs. It is generic in that different kinds of summarisation are possible. The framework allows to highlight

\textsuperscript{5}https://www.freebase.com/
4.3. Conclusion and Future Work

Figure 4.13.: Graph summarisation using the Cascading framework. A node represents an operation over the incoming tuples, which if superscripted with GB involves an aggregate operation, i.e., there is a GROUP BY operation. The * superscript over the RDF nodes indicates that the dictionary is used for mapping the data back into plain text. The triangle-shaped nodes represent points in the flow where the data is written to or read from.
the underlying structure of an entity graph. Since the produced summary is a graph homomorphic to the entity graph, it can be used instead of the original graph.

This property of the summary has a positive consequence: if a summary is smaller that its entity graph, using it instead of the entity graph positively impacts the performance of upstream applications. Indeed, one may leverage the summary to understand the structure of a graph so to pin-point elements of interest; then, the entity graph can be used to retrieve those specific elements.

As a future work, we plan to investigate how a summary may be updated pending changes on entity graph it was created from. This would improve the management of summaries, and keeping the information in the summary as fresh as possible.

In addition, we will study how to summarize a collection of datasets. This differs from the summarisation of a single dataset since we need then to consider the provenance of edges.
Chapter 5.
Graph Summary Precision

The summary of a graph reflects the structure of that graph. Concretely, this means that all possible paths in the graph are also possible in its summary. However, this does not imply the converse, i.e., that all possible paths in a summary do exist in the graph it represents. The structure of the summary stems from the grouping of the nodes in the graph by the summarisation relation. Therefore, the question of the precision of the summary with regards to the graph it represents is raised.

We judge the precision of a summary by the difference of traversing a summary or the graph it represents. Our rationale is the larger the difference, the less precise the summary is. We consider the precision of a summary with regards to the graph it represents from two directions. Firstly, we are concerned with the existence of paths in the graph. Secondly, we consider whether combinations of Attributes and/or Types in the summary do exist in the graph.

In this chapter, we present a model for measuring the precision of a summary with regards to the graph it was created from. The model rely on a summary that is set as the gold standard, to which we compare both directions, i.e., the existence of paths and of Attributes/Types combinations.

5.1. Graph Summarisation Lattice

We have introduced in Chapter 4 a framework for defining graph summaries. Each summary presented consider different properties of the graph for the summarisation relation, e.g., the Types summary $R_t$ considers the type information, while the Attributes summary $R_a$ the label of edges. Since the summarisation relations consider different aspect of the graph, the nodes are grouped differently. This produces summaries of varying structure. In this chapter, we investigate how the summaries differ, whether a summary is better suited to a certain type of application that another.

In order to answer these questions, we propose a Precision model which measures the amount of errors in a summary. This model relies on the ordering of the summaries. We present in this section a partial order relation on the summaries.

We denote with $\subseteq$ a binary relation on the set of summarisation relations. It relates two summarisation relations if one can be expressed as a composition of the other.
Chapter 5. Graph Summary Precision

![Diagram](G \xrightarrow{R_1} S_1 \xrightarrow{S} S_2 \xleftarrow{R_2})

Figure 5.1.: Summarisation relations $R_1$ and $R_2$ ordered with the binary relation ⊑ in Definition (5.1.1)

**Definition 5.1.1 (Relation ⊑)**

Let $G = \langle V, A, l_V \rangle$ be a graph, $S_1 = \langle W_1, B_1, l_{W_1} \rangle$ be the summary of $G$ according to $R_1 \subseteq V \times W_1$, and $S_2 = \langle W_2, B_2, l_{W_2} \rangle$ be the summary of $G$ according to $R_2 \subseteq V \times W_2$.

We say that $R_1$ precedes $R_2$, noted as $R_1 \preceq R_2$, if there exists a relation $S \subseteq W_1 \times W_2$ such that $R_2$ is a composition of $R_1$ and $S$ as follows:

$$R_2 = S \circ R_1 = \{ (x, z) \in V \times W_2 \mid \exists y \in W_1 \ (x, y) \in R_1 \land (y, z) \in S \}$$

A **Lattice** is a partially ordered set that contains an infimum and a supremum. The infimum is an element of the set that is “smaller”, according to the partial order, than all other elements. Conversely, a supremum is an element that is “larger” than any other element.

We use the binary relation ⊑ to order the summarisation relations based on their definition: two summarisation relations can be ordered if one can be expressed with the other. Figure 5.1 is a depiction of the binary relation ⊑ used for ordering summarisation relations. In this set, the infimum is the identity relation that maps the graph to itself, and the supremum is the relation that maps all nodes to a single sumnode.

**Theorem 5.1.1 (Graph Summarisation Lattice)**

The binary relation ⊑ is a Partial order. The set of summarisation relations with the partial order ⊑ forms a lattice that we denote as the graph summarisation lattice.

**Proof.** Let $G = \langle V, A, l_V \rangle$ be a graph, and for some set $A$, let $I_A = \{ (x, x) : x \in A \}$ be the identity relation on $A$.

**reflexivity:** Let $S = \langle W, B, l_W \rangle$ be the summary of $G$ according to $R \subseteq V \times W$. We have $R = I_W \circ R$, therefore $R \preceq R$.

**antisymmetry:** For $i \in \{1, 2\}$, let $S_i = \langle W_i, B_i, l_{W_i} \rangle$ be the summary of $G$ according to $R_i \subseteq V \times W_i$. If we suppose $R_1 \preceq R_2 \land R_2 \preceq R_1$, then $\exists S \subseteq W_1 \times W_2$ and $\exists T \subseteq W_2 \times W_1$ such that $R_2 = S \circ R_1$ and $R_1 = T \circ R_2$. By substitution, we have $R_2 = S \circ T \circ R_2$. Hence, $S \circ T = I_{W_2}$. Since the identity relation is a bijection, we have a one-to-one mapping between $W_1$ and $W_2$. Therefore, we have $R_1 = R_2$. 

70
5.1. Graph Summarisation Lattice

Figure 5.2.: Lattice of the summarisation relations set according to the partial order $\subseteq$

**transitivity:** For $i \in \{1, 2, 3\}$, let $S_i = \langle W_i, B_i, b_W \rangle$ be the summary of $G$ according to $R_i \subseteq V \times W_i$. If we suppose $R_1 \subseteq R_2 \subseteq R_3$, then $\exists S \subseteq W_1 \times W_2$ and $\exists T \subseteq W_2 \times W_3$ such that $R_2 = S \circ R_1$ and $R_3 = T \circ R_2$. Let $U \subseteq W_1 \times W_3$ be a relation defined as the composition of $S$ and $T$, i.e., $U = T \circ S$. Since the composition operation $\circ$ is associative, we have then $R_3 = U \circ R_1$. Therefore, $R_1 \subseteq R_3$.

Therefore, the binary relation $\subseteq$ is a partial order.

The set of summarisation relations can then be ordered according to $\subseteq$. The Figure 5.2 depicts the lattice that is formed by the partial order $\subseteq$ for the summarisation relations introduced in Section 4.1.3. For example, we remark that the Types summary $R_t$ is smaller than the IO Attributes & Types summary $R_{ioat}$, i.e., $R_{ioat} \subseteq R_t$.

**Remark.** The graph lattice depicts that all summaries presented in Section 4.1.3 can be generated from the IO Attributes & Types summary $R_{ioat}$. In cases where the IO Attributes & Types summary has significantly less edges and nodes than the entity graph, this allows a more efficient computation of all summaries which summarisation relation $R$ are smaller than the latter summary, i.e., $R_{ioat} \subseteq R$.

**Example.** The Figure 5.3 depicts the sumnodes built from two summarisation relations over the graph on Figure 2.2, i.e., the Types summarisation relation $R_t$ as dotted lines, and the Attributes & Types summarisation relation $R_{at}$ as dashed lines. The solid lines represent a relation $S \subseteq W_{at} \times W_t$, where $W_{at}$ is the set of sumnodes in the Attributes & Types summary, and $W_t$ the sumnodes in the Types summary. Using the binary relation $S$, we have the order $R_{at} \subseteq R_t$ between the two summarisation relations.

With the $R_{at}$ relation, the nodes $v_6$ and $v_7$ belong to different sumnodes, while they belong to the same with the $R_t$ relation. The same $R_t$ summary is built from the entity graph and from the $R_{at}$ summary. The partial order allows then to leverage pre-computed summaries in order to generate a new one.
Figure 5.3.: Sumnodes of summarisation relations $R_t$ and $R_{at}$ for the graph in Figure 2.2.

The dashed lines represent the $R_{at}$ relation, the dotted lines the $R_t$ relation, and the solid lines the relation derived from the partial order $\subseteq$, i.e., $R_{at} \subseteq R_t$. The nodes with superscript $t$ belong to the summary built from $R_t$, while the nodes with superscript $at$ belong to the summary built from $R_{at}$.

5.2. Summary Error

The confidence one may put on knowledge deduced from a summary has a more or less severe impact depending on the application. Also, the severity of an error depends on what information about the entity graph is affected by the error. In this section, we introduce a model for measuring the precision of a summary.

5.2.1. Error Model

The graph summary highlights the structure of the entity graph. Errors in the summary boil down to the presence of invalid edges. Indeed, a path or a combination of paths may exist in the summary $S$, but not in its entity graph $G$. This precision model accounts for the paths that exist in the summary but not in the entity graph.

As per the recursive definition of the bisimulation summarisation relation $R_{fbd}$ presented in Section 4.1.2, a node in the bisimulation summary $S_{fbd}$ refers to nodes of $G$ that share the same incoming and outgoing paths. Thus, the $S_{fbd}$ summary is the most precise summary for an entity graph with regards to the structure, i.e., all paths in the summary do exist in the original entity graph.

According to the partial order $\subseteq$, the bisimulation summary $S_{fbd}$ of a graph $G$ is “smaller” than any of the other presented summaries on $G$, i.e., for any summarisation relation $R$ we have $R_{fbd} \subseteq R$. Therefore, any edge on a summary can be inferred from the $S_{fbd}$ summary. However, the converse is not necessarily true.

We define the set $\text{Err}(R)$ as the set of inferred edges that are erroneous, i.e., the edges that do exist in a summary, but not in the entity graph. Since the summary $S_{fbd}$ is the most precise with regards to the structure of the entity graph, we use the summary $S_{fbd}$ instead of the entity graph $G$ to define the set $\text{Err}(R)$.
5.2. Summary Error

Definition 5.2.1 (Summary Error $\text{Err}(R)$)

Let $G = \langle V, A, l_V \rangle$ be a graph, $S = \langle W, B, l_W \rangle$ be the summary of $G$ according to $R \subseteq V \times W$, and $S_{\text{fbt}} = \langle W_{\text{fbt}}, B_{\text{fbt}}, l_{W_{\text{fbt}}} \rangle$ the bisimulation summary generated with the relation $R_{\text{fbt}}$. The set $\text{Err}(R) \not\subseteq B_{\text{fbt}}$ is the set of edges between nodes of $S_{\text{fbt}}$ that are inferred from $S$ as per the $\sqsubseteq$ relation, but that do not exist in the summary $S_{\text{fbt}}$.

$$\text{Err}(R) = \{(u, \alpha, v) \in W_{\text{fbt}} \times \mathcal{L} \times W_{\text{fbt}} \mid \exists (x, a) \in R \ (y, b) \in R : (a, \alpha, b) \in B$$

$$\wedge (x, u) \in R_{\text{fbt}} \ (y, v) \in R_{\text{fbt}} (u, \alpha, v) \not\in B_{\text{fbt}}\}$$

Remark. The set $\text{Err}(R)$ is equal to all possible combinations of nodes pair on $G$, keeping only the ones that exists in the summary $S$ but not in the bisimulation summary $S_{\text{fbt}}$.

Inferred Graph $\mathcal{I}(R)$

We introduce the graph $\mathcal{I}(R)$ as the bisimulation summary $S_{\text{fbt}}$ that is augmented with inferred edges from the $\text{Err}(R)$ set. This graph is used for computing the set of true and false positive edges in Section 5.2.3.

Definition 5.2.2 (Inferred Graph)

Let $G = \langle V, A, l_V \rangle$ be a graph, $S = \langle W, B, l_W \rangle$ be the summary of $G$ according to $R \subseteq V \times W$, and $S_{\text{fbt}} = \langle W_{\text{fbt}}, B_{\text{fbt}}, l_{W_{\text{fbt}}} \rangle$ be the bisimulation summary of $G$ according to $R_{\text{fbt}}$ such that $R_{\text{fbt}} \sqsubseteq R$.

We call the inferred graph $\mathcal{I}(R) = \langle W_{\text{fbt}}, C, l_{W_{\text{fbt}}} \rangle$ the graph which nodes and edges are those of the bisimulation summary $S_{\text{fbt}}$, augmented with the edges in $\text{Err}(R)$, i.e.,

$$C = B_{\text{fbt}} \cup \text{Err}(R)$$

The Figure 5.4 depicts the inferred graph $\mathcal{I}(R_t)$ according to the Types summary $S_t$ in Figure 4.3. Sink sumnodes are omitted for clarity. Sumnodes of the bisimulation summary $S_{\text{fbt}}$ are depicted with solid lines, and sumnodes of the Types summary $S_t$ with dashed lines. Edges from the $\text{Err}(R_t)$ set are represented with grey dotted arrows, while we depict sumnodes of the Types summary with dashed arrows.

Given that $R_{\text{fbt}} \sqsubseteq R_t$, the nodes $S_{1\text{fbt}}$ and $S_{2\text{fbt}}$ are mapped to the node $S_t^1$. Similarly, the nodes $S_{3\text{fbt}}$ and $S_{5\text{fbt}}$ are mapped to the node $S_t^3$. Since we have that $(S_t^1, \text{works}, S_t^3) \in B_t$, the edges $(S_{1\text{fbt}}, \text{works}, S_{5\text{fbt}}) \in \text{Err}(R_{\text{fbt}})$ and $(S_{2\text{fbt}}, \text{works}, S_{5\text{fbt}}) \in \text{Err}(R_{\text{fbt}})$ can be inferred, for example.

Edges from the $\text{Err}(R_t)$ set cause the nodes $S_{1\text{fbt}}$, $S_{5\text{fbt}}$, and $S_{6\text{fbt}}$ to be connected. This generates the path $\text{works.location.capital}$ that exists in the Types summary, but not in the bisimulation summary, and thus by extension, in the entity graph $G$. 

73
Chapter 5. Graph Summary Precision

Figure 5.4.: The inferred graph $\mathcal{I}(R_t)$ based on the summaries of Figures 4.3 and 4.5. Sink sumnodes are omitted for clarity. Sumnodes of the bisimulation summary $S_{fbt}$ are depicted with solid lines, and sumnodes of the Types summary $S_t$ with dashed lines. Edges from the $Err(R_t)$ set are represented with grey dotted arrows, and sumedges from the Types summary with dashed arrows.
5.2.2. Classification of Errors

Depending on the kind of edge in the set $\text{Err}(R)$, we identify three categories: connectivity, attribute, and type. The connectivity category reflects errors of a summary with regards to the structure of the entity graph, while the attribute and type categories with regards to its schema.

We illustrate the three categories in the following with regards to the Types $S_t$ and Attributes $S_a$ summaries only. Because the presented summarisation relations are bigger than $R_t$ and $R_a$ as per the partial order $\subseteq$, any error experienced with either $R_t$ or $R_a$ may also occur with the others.

**Connectivity Error** $\text{Err}(R)_{\text{con}}$

The connectivity error captures inferred paths that do not exist in the bisimulation summary $S_{\text{fbt}}$ but that do in the summary. For example, the Figure 5.4 depicts the inferred path creator from the node $S_{\text{fbt}}^0$ to $S_{\text{fbt}}^2$. We do not consider the sink sumnode $\emptyset$ in the connectivity error, since it doesn’t provide any outgoing edge. We define $\text{Err}(R)_{\text{con}}$ as the set $\text{Err}(R)$ minus the edges leading to the sink sumnodes.

**Definition 5.2.3 (Connectivity Error)**

Let $G = \langle V, A, l_V \rangle$ be a graph and $S = \langle W, B, l_W \rangle$ be its summary as per the summarisation relation $R \subseteq V \times W$. Let $\text{Err}(R)$ be the summary error of $S$. We define the connectivity error set $\text{Err}(R)_{\text{con}}$ as follows:

\[
\text{Err}(R)_{\text{con}} = \{ (x, \alpha, y) \in \text{Err}(R) \mid y \neq \emptyset \}
\]

**Attribute Error** $\text{Err}(R)_{\text{attr}}$

The attribute error captures edges from the summary error set $\text{Err}(R)$ that have an impact on the schema of the entity graph, due to additional Attributes inferred by the summary. For example, the node $S_t^4$ in Figure 4.3 implies the existence of a node in $G$ with edges capital and label, which is however not the case. We define $\text{Err}(R)_{\text{attr}}$ as the set that contains edges of $\text{Err}(R)$ in which the attribute is not in the set $\text{Attributes}(x)$ of the node $x$.

**Definition 5.2.4 (Attribute Error)**

Let $G = \langle V, A, l_V \rangle$ be a graph, $S = \langle W, B, l_W \rangle$ be the summary of $G$ according to $R \subseteq V \times W$, and $S_{\text{fbt}} = \langle W_{\text{fbt}}, B_{\text{fbt}}, l_{W_{\text{fbt}}} \rangle$ be the bisimulation summary of $G$ according to $R_{\text{fbt}}$. Let $\text{Err}(R)$ be the summary error of $S$. We define the attribute error set $\text{Err}(R)_{\text{attr}}$ as follows:

\[
\text{Err}(R)_{\text{attr}} = \{ (x, \alpha, y) \in \text{Err}(R) \mid \alpha \notin \text{Attributes}(x) \}
\]

where

\[
\text{Attributes}(x \in W_{\text{fbt}}) = \{ \alpha \mid \exists y \in W_{\text{fbt}}(x, \alpha, y) \in B_{\text{fbt}} \}
\]
Chapter 5. Graph Summary Precision

**Type Error** $\text{Err}(R)_{\text{type}}$

The type error captures edges from the summary error set $\text{Err}(R)$ that impact on the schema of the entity graph, due to additional types inferred from the summary. For example, in the Figure 4.7 the node $S_2$ contains the nodes $\{v_3, v_4, v_5\}$, since all three have the same set of Attributes, i.e., $\text{label}$, $\text{location}$ and $a$. Then, we may infer from that node the possible existence of a node in $G$ with types $\text{Place}$ and $\text{City}$, which is however not the case. We define $\text{Err}(R)_{\text{type}}$ as the set that contains additional types inferred from $\text{Err}(R)$, i.e., a type attribute $(x, \tau, y) \in \text{Err}(R)$, in which the target $y$ is not in the set $\text{types}(x)$ of the node $x$.

**Definition 5.2.5 (Type Error)**

Let $G = \langle V, A, l_V \rangle$ be a graph, $S = \langle \mathcal{W}, \mathcal{B}, l_{\mathcal{W}} \rangle$ be the summary of $G$ according to $R \subseteq V \times \mathcal{W}$, and $S_{fbt} = \langle \mathcal{W}_{fbt}, \mathcal{B}_{fbt}, l_{\mathcal{W}_{fbt}} \rangle$ be the bisimulation summary of $G$ according to $R_{fbt}$. Let $\text{Err}(R)$ be the summary error of $S$. We define the type error set $\text{Err}(R)_{\text{type}}$ as follows:

$$\text{Err}(R)_{\text{type}} = \{ (x, \tau, y) \in \text{Err}(R) \mid y \notin \text{types}(x) \}$$

where

$$\text{types}(x \in \mathcal{W}_{fbt}) = \{ \mathcal{W}_{fbt}(c) \mid \exists c \in \mathcal{W}_{fbt}(x, \tau, c) \in \mathcal{B}_{fbt} \}$$

5.2.3. Sumedge Precision

If we were to traverse the inferred graph $\mathcal{I}(R)$, we can follow sumedges of the $\mathcal{B}_{fbt}$ set from the bisimulation summary $S_{fbt}$, but also from the summary error set $\text{Err}(R)$. The former sumedges are *true positive* sumedges, while the latter are *false positive* sumedges.

Our rationale is that following sumedges from the first set can be reflected on the entity graph by following an instance of that summary path, hence it is a true positive sumedge. On the contrary, following a sumedge from the summary error set can not be reflected on the entity graph since there are no instances of that summary path that forms an existing edge.

We define the true positive sumedges for a given sumnode as its outgoing sumedges that belong to the bisimulation summary.

**Definition 5.2.6 (True Positive Sumedges)**

Let $G = \langle V, A, l_V \rangle$ be a graph and $S_{fbt} = \langle \mathcal{W}_{fbt}, \mathcal{B}_{fbt}, l_{\mathcal{W}_{fbt}} \rangle$ be the bisimulation summary of $G$ according to summarisation relation $R_{fbt}$. The set of true positive sumedges is defined as follows:

$$\text{TP}(x \in \mathcal{W}_{fbt}) = \{ (\alpha, y) \in \mathcal{L} \times \mathcal{W}_{fbt} \mid (x, \alpha, y) \in \mathcal{B}_{fbt} \}$$
5.3. Evaluation

We define the false positive sumedges for a given sumnode with regards to a summarisation relation $R$ as its outgoing sumedges that belong to the summary error set $\text{Err}(R)$.

**Definition 5.2.7 (False Positive Sumedges)**

Let $G = \langle V, A, l_v \rangle$ be a graph, $S = \langle W, B, l_w \rangle$ be the summary of $G$ according to $R \subseteq V \times W$, and $S_{fbt} = \langle W_{fbt}, B_{fbt}, l_{W_{fbt}} \rangle$ be the bisimulation summary of $G$ according to $R_{fbt}$.

$$FP(R, x \in W_{fbt}) = \{ (\alpha, y) \in L \times W_{fbt} \mid (x, \alpha, y) \in \text{Err}(R) \}$$

We define $\text{Prec}(R, x)$ the edge precision of a sumnode of the bisimulation summary $x \in W_{fbt}$ with regards to a summarisation relation $R$ as the proportion of the true positives over all the positive sumedges.

**Definition 5.2.8 (Sumedge Precision)**

Let $G = \langle V, A, l_v \rangle$ be a graph, $S = \langle W, B, l_w \rangle$ be the summary of $G$ according to $R \subseteq V \times W$, and $S_{fbt} = \langle W_{fbt}, B_{fbt}, l_{W_{fbt}} \rangle$ be the bisimulation summary of $G$ according to $R_{fbt}$.

$$\text{Prec}(R, x) = \frac{|TP(x)|}{|TP(x)\cup FP(R, x)|}$$

The sets $TP(x)$ and $FP(x)$ contain the true and false positive edges which source is $x$, respectively. For example, in Figure 5.4, the edge $(S_{fbt}^0, \text{creator}, S_{fbt}^1)$ is in $TP(x)$, since it does exist in the bisimulation summary $S_{fbt}$. The edge $(S_{fbt}^0, \text{creator}, S_{fbt}^2)$ is in $FP(x)$, since it does not exist in $S_{fbt}$. In total, this results that $\text{Prec} \left( R, S_{fbt}^0 \right) = \frac{3}{4}$.

The probability interpretation of $\text{Prec}(R, x)$ is the probability that a randomly selected edge is correctly summarised. We note the recall is always equal to 1, since there is no false negative edges in the presented summarisation relations.

We use $\text{Prec}(R, x)$ as the precision measure for each of the classification of errors. We note that for a same node, the edge precision may vary between the three categories. As an example, consider the node $S_{fbt}^0$ in Figure 5.4. The edge attribute precision is equal to 0, since the Attributes in both $S_{fbt}$ and $I(R_0)$ summaries for this node are $\text{creator}$ and $\text{author}$. However, the connectivity precision is equal to $\frac{3}{4}$.

5.3. Evaluation

In this section, we evaluate the graph summarisation under three different aspects:

- **Performance** the amount of computational power needed by the implementation of a summarisation relation for generating a summary;
- **Compression** the ability of the graph summarisation to reduce the number of nodes and edges needed for describing the graph’s structure; and
Chapter 5. Graph Summary Precision

**Precision** the precision of a generated graph summary with regards to the original graph, as measured by the **Precision model**.

We present those three dimensions of the evaluation in Section 5.3.1. In order to give ground to the study of the graph summarisation under those three dimensions, we consider a collection of 14 datasets described in Section 5.3.2 of varying size and complexity. The size of a graph to summarise will impact on the performance and compression of a summarisation implementation, while the complexity in terms of vocabulary (i.e., the use of Types and Attributes and combinations thereof) will manifest itself on the precision of a summary.

In this evaluation, we consider the following parameters:

- we want to summarise a graph \( G = \langle V, A, l_V \rangle \);
- the result of summarising the graph \( G \) is the summary graph \( S = \langle W, B, l_W \rangle \) according to the summarisation relation \( R \);
- with regards to the classification in Section 5.2.2, we measure the error in the created summary by comparing it to the bisimulation summary \( S_{fbt} \) of \( G \), i.e., \( S_{fbt} = \langle W_{fbt}, B_{fbt}, l_{W_{fbt}} \rangle \) according to the summarisation relation \( R_{fbt} \); and
- The bisimulation summarisation relation described in Section 4.1.2 maps nodes of the graph to a same sumnode if they all share the same outgoing and incoming paths. The bisimulation summarisation relation is smaller with regards to the partial order \( \sqsubseteq \) than any of the approximate graph summarisation presented in Section 4.1.3 since its summarisation relation is more restrictive. Therefore, we assume the existence of a relation \( S \subseteq W_{fbt} \times W \) that maps the bisimulation summary to a summary generated from the summarisation relation \( R \), i.e., such that \( R_{fbt} \sqsubseteq R \).

5.3.1. Design

We describe in this section the design of the evaluation. We first present the environment of our experimental framework. Then, we describe the dimensions of our evaluation.

Environment

The bisimulation summary \( S_{fbt} \) presented in Section 4.1.2 is the most precise summary of the entity graph, since all incoming and outgoing (and combination of) paths in the summary do exist in the entity graph \( G \). In this evaluation, we use as the **gold standard** summary of an entity graph the FT-Bisimulation summary instead of the Bisimulation summary for performance reason. This gold standard summary ensures that all outgoing paths do exist. The presented summarisation relations have been implemented using the Hadoop\(^1\) MapReduce framework according to the algorithm outlined in Section 4.2.2.

5.3. Evaluation

Evaluation Dimensions

We present in this section three dimensions we use for evaluating a summary, i.e., the summary compression, the computational performance of the summarisation relation, and the precision of the summary.

Summary compression. A summary is smaller if its size $|B|$ (number of sumedges) and its order $|W|$ (number of sumnodes) are inferior to the size $|A|$ and the order $|V|$ of the graph it was created from.

Apart from the Unique Type summary $R_{ut}$, the presented summarisation relations in Section 4.1.3 create a summary which is always smaller or equal to the graph, since there is a one-to-one mapping from the entity graph to the summary.

The unique type summarisation relation $R_{ut}$ may assign several sumnode to a same node $u \in V$ when the set of Types is greater than one, i.e., when $|\text{types}(u)| > 1$. The order of a Unique Type summary $S_{ut} = \langle W_{ut}, B_{ut}, b_{W_{ut}} \rangle$ is at most equal to the number of types in the graph, i.e., $|W_{ut}| = |\{ b_{W_{ut}}(c) \mid \forall (u, c) \in V^2, (u, \tau, c) \in B_{ut} \land \tau \in T \}|$. However, in the worst case, its size can equal $|B_{ut}| = |W_{ut}|^d$, with $d$ being the diameter$^2$ of the entity graph. Indeed, every node of the graph can be associated with every types. Therefore when applying the unique type summarisation relation $R_{ut}$ over such a graph, we need for each path in $G$ to create an edge to each type for every node of the path. Figure 5.5 depicts the size of the Unique Type summary in the worst case scenario, where all nodes are both of types $A$ and $B$.

The summary is sensitive to heterogeneous entity graph. Indeed, there can be as many sumnodes in an Attributes summary $S_a = \langle W_a, B_a, b_{W_a} \rangle$ as there is elements in the attribute powerset minus the empty set, i.e., $|W_a| = |\mathcal{P}(\{ \alpha \mid \forall (u, v) \in W_a, (u, \alpha, v) \in W_a \})| - 1$. Such an order of the summary can be expected for any summarisation relation based on the set of Attributes, i.e., the summarisation relations $R_{at}, R_{ioa}, R_{ioat}$, and $R_{fbt}$.

We measure the amount of information from a given graph $G_1 = \langle V_1, A_1, l_{V_1} \rangle$ that is compressed into another graph $G_2 = \langle V_2, A_2, l_{V_2} \rangle$. To this end, we compare the size and order of $G_2$ against the size and order of $G_1$. We report this comparison as the ratio $G_2 : G_1$:

$$G_2 : G_1 = \frac{|V_2| + |A_2|}{|V_1| + |A_1|}$$

Algorithm performance. We evaluate the computational performance of a summarisation relation by analysing the CPU time$^3$ on the Step 3 of the graph.

$^2$The diameter $d$ of a graph $G$ is the maximum distance between any two nodes of the graph. The distance between two nodes is the number of edges in the shortest path that connects the two.

$^3$The accumulated CPU time as reported with the CPU_MILLISECONDS counter of Hadoop.
Chapter 5. Graph Summary Precision

summary computation, as described in Section 6 for the MapReduce case. We
do not report on the Step 2 because the evaluated summarisation relations
have the same complexity, achieving similar times on this step.

Summary precision. With regards to all three classification of errors, we evaluate
the precision of a summary thanks to the true and false positive sumedges
set \( TP(x) \) and \( FP(x) \). We report the precision using two measures, i.e., \( P_1 \) and \( P_2 \).

The measure \( P_1 \) reflects the average number of true positive sumedges
from a randomly selected node in the inferred graph \( \mathcal{I}(R) \). In the equation
below of \( P_1 \), from right to left, we sum the sumedge precisions \( Prec(R, x) \) of a
node \( x \in \mathcal{W}_{fb} \), which we average by the number of nodes within \( C(c) \). Given
that there exists a relation \( S \subseteq \mathcal{W}_{fb} \times \mathcal{W} \) such that \( R_{fb} \sqsubseteq R \), the expression
\( C(c) \) is the set of sumnodes in \( \mathcal{W}_{fb} \) that are mapped to the sumnode \( c \in \mathcal{W} \)
as per the relation \( S \). Finally, we average over the total number of nodes
within the bisimulation summary.

\[
P_1 = \frac{1}{|\mathcal{W}_{fb}|} \times \sum_{c \in \mathcal{W}} \left( \frac{1}{|C(c)|} \times \sum_{x \in C(c)} Prec(R, x) \right)
\]

where \( C(c) = \{ x \in \mathcal{W}_{fb} \mid (x, c) \in S \} \)

The measure \( P_2 \) reflects the overall chance to randomly select a true posi-
tive sumedge in the inferred graph \( \mathcal{I}(R) \):

\[
P_2 = \frac{\sum_{x \in \mathcal{W}_{fb}} TP(x)}{\sum_{x \in \mathcal{W}_{fb}} TP(x) + FP(x)}
\]

5.3.2. Datasets

We use in this evaluation several datasets of varying complexity. We list in the Table 5.1
the datasets along with some descriptive statistics. The datasets are grouped according
to their complexity into four categories, i.e., Low, Medium, High, and Very High. We
determine the complexity with regards to the number of nodes and edges in a graph, as
well as with the number of unique types and attributes it possesses.

For each dataset, we present two aspects, i.e., the schema and the structure of the
entity graph. With regards to the schema complexity, we report the number of unique
edge labels and types of the entity graph. With regards to the structure complexity, we
report the size and order of \( G \) and \( S_{fb} \). The order values omit the number of sink nodes,
i.e., nodes without outgoing edges which includes literal nodes in the RDF data model.

The \( G : S_{fb} \) column reports the compression ratio of the graph \( G \) when the bisimulation
summarisation is used. We remark that the size and order of the bisimulation summary
\( S_{fb} \) is significantly smaller than of the entity graph. This emphasize the performance
benefits of using a summary instead of the entity graph itself in an application. We note
the lower the compression ratio, the more complex the graph is to summarise. Although
the datasets \texttt{bmb} and \texttt{wb} have a similar size and order, the compression ratio of the summary on \texttt{bmb} is greater than for \texttt{wb}, i.e., 85.43 to 13125.89. This shows that the dataset \texttt{bmb} has a more heterogeneous structure than that of \texttt{wb}.

### 5.3.3. Results

We evaluate and compare in this section the different graph summarisation relations according to the compression ratio, the computational complexity, and the precision of the generated summary. Then, we discuss the trade-offs with respect to these three dimensions. We note the mean of measurements in a category, i.e., \textit{Low}, \textit{Medium}, \textit{High}, and \textit{Very High}, as \(\mu_L\), \(\mu_M\), \(\mu_H\), and \(\mu_{VH}\), respectively.

#### Summary Compression

We depict in Figure 5.6 the average compression ratio in a log scale in each category of complexity for every summarisation relation. The raw data is reported in Table A.1, in the Appendix section.

The summaries based only on the type information, i.e., the Unique Type summary \(S_{ut}\) and the Types summary \(S_t\), exhibit a higher compression compared to that of the gold standard, i.e., \(G: S_{fud}\). The compression ratio is twice as much on the \textit{Low} and \textit{Medium} datasets and slightly higher on the more complex datasets. On average, the Types summary and Unique Type summary \(S_{ut}\) summaries present a similar compression ratio.

The attribute feature appears to be a better feature for summarising a graph than type. Indeed, the compression ratio of the Attributes summary \(S_a\) and IO Attributes summary \(S_{ioa}\) summaries remain stable across the dataset categories. However, this is
Table 5.1.: Descriptive statistics of the datasets. The Schema column reports the number in $G$ of unique edges labels and types. The column $G$ reports the order and the size of the entity graph. The column $S_{ft}$ reports the order and the size of the bisimulation summary $S_{ft}$ of the graph $G$. The $G : S_{ft}$ column reports the compression ratio of the graph $G$ when the bisimulation summarisation is used.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Schema</th>
<th>$G$</th>
<th>$S_{ft}$</th>
<th>$G : S_{ft}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Types</td>
<td>Attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Very High</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dbpedia [ht13]</td>
<td>288 524</td>
<td>22 369</td>
<td>65 042 837</td>
<td>233 051 608</td>
</tr>
<tr>
<td><strong>High</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>twc-logd [TWC13]</td>
<td>450</td>
<td>10 060</td>
<td>3 398 947</td>
<td>67 505 792</td>
</tr>
<tr>
<td>enipedia [EIS13]</td>
<td>128</td>
<td>267</td>
<td>4 155 20</td>
<td>4 463 566</td>
</tr>
<tr>
<td><strong>Medium</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3kat [BSL13]</td>
<td>20</td>
<td>30</td>
<td>85 795 956</td>
<td>592 778 746</td>
</tr>
<tr>
<td>ecs [Uni13]</td>
<td>24</td>
<td>120</td>
<td>167 390</td>
<td>955 112</td>
</tr>
<tr>
<td>lobid [hb13]</td>
<td>19</td>
<td>46</td>
<td>1 246 917 274</td>
<td>625 941 644</td>
</tr>
<tr>
<td>bnb [Ser13]</td>
<td>27</td>
<td>53</td>
<td>1 224 630 6</td>
<td>89 733 453</td>
</tr>
<tr>
<td>datos [OEG13]</td>
<td>23</td>
<td>143</td>
<td>7 412 312</td>
<td>58 048 932</td>
</tr>
<tr>
<td>gnd [NN13]</td>
<td>22</td>
<td>35</td>
<td>962 930</td>
<td>7 940 373</td>
</tr>
<tr>
<td>eures [Jen13b]</td>
<td>18</td>
<td>49</td>
<td>288 862</td>
<td>4 146 421</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>europeana [HI13]</td>
<td>5</td>
<td>58</td>
<td>5 559 452</td>
<td>40 773 834</td>
</tr>
<tr>
<td>wb [Cap13]</td>
<td>4</td>
<td>174</td>
<td>11 210 832</td>
<td>84 345 613</td>
</tr>
<tr>
<td>cordis [Jen13a]</td>
<td>7</td>
<td>63</td>
<td>729 780</td>
<td>7 101 623</td>
</tr>
<tr>
<td>ny-times [SL13]</td>
<td>2</td>
<td>38</td>
<td>22 662</td>
<td>345 888</td>
</tr>
</tbody>
</table>
5.3. Evaluation

Figure 5.6.: Compression ratio $G : S$ of graph summaries. The ratio axis is reported in logarithmic scale.

not the case with the summaries based only on the type, i.e., the $S_{ut}$ and Types summary summaries.

We can observe a correlation between the compression ratio and the number of types. We note that the Attributes summarisation $R_a$ actually achieves the lowest ratio on the very high dataset. This shows that the use of attributes in dbpedia is homogeneous across types. The compression ratio of the IO Attributes summary $S_{ioa}$ is on average on par with the ratio of the Attributes summary $S_a$, indicating a certain homogeneity of incoming edges in the entity graph.

By using both type and attribute information as with the Attributes & Types summarisation relation $R_{at}$, we remark that this can lead to a significant decrease of the summary’s compression, as reported on the Medium and High categories. Indeed, the combination of the features on the High datasets exhibits a much higher ratio than when taken separately. We can conclude that a sparse usage of attributes with a type creates an explosion of combinations. In general, the type feature is less stable than the attribute feature.

Performance of the Summarisation Relation

Figure 5.7 depicts the average CPU time in ms for each category of complexity. This time accounts for the Step 3 (Section 6) in the graph summarisation computation. For each dataset which raw data is reported in Table A.2, the times are the average of two runs of a summarisation relation.

The unique type $R_{ut}$ and types $R_t$ summarisation relations can have an undefined sumnode (Section 4.1.1) due to some nodes having no type information. For performance reason, the undefined sumnode $U$ is filtered from the resulting summary. Therefore, reported run times in Table A.2 from $R_{ut}$ and $R_t$ cannot be compared to others.

On the Medium, High, and Very High datasets, the time taken by the $R_{ut}$ summarisa-
Chapter 5. Graph Summary Precision

Figure 5.7: Performance comparison of the graph summaries

The summarisation relations based only on the type feature, i.e., $R_{ut}$ and $R_{t}$, provide a low connectivity precision. Indeed, they show on average a connectivity precision of $10^{-8}$.

Graph Summary Precision

We discuss in this section the precision of a summary with regards to the connectivity first, then to the type and attribute next. We do not report the precision in any error classification for the dbpedia dataset (Very High category). The reason is we were unable to evaluate the precision on it due to performance issues. While the performance evaluation did not account for the undefined sumnode $\mathcal{U}$, we do consider it for the precision evaluation.

Connectivity Precision

Figure 5.8 depicts the average connectivity precision for $P1$ and $P2$ over three categories of complexity, i.e., Low, Medium, and High. The raw results for each dataset are reported in Table A.3 of the Appendix section; for each category of dataset complexity, we report as well the mean $\mu$ of the connectivity precision.

We discuss first the connectivity error with regards to the precision $P1$ plotted on Figure 5.8a. The summarisation relations based only on the type feature, i.e., $R_{ut}$ and $R_{t}$, provide a low connectivity precision. Indeed, they show on average a connectivity precision of $10^{-8}$.

84
5.3. Evaluation

precision of 25% according to $P_1$, i.e., $\mu_H = 0.2414$ for the types summarisation relation $R_t$.

Summarisation based on the attribute feature only, i.e., the attributes summarisation relation $R_a$, provide also a low precision on Medium and High categories. On the Low category, the attribute feature exhibits a better precision than the type feature, i.e., $\mu_L = 0.5579$ for $R_a$ against $\mu_H = 0.3617$ for $R_{ut}$. However, when the type and attribute features are combined in the attributes $\$ types summarisation relation $R_{at}$, this provides a significant increase of the precision across all three categories. According to $P_1$, we reach on average a 50% connectivity precision (0.5124) on the High category for $R_{at}$, and at least 20% on Medium.

We remark that the incoming attribute in the IO attributes summarisation relation $R_{ioa}$ is an important feature that increases the precision. The $R_{ioa}$ summarisation provides a precision of 30% on the Medium up to 50% on High, whereas $R_a$ reaches 15% on Medium and 10% on High. Overall, we can achieve a good average connectivity precision with $R_{ioa}$ against $\sim_{ut}$. However, when we add the type feature.

We remark that the overall precision $P_2$ is very low on the datasets of Medium and High complexities. This suggests that few nodes of the summary have a high out-degree, creating a combinatorial explosion of false positive edges. This will be investigated in future work.

Schema Precision

Figure 5.9 depicts the average type and attribute precisions over the categories of complexity. In red we represent the attribute schema error, and in blue the type schema error. Bars with north east lines pattern depict $\mu_L$, with dotted pattern $\mu_M$, and with north west lines $\mu_H$. The raw data is reported in Table A.4 of the Appendix section.

The summarisation relations $R_{at}$ and $R_{t}$ based on the type feature provide an attribute precision above at least 60% for $P_1$ ($\mu_M = 0.5927$ for $\sim_{ut}$). On the contrary, the attribute feature reports a good type precision, reaching on average at least 90% for $R_a$, i.e., $P_1 = 0.9222$ on the Medium datasets.

Incoming attributes do not increase much the type precision, since the type precision of $R_a$ stays on par with $R_{ioa}$. The $R_{at}$ relation provides a perfect summarisation of the graph schema. Again, the significant differences between $P_1$ and $P_2$ suggests one more time that few nodes of the summary contains a high out-degree, creating a combinatorial explosion of false positive edges.

Discussion

In conclusion, we observe that a combination of both type and attribute features is necessary to achieve a good precision. The results show that taking Incoming attributes as a feature of the summarisation relation is important for the connectivity precision, but not for the schema.

However, there is place for improvement for overall connectivity precision especially on certain datasets. We observe that the precision $P_2$ leads to very low precision values which is caused by a few summary nodes with a high out-degree. This indicates that the
Figure 5.8.: Connectivity precision bar chart
Figure 5.9.: Schema precision bar chart. In red we show the attribute schema error, and in blue the type schema error. Bars with north east lines pattern depict $\mu_L$, with dotted pattern $\mu_M$, and with north west lines $\mu_H$. 
model of $P2$ is not appropriate for measuring the precision of a summary in terms of connectivity and schema.

**Trade-Offs**

We report in Figure 5.10a the trade-off between the average connectivity precision and the average compression ratio across all datasets among all the relations. We can distinguish two groups of relations, the type-based summarisation relations, i.e., the types $R_t$ and the unique type $R_{ut}$ summarisation relation, and the attribute-based ones, i.e., the attributes relation $R_a$, the attributes & types relation $R_{at}$, the IO attributes relation $R_{ioa}$, and the IO attributes & types relation $R_{ioat}$.

The type-based relations provide the best compression ratio, but also the worst precision. In the attribute group, the compression ratio among relations is close to each other, but their precision differs greatly, with $R_{ioat}$ ahead.

This suggests that in terms of trade-off between connectivity precision and compression ratio, $R_{ioat}$ is the best candidate.

We report in Figure 5.10b the trade-off between the average schema precision and the average compression ratio across all datasets among all the relations. Again we can distinguish the same two groups. However, in the attribute group, the precision does not differ as much among the candidates, each one being either equal or very close to 1. In the type group, the Types summarisation relation $R_t$ provides a reasonable precision for a good compression ratio.

This suggests that if the precision is primordial, the Attributes & Types relation $R_{at}$ is the best candidate, providing a perfect schema precision for the best compression ratio. However, if the compression is primordial instead and that some imperfection can be tolerated, then the Types summarisation relation $R_t$ is the best candidate.

We report in Figure 5.10c (respectively, 5.10d) the trade-off between the average connectivity precision (respectively, average schema precision) and the average CPU time across all datasets among all the summarisation relations. Among the attribute-based algorithms $R_a$, $R_{at}$, $R_{ioa}$ and $R_{ioat}$, the latter is the one that achieves the best runtime with the highest precision. Among the type-based algorithms, the Types summarisation $R_t$ achieves a lower runtime and a higher precision than the Unique Type relation $R_{ut}$.

If the schema precision is primordial and a low connectivity precision can be tolerated, the Types summarisation $R_t$ is the best candidate as it provides a high schema precision, with the best CPU time. On the contrary, if the connectivity is primordial, the summarisation relation $R_{ioat}$ is the best candidate, but this at the cost of a longer runtime.
5.3. Evaluation

Figure 5.10.: Efficiency and precision trade-offs of the candidate summarisation relations. The values are taken as the average across all dataset categories.
5.4. Conclusion and Future Work

In this chapter, we presented a technique for measuring the precision of a summary with regards to two aspects of the entity graph: the schema and the connectivity. We compare the summary to evaluate against a gold standard which allow us to compute the amount of false connections introduced by the summary.

We then analyse the trade-offs between the efficiency and the precision of the graph summarisation algorithms. We have performed the evaluation of the algorithms over 14 real-world datasets of varying size and complexity. The experimental results show that it is possible to approximate quite accurately the gold standard graph summary but with a much lower space and time complexity. The experimental results also provide initial evidence about the applicability of the algorithms in different context, e.g., schema summarisation or structure summarisation.

In future work, we will concentrate on finding features of the entity graph that can be used to improve the connectivity precision of the summary, while still being efficient to compute on a shared-nothing environment such as Hadoop.

Furthermore, we will investigate how the summary precision model may extended so it may indicate which part of a summary is more or less precise that another. This would allow a more comprehensive understanding of the entity graph a summary represents.
Chapter 6.

Directed Acyclic Graphs Ranking

The Web of Data can be seen as an encyclopaedia spanning over the entire Web. It contains rich information about concrete and abstract things covering a variety of domains, e.g., people, organisations, companies, media such as music or films, geographical places of interest. In Web Data, each of these is referred to as an entity to which is associated structured data, e.g., the name of a person or the description of a country.

The amount of entities on the Web that are associated with a structured description has reached a state where the search for specific entities can only be handled with systems such as web search engines. These systems are based on the Information Retrieval (IR) paradigm, where a set of documents matching a query is retrieved and ranked as per their relevance with regards to the query.

Traditionally, documents ranked by such systems contain little to no structure. This poses a challenges when applied to Web Data, since we have now data with a complex graph structure. Moreover, entities exhibit a high heterogeneity: the modelling of an entity can vary from datasets to datasets, even within a same domain; entities from a same domain can have a completely different set of descriptive attributes, or use attributes that don’t reflect the data. This calls for a novel approach in ranking graph structured data.

Current IR search engines are not adapted to rank entities that are, in addition to highly structured, also heterogeneous. In web search engines, field-based ranking models are popular for the ranking of structured documents such as HTML pages. However, such models have the following shortcomings:

1. the fields of the ranking model need to be known a priori;
2. a field with multiple values is not considered; and
3. the structure within a field cannot be modelled.

In this chapter, we investigate how the rich structure made available by the Web of Data can be leveraged to improve the ranking of entities. We introduce the “MF” ranking model as an extension to field-based approaches for ranking graph data. The underlying data structure of this model is a directed acyclic graph (DAG), which allows to support ranking data having many possible modeling. This novel model addresses the previously stated shortcomings:
Chapter 6. Directed Acyclic Graphs Ranking

1. the structure of the ranking model reflects the structure of the data;

2. fields can have an arbitrary number of values; and

3. the ranking model is a directed acyclic graph (DAG), allowing to support a vast number of data modelling.

6.1. Related Work

The Web of Data consists of a wide range of heterogeneous datasets, where the schema and the ontology can vary from one to the other. To overcome this diversity of attributes, different approaches for defining the Fields of an entity have been proposed.

From an RDF perspective, the authors consider in [PAAG+10] five weighted fields to represent the structure of an entity: literals (textual values), keywords extracted from the root’s label, i.e., the subject URI, keywords extracted from the incoming links, the entity’s Types and keywords extracted from object URIs. Compared to the BM25F and PL2F field-based approaches defined in 6.3, this approach is not able to grasp the rich structure of the data since attributes are discarded.

The BM25F and PL2F approaches we use in our experiments Section 6.6 are similar in nature to the BM25F adaptation proposed in [BMV11], where the authors consider one field per attribute in the data collection and can assign a different weight to each attribute. However, they restrict their approach to attributes with literal values, discarding those with URI values. In contrast to [BMV11], we consider attributes both with literal and URI values. URIs in an RDF graph carry relevant keywords, with regards to (1) the entity in general when considering the subject URI; (2) the attribute when considering the predicate URI; and (3) the related entity when considering the object URI.

However, all these approaches are an adaptation of the field-based ranking model in which multiple values associated to a same attribute are aggregated into a single value. This simplification of the underlying data model is inadequate for structured data. Therefore, we propose an extension of field-based ranking models in Section 6.4 to take into consideration multi-valued attributes and show that our model can be effectively applied to different ranking frameworks.

6.2. Ranking Model Foundations

We describe in this section vocabulary terms that will be reused throughout this chapter. We consider here a dataset to be a collection of entities, each entity being a graph. An entity may represent a person, a place, or an organisation. The unit of information which is retrieved and ranked is an entity.

We consider an entity to be a graph centered around a node, that we call the root, where each edge provides some description of that node. We define a rooted entity as
the entity description (Definition 2.3.3) of a root node, where every node in the entity description is reachable from that root node.

**Definition 6.2.1 (Rooted Entity)**

Let $G = (V, A, l_V)$ be a graph, and let $E(u \in V) \subseteq A$ be the entity description of the node $u$. We call the entity description $E(u)$ a rooted entity in $u$ if and only if there is a path from $u$ to every node in $E(u)$.

Figure 6.1 depicts four entities identified by the root nodes $me, _:b1, _:b2$ and paper/547.

We define here some vocabulary used in the rest of the chapter:

- **term**: a node label can be split over some character, e.g., a white space; we refer to a token resulting from this operation a term.

- **length**: the length denotes the number of terms, e.g., in a node.

### 6.3. Field-Based Ranking Models

In this section, we explain how to adapt two existing Field-based ranking frameworks to the graph model described in Chapter 2. A field-based ranking model views a graph as composed of multiple normalized weighted fields. For example, a field can be the title, the author or the body of the document.

In the Probabilistic Relevance Framework [RZ09] (PRF), BM25F [ZCT+04] is a field-based ranking function popular for web search. The Divergence From Randomness [AVR02] (DFR) framework gives birth to many ranking models, in particular to PL2F [MPH+05] which is of the field-based family.

In field-based ranking approaches, different parts of the entity description are arranged into several categories that we denote as fields. A field is then composed of field labels.
Chapter 6. Directed Acyclic Graphs Ranking

and/or edge labels. A score of the entity is computed to reflect how relevant the entity is with regards to a query; the score computation is dependent of the fields.

In presence of a multi-valued attribute, the common approach is to merge the content of all the values into one single value.

Ranking Features
The following features are used in the field-based ranking functions:

**field length** refers to the number of terms in a node label. In case of a multi-valued fields, it refers to the number of terms across all the nodes associated with the field.

**average field length** is equal to the mean of field length across entities.

Normalised Term Frequency
The ranking models presented here are from the TF-IDF family, where TF is a function of the term frequency, and IDF is a function of inverse document frequency.

The intuition behind TF is that it captures the relevancy of a term within a graph, e.g., the more a term occurs in a graph, the more important it is for that graph.

In contrary, the intuition behind IDF is to capture the importance of a term within the dataset: the more a term occur in the dataset, the less relevant it is. For example, the term “food” would occur frequently in a cooking recipes book, however it may not be an interesting feature in a ranking function.

In general, the term frequency is not used “as is” in the TF function: it is first normalised before applying the TF function. Term frequency normalisation is useful in cases where the length of a graph varies across the dataset: it allows to capture this variability into the ranking function, making the score of entities description comparable. In field-based ranking functions, the term frequency is normalized at the field level, capturing the length variability of a field across entities description.

Algorithm
Field-based ranking algorithms are constructed with the following basic operations. The importance of a query term in an entity is measured simply by how many times it occurs, i.e., its frequency.

**Step 1:** In a first step, each field of an entity is assigned a value that reflects its relevance with regards to a term in the query.

**Step 2:** In a second step, the values of each field are summed up and a function is applied on the result so to fit the model of the ranking algorithm.

**Step 3:** Finally, in a third step the relevance of the entity for each term in the query are summed up in order to compute the score of the entity, which represents how important the entity is with regards to the query.
6.3. Field-Based Ranking Models

6.3.1. BM25F Ranking Function

Using the field-based ranking algorithm BM25F [ZCT+04], we show in this section how to compute the importance \( \text{Score}(e, q) \) of an entity \( e \) for a query \( q \).

The first step consists in computing the normalised term frequency \( \tilde{f}_{t,e,a} \) with regards to the length of a field in the entity, compared to its average in the dataset.

\[
\tilde{f}_{t,e,a} = \frac{\alpha_a \times f_{t,e,a}}{1 + b_a \times \left( \frac{l_{a,e}}{l_a} - 1 \right)} \tag{6.1}
\]

where:

- \( t \) is a term in the query \( q \);
- \( f_{t,e,a} \) is the frequency of the term \( t \) in the field \( a \) of the entity \( e \);
- \( \alpha_a \) is a weight of the field \( a \);
- \( b_a \) is the normalisation parameter for the field \( a \) with \( b_a \in [0, 1] \);
- \( l_{e,a} \) is the field length of the attribute \( a \) in the entity \( e \); and
- \( l_a \) is the average field length of the attribute \( a \).

In a second step, we sum up the normalised term frequency from all fields, which result is passed to the saturation function. It is called the “saturation” function because it ensures that a term occurring many times in the entity does not out-weight the importance of other query terms.

\[
tfn_e = \frac{\sum_{a \in e} \tilde{f}_{t,e,a} \times (k_1 + 1)}{\sum_{a \in e} \tilde{f}_{t,e,a} + k_1} \tag{6.2}
\]

where \( k_1 \) is the saturation parameter.

In a third step, we combine the contribution of each query term, weighted by their importance in the dataset.

\[
\text{Score}(e, q) = \alpha_e \times \sum_{t \in q} q_t \times tfn_e \times \omega_t \tag{6.3}
\]

where:

- \( q_t \) is the weight of the query \( q \) for the term \( t \), i.e., its frequency within the query \( q \);
- \( \omega_t \) is the result of the IDF function for the term \( t \); and
- \( \alpha_e \) is a weight of the entity \( e \).
Chapter 6. Directed Acyclic Graphs Ranking

The IDF function is defined as

\[ \omega_t = 1 + \log \left( \frac{N}{v_t + 1} \right) \]

where \( N \) is the total number of entities in the dataset and \( v_t \) is the total number of entities that have occurrences of the term \( t \).

6.3.2. PL2F Ranking Function

The Divergence From Randomness (DFR) is a framework for creating ranking models. A model in the DFR framework are based on the combination of three components:

1. the information gain;
2. the randomness model; and
3. the term frequency normalisation model.

Using PL2F \[\text{MPH}^+05\] we show in this section how to compute the importance \( \text{Score}(e, q) \) of an entity \( e \) for a query \( q \).

In the first step, the term frequency normalisation model used is thenormalisation 2F. Given a query term \( t \) in the entity \( e \), this model is a per-field normalisation of the term frequency and is defined as follows:

\[
\bar{f}_{t,e,a} = \alpha_a \times f_{t,e,a} \times \log_2 \left( 1 + c_a \times \frac{l_a}{l_{e,a}} \right)
\]  

(6.4)

where:

- \( t \) is a term in the query \( q \);
- \( f_{t,e,a} \) is the frequency of the term \( t \) in the field \( a \) of the entity \( e \);
- \( \alpha_a \) is a weight of the field \( a \);
- \( c_a \) is a per-field parameter with \( c_a \in ]0, +\infty[ \);
- \( l_{e,a} \) is the field length of the attribute \( a \) in the entity \( e \); and
- \( l_a \) is the average field length of the attribute \( a \).

In the second step, we apply the other two components of PL2F over the sum \( \text{tf}n_e \) of the normalised term frequency across fields, i.e., \( \text{tf}n = \sum_{a\in e} \bar{f}_{t,e,a} \):

information gain: the Laplace after-effect model \( P_{\text{risk}} \) is used to estimate the information gain \( 1 - P_{\text{risk}} \):

\[ P_{\text{risk}} = \frac{\text{tf}n_e}{1 + \text{tf}n} \]  

(6.5)
6.4. MF Ranking Model

randomness model: the Poisson distribution $P_P$ is used to model the randomness:

$$P_P = \frac{\lambda^{tfn}}{tfn!} \times e^{-\lambda} \quad \text{where} \quad \lambda = \frac{F}{N}$$ (6.6)

where $F$ is equal to the frequency of the term $t$ in the dataset, and $N$ the number of entities in the dataset. The factorial is approximated with the Stirling’s formula:

$$tfn! = \sqrt{2\pi} \times tfn^{tfn+0.5} \times e^{-tfn}$$

The weight $w_{e,t}$ of the query term $t$ in the entity $e$ is computed by combining the three components:

$$w_{e,t} = (1 - P_{risk}) \times (\log_2 P_P)$$ (6.7)

In the third step, we combine the weight of each query term for the entity to compute the score $Score(e, q)$:

$$Score(e, q) = \alpha_e \times \sum_{t \in q} qtw \times w_{e,t}$$ (6.8)

where $qtw = \frac{q_t}{q_{t,\text{max}}}$ is the weight of the query $q$ for the term $t$ with $q_{t,\text{max}}$ the maximum of $q_t$ in $q$.

6.4. MF Ranking Model

In general, an entity contains some structure which provides relevant information about its content. For example, an entity that describes a book would be divided into chapters, each chapter into sections, each section into a set of paragraphs. In order to retain this structure for later use in the ranking, we generalize Field-based approaches with the “MF” ranking model for Directed Acyclic Graphs (DAG) by integrating the complete graph structure into its model.

First, we present the revised entity model and introduce the formal model of MF. Next we present two extensions to the MF model. Finally, we describe weights developed for the MF model.

6.4.1. Model

In traditional field-based ranking models, an entity is modelled as a set of fields, each being a bag of words. However in doing so, the rich structure of the entity is lost. For example, in Figure 6.2a where the graph of an entity rooted in $v_0$ is depicted, we can create three fields, one for each attribute $a$, $b$, and $c$; the label of nodes are then distributed as follows:

- $v_1$ is assigned to field $a$;
- $v_2$, $v_4$, and $v_5$ are assigned to field $b$; and
v_3 \text{ and } v_4 \text{ are assigned to the field } c.

However, a shortcoming is that the relationship between nodes is discarded within a field. Indeed, in traditional field-based ranking models, most of the entity’s structure is discarded.

Rather that representing the entity as a set of fields, the MF ranking model follows the structure of the graph. Each node of the graph — not a set of nodes — is represented as a bag of words. This allows to capture the structure of the graph into the ranking model.

Eliteness. In [Har74], Harter introduced the notion of eliteness in order to model content-bearing terms: a document is said to be elite in term \( t \) if it is somehow “about” the topic associated with \( t \). In [RvRP81], Robertson et al. introduce the relationship between the eliteness of a term in a document and its frequency: an elite term is most likely to be reused in the document, hence the term frequency is used as evidence of the term eliteness in the document.

In [ZCT+04], Zaragoza et al. extend the notion of eliteness to documents with multiple fields. In [RZT04], the authors argue that the normalized frequencies of a term in each field should be combined before applying the term weighting model.

Similarly in our MF model, the term eliteness in an entity is shared between its attributes. The values related to a same attribute are associated to a same topic, described by the edge label. Therefore, a term eliteness in an attribute is shared between its values. For nested entities that are more than a hop away from the root, the same interpretation applies.

In the following paragraphs, we describe how the term eliteness is garnered from every node of the entity description.

Expanded entity graph. Our model revolves around two core operations: on the one hand, the importance of an entity with regards to a query is computed on the leaves; on the other hand, internal nodes are used for aggregating the scores. In order to take into account (a) the label of internal nodes; and (b) the label of edges, we expand the graph so that each label becomes a leaf. To do so, we transform the entity graph into a graph where each label — edge and node — is a node, as depicted in Figure 6.2b. Then, each node of that transformed graph becomes a leaf of the tree, as depicted in Figure 6.2c; this is the expanded entity graph of the entity in Figure 6.2a.

Algorithm. The score of an entity with regards to a query is computed with a bottom-up approach. The leaves of the entity’s DAG contain information about the entity; a function that leverage the content and which returns a numerical value is then applied on that level. The internal nodes of the DAG aggregates the values computed on its outgoing nodes using a second function; the aggregation continues until the root of the entity is reached.
6.4. MF Ranking Model

(a) An entity description represented as a DAG. The root of the entity is the node $v_0$.

(b) The graph of the entity in Figure 6.2a where each label — of a node and an edge alike — becomes a node.

(c) The expanded graph of the entity in Figure 6.2a where each node of the graph in Figure 6.2b becomes a leaf. Nodes labelled with * depict aggregations.

Figure 6.2.: Depiction of the MF model
For instance, we compute the contribution of the leaves “d”, “v_5”, and “v_4”, which are aggregated on the parent node. The aggregated value is then combined with its sibling nodes, i.e., “b” and “v_2”. These operations are repeated until the root of the tree is reached.

**MF Model Normalisations.** Ranking models use statistics garnered from the entity to estimate its relevance with regards to a query. For instance, the frequency of a term can be used as an indicator for its importance in describing the entity. However, the entity is part of a dataset and the significance of the frequency as a measure varies between entities. This variability may be due to several factors, e.g., the writing style: two entities can provide the same amount of information about a certain topic, but one uses twice as much terms as the other.

In order to account for this variability in the dataset, there is a need for normalising [MRS08] the statistics garnered from the entity. For instance, in BM25F [ZCT+04] the frequency of a term within a field is normalised based on (a) the length of the field; and (b) the average length of the field within the dataset. Thanks to the normalisation, it is possible to compare the relevance with regards to a query of entities within a dataset.

With traditional field-based ranking models, there is one level of normalisation that is performed on the content level, as discussed in the previous paragraph. We can provide an additional normalisation, making two possible levels of normalisations with the MF ranking model:

1. a normalisation on the content level to account for the variability of a node’s length within a dataset; and
2. a normalisation of the outgoing degree of a node.

The first normalisation is applied on the leaf nodes of the DAG. The second normalisation is applied on the internal nodes during the aggregation step. This additional level of normalisation allows a finer tuning of the ranking function.

### 6.4.2. MF Ranking Functions

In this section, we describe $BM25MF$ and $PL2MF$, the MF extensions of BM25F [ZCT+04] and PL2F [MPH+05], respectively. We present first the features needed for the MF ranking model and then define both extensions.

**Ranking features.** The MF model introduces two levels of normalisations. In order to implement them, the following features are needed:

**Features for Normalisation 1.**

- **leaf length:** refers to the number of terms in the label of a leaf;
- **average leaf length:** is the mean of the leaf length of sibling nodes for a particular attribute;
6.4. MF Ranking Model

**Features for Normalisation 2.**

- **attribute cardinality:** is equal to the number of values an attribute possesses;
- **average attribute cardinality:** is equal to the mean of the attribute cardinality across the entities where that attribute appears.

**BM25MF**

BM25F is extended by adapting the Equation (6.1). The resulting normalised term frequency is then passed to the saturation function in the Equation (6.2). The computation of the entity score with BM25MF consists of applying two functions, $f_{t,L}$ and $f_{t,I}$, applied on a leaf and on an internal node, respectively.

The BM25MF ranking function is defined following the two levels of normalisations. The $f_{t,L}$ function implements the Normalisation 1 on the content and is defined as follows:

$$f_{t,L} = \frac{\alpha_v \times F_t}{1 + b_v \times \left( \frac{l}{l_a} - 1 \right)} \quad (6.9)$$

in which the following notations are used:
- $F_t$ is the frequency of the term $t$ within the leaf;
- $l$ is the leaf length of the leaf; and
- $l_a$ is the average leaf length.
- $\alpha_v$ is a weight specific to a leaf; and
- $b_v$ where $b_v \in [0, 1]$ is a parameter of the Normalisation 1.

After applying the $f_{t,L}$ function, results from the leaves are summed up on the parent nodes. Then, a parent node applies the function $f_{t,I}$ over the sum. This process is repeated until the root of the DAG is reached. The $f_{t,I}$ function implements the Normalisation 2 on the degree of an internal node. This function is defined as follows:

$$f_{t,I} = \frac{\alpha_a \times \sum f_{t,L}}{1 + b_a \times \left( \frac{|a|}{|a_e|} - 1 \right)} \quad (6.10)$$

in which the following notations are used:
- $|a_e|$ is the attribute cardinality of the attribute $a$ in the entity $e$;
- $|a|$ is the average attribute cardinality of the attribute $a$;
- $\alpha_a$ is a weight specific to an internal node; and
- $b_a$ where $b_a \in [0, 1]$ is a parameter of the Normalisation 2.
Chapter 6. Directed Acyclic Graphs Ranking

**Example.** If we consider the Figure 6.2c for instance with the following state:

- the node labelled $v_4$ is replaced with the sentence “The dog sleeps in his dog house.”;
- the node labelled $v_5$ is replaced with the sentence “This dog runs like the wind blows.”;
- the node labelled $d$ is replaced with “description”.

If we are given the query “dog”, the Equation (6.9) over the leaves $d$, $v_5$, and $v_4$ outputs the following:

**Node labelled $v_4$:** $f_{t,L} = \frac{2}{1 + \left(\frac{14}{14} - 1\right)} = 4$ with $\alpha_v = 1$, $b_v = 1$, $F_t = 2$, $l = 7$, and $l_a = 14$. Here, we assume that the average leaf length $l_a$ for the attribute $d$ is 14.

**Node labelled $v_5$:** $f_{t,L} = \frac{1}{1 + \left(\frac{14}{14} - 1\right)} = 2$ with $\alpha_v = 1$, $b_v = 1$, $F_t = 1$, $l = 7$, and $l_a = 14$.

**Node labelled $d$:** $f_{t,L} = 0$.

On the parent node, those three values are summed up and passed to the Equation (6.10). We get the result of $f_{t,I} = \frac{4 + 2}{1 + \left(\frac{14}{14} - 1\right)} = 4$ with $\alpha_a = b_a = 1$, $|a_c| = 3$, and $|a| = 2$; we assume that the average attribute cardinality for the attribute $d$ is 2.

The result of the $f_{t,I}$ function is then passed to the parent node; it is summed with the outputs of the $f_{t,L}$ function on the leaves $b$ and $v_2$. The same process is applied on each branch until the root entity is reached, where we finally assigns a score to the entity using Equation (6.2) that represents its relevance with regards to the query.

**PL2MF**

PL2F is extended by adapting the Equation (6.4). The resulting normalised term frequency is then passed to the normalisation function (6.5). Similarly to the BM25MF extension, the computation of the entity score with PL2MF consists of applying two functions, $f_{t,L}$ and $f_{t,I}$, applied on a leaf and on an internal node, respectively. We reuse the same notation as the ones from the previous section.

The PL2MF ranking function is defined following the two levels of normalisations. The notation used in the following two functions are similar to the one introduced in Section 6.4.2 for the BM25MF extension. The $f_{t,L}$ function implements the Normalisation 1 on the content and is defined as follows:

$$f_{t,L} = \alpha_v \times F_t \times \log_2 \left(1 + c_v \times \frac{l_a}{l}\right)$$  \hspace{1cm} (6.11)

where $c_v$ is a parameter particular to the PL2F ranking function applied on leaves, with $c_v \in [0, +\infty[$.
The \( f_{t,I} \) function implements the Normalisation 2 on the degree of an internal node. This function is defined as follows:

\[
f_{t,I} = \alpha_a \times \log_2 \left( 1 + c_a \times \frac{|a|}{|a|_{c_e}} \right) \times \sum f_{t,L}
\]  

(6.12)

where \( c_a \) is a parameter particular to the PL2F ranking function applied on internal nodes, with \( c_a \in ]0, +\infty[ \).

**Generalisation**

With both extensions in Equations (6.9) and (6.11), we normalize the term frequency \( F_t \) based on the average field length \( l_a \). In Equations (6.10) and (6.12), we further normalize the term frequency based on the average attribute cardinality \( |a| \). In addition to attribute-specific weights \( \alpha_a \), the MF ranking model allows value-specific weights in its implementations with the parameter \( \alpha_v \). We present value and attribute specific weights in the next section.

If we assume a single value per attribute to match field-based ranking models, then the Equations (6.9) and (6.10) are transformed into the Equation (6.2), with \( \alpha_a \times \alpha_v \) as the BM25F’s attribute weight, and \( b_v \) as the attribute normalisation parameter. BM25MF is under this condition equivalent to BM25F.

Under the same assumption, the Equations (6.11) and (6.12) are transformed into the Equation (6.4), with \( \alpha_a \times \alpha_v \times \log_2 (1 + c_a) \) as the PL2F’s attribute weight. PL2MF is under this condition equivalent to PL2F.

Therefore, the MF model is a generalisation of field-based models for directed acyclic graphs.

**6.5. Weights**

In this section, we introduce several weights for the MF model. Those weights are developed specifically for improving the task of “entity search”: given a query, retrieve the most relevant entity. We first present two query-dependent weights: (1) the Query Coverage weight which indicates how well the query terms are covered by an entity, an attribute or a value; and (2) the Leaf Coverage weight which indicates how well a value node is covered by a query. Next, we describe the Attribute and Entity Labels query-independent weights.

**6.5.1. The Query Coverage Weight**

The purpose of the Query Coverage (QC) weight is to lower the importance given to an entity, an attribute or a value with respect to the number of query terms it covers. This weight is combined with the ranking function using the parameters \( \alpha_e \), \( \alpha_a \) and \( \alpha_v \). For
example, given a query composed of three terms, if a value contains only occurrences of one query term, this value will then weight less than a value containing occurrences of more than one query term.

This weight integrates the IDF weight of query terms so that the coverage takes into account the importance of the terms it covers. For example, if two entities have occurrences of one of the three query terms, the coverage would then be \( \frac{1}{3} \) for both. Thanks to the IDF weights, the entity with the more important term will have a higher coverage weight than the other one. The query coverage weight is computed as follows:

\[
QC = \frac{\sum_{t \in X \cap q} \omega_t^2}{\sum_{t \in q} \omega_t^2}
\]

where \( X \) is either a value, an attribute set or the entity and \( q \) is the query.

### 6.5.2. The Leaf Coverage Weight

Values part of a multi-valued attribute share a common topic, described by said attribute. However, the relevancy of each value with regard to the query is different. We can assume that, given a same attribute, an entity where two terms occur in a single value is more relevant than another entity where each term occurs in two values. This can be seen as a way to integrate some kind of term proximity measure in the retrieval model, i.e., two words are considered close to each other when they occur in the same value. Reflecting this difference into the importance of a term using an appropriate value-specific weight can improve the ranking efficiency.

The Leaf Coverage (LC) weight reflects the proportion of terms in a value matching the query, i.e., how much it is covered by the query. Since a leaf in the MF ranking model holds some content about the entity, we assume that the more query terms match a large portion of the leaf, the more this leaf is a “precise” description of the query. This weight is integrated into the MF ranking function using the parameter \( \alpha_v \).

The leaf coverage is defined as the quotient of the query terms frequencies in the leaf over its leaf length:

\[
LC = \frac{\sum_{t \in v \cap q} F_t}{l}
\]

where \( q \) is the query, \( F_t \) is the frequency of the term \( t \) in the leaf \( v \), and \( l \) the leaf length. However, we remark that this definition disadvantages longer values over shorter ones in the case where the sum of term frequencies is equal. Indeed, given a query with two terms which occur \( x \) times in total within two leaves, the small leaf with occurrences of one term would receive a higher weight than the larger leaf with occurrences of the two; this is due to the division by the leaf length \( l \).

In order to have a better control over the effect of the LC weight, we developed a control function which (1) imposes a fixed lower bound \( \alpha \) to prevent short values receiving a
6.6. Experiments

In order to evaluate the MF model, we perform several experiments using three different datasets. We start by experimenting on the normalisation parameters in order to study their impact on the effectiveness of the approach. We then compare the MF ranking functions against other traditional ones. We finally discuss the consequence of not considering the attribute label as a source of potential relevant terms and demonstrate the

\[
\frac{\alpha}{1 + (\alpha - 1) \times LC^n}
\]

where \( \alpha \in ]0, 1[ \) is a parameter that sets the lower bound of LC, and \( n \) is a parameter that controls the effect of the coverage on the leaf. Figure 6.3 depicts effects of the two parameters \( \alpha \) and \( n \) over the leaf coverage LC. We remark that the higher \( n \) is, the higher the coverage needs to be for the leaf to receive a weight higher than \( \alpha \).

6.5.3. The Attribute and Entity Labels Weights

The Attribute and Entity Labels (AEL) weights balance the importance of an entity or an attribute depending on its label. This weight is integrated into the ranking function using the \( \alpha_a \) parameter. The weight value is defined empirically, and depends on the attribute. For instance, in a dataset concerned about books, a query match on the title might be considered more important than a match on the author’s description.

6.6. Experiments

In order to evaluate the MF model, we perform several experiments using three different datasets. We start by experimenting on the normalisation parameters in order to study their impact on the effectiveness of the approach. We then compare the MF ranking functions against other traditional ones. We finally discuss the consequence of not considering the attribute label as a source of potential relevant terms and demonstrate the
effectiveness of the proposed weights.

While there are other ways to perform entity ranking on the Web of Data, e.g., one can look at the other SemSearch 2011 candidates, in this evaluation we concentrate on demonstrating how the MF model specifically extends and improves the very popular PRF and DFR frameworks.

6.6.1. Datasets

The datasets we are using in our experiments are the following:

**INEX09** a dataset of 2,491,134 triples from DBpedia containing the description of entities in English, and converted for the INEX evaluation framework [PAAG+10];

**SS10** the “Billion Triple Challenge”¹ (BTC) dataset, containing more than one billion triples, with the assessments (a file containing relevance judgments on a sample of query answers) of the SemSearch2010² challenge;

**SS11** the BTC dataset with the assessments of the “Entity Search track” of the SemSearch2011³ challenge.

The INEX09 dataset is significantly different than the other two based on BTC. Indeed, BTC is a heterogeneous dataset, created from web crawls of several search engines. INEX09 is a highly curated dataset from DBpedia.

**Pre-processing.** In order to extract the entities from those datasets that are to be ranked, we performed the following operations:

1. extracted the **Entity descriptions** consisting of the outgoing edges and nodes that are 1-hop away;

2. analysed⁴ each description so that it may be retrieved based on a query that is pre-defined by the SemSearch⁵ challenges; and

3. ranked the retrieved entities according to a ranking model to evaluate. The quality of this ranking is the subject matter in the following sections.

---

¹Billion Triple Challenge: [http://vmlion25.deri.ie](http://vmlion25.deri.ie)
²SemSearch2010: [http://km.aifb.kit.edu/ws/semsearch10](http://km.aifb.kit.edu/ws/semsearch10)
³SemSearch2011: [http://km.aifb.kit.edu/ws/semsearch11/](http://km.aifb.kit.edu/ws/semsearch11/)
⁴The analysis of a text consists in splitting on a separator character such as whitespace, in order to produce tokens to be indexed. In the case of URIs, additional separators such as slashes ‘/’ and dots ‘.’ are also used. See [MHG10, BCC16] for further details.
⁵See footnotes 2 and 3
6.6.2. The Normalisation Parameters

In this section, we study the impact on the ranking of the normalisation parameters. In addition to the length normalisation of field-based ranking function (Normalisation 1), the MF ranking model offers an additional normalisation on the attribute’s cardinality (Normalisation 2). The effectiveness of the PRF and DFR frameworks depends on finding the right values for the normalisation parameters. However, these parameters are highly dependent on the dataset.

Figures 6.4a and 6.4b depict the impact of the normalisation parameters on the ranking performance of BM25MF and PL2MF, respectively. For all three datasets, the figures depict on the Y axis the Mean Average Precision (MAP) scores, and on the X axis the Normalisation 1, i.e., with parameter $b_v$ in Figure 6.4a (resp., parameter $c_v$ in Figure 6.4b). Each curve plots the results with a fixed Normalisation 2 parameter, i.e., $b_a$ in Figure 6.4a and $c_a$ in Figure 6.4b.

The grid of parameters values in Figure 6.4a ranges from 0 to 1 with a step of 0.1. In Figure 6.4b, the grid ranges from 0.5 to 10.5 with a step of 1. Although these parameters can be attribute and value-specific, this experiment considers a constant parameter in order to reduce the number of combinations and to lower the variability of the results. Dashed lines depict the MAP scores of BM25F and PL2F and solid lines the scores of their MF extension, BM25MF and PL2MF respectively.

These plots show that Normalisation 2, the normalisation introduced with the MF ranking model, improves the performance. Indeed, the Normalisation 1 parameters $b_v$ and $c_v$, which are traditional to the field-based ranking model, do not grasp completely the heterogeneity in the data, which results in lower performance when compared to the MF extensions. This indicates that the distinction between leaves within an attribute has a positive effect on the ranking.

6.6.3. Comparison between MF and Field-Based Models

In this section, we evaluate and compare the performance of the BM25 and PL2 ranking models against their MF extensions, BM25MF and PL2F respectively, and show the superiority of the MF model. The TF-IDF function is used as a baseline.

We outline below the ranking functions that are used in the experiments. We reuse the terminology for the equations that was presented in the previous section.

$TF-IDF$ is a logarithmic function of the term frequency and defines the Equation (6.3) as $tfn_e = \log(F_t) + 1$, where $F_t$ is the number of occurrences of the term $t$ in the entity.

$BM25 [RW94]$ considers the document as a simple bag of words. It is a function of the term frequency derived from a two-Poisson model and it uses an entity-length normalisation. The entity length is computed as the sum of the field length defined in the Section 6.3.
Chapter 6. Directed Acyclic Graphs Ranking

(a) Evaluation of BM25MF normalisation parameters. A curve plots a fixed $b_v$ value (Equation (6.9)) with $b_a$ (Equation (6.10)) varying from 0 to 1 with a precision step of 0.1.

(b) Evaluation of PL2MF normalisation parameters. A curve plots a fixed $c_v$ value (Equation (6.11)) with $c_a$ (Equation (6.12)) varying from 0.5 to 10.5 with a precision step of 1.

Figure 6.4.: Impact of the normalisation parameters on the MF ranking functions. The figures report the MAP values of the respective datasets on the Y axis. On the X axis and with each curve, we vary the values of normalisation parameters.
6.6. Experiments

<table>
<thead>
<tr>
<th></th>
<th>INEX09</th>
<th>SS10</th>
<th>SS11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25MF</td>
<td>b_a = 0.00, b_v = 0.75</td>
<td>b_a = 0.58, b_v = 0.75</td>
<td>b_a = 0.58, b_v = 0.75</td>
</tr>
<tr>
<td>BM25</td>
<td>b = 0.20</td>
<td>b = 0.20</td>
<td>b = 0.20</td>
</tr>
<tr>
<td>BM25F</td>
<td>b_a = 0.82</td>
<td>b_a = 0.82</td>
<td>b_a = 0.82</td>
</tr>
<tr>
<td>PL2MF</td>
<td>c_a = 9.19, c_v = 0.76</td>
<td>c_a = 1.52, c_v = 1.03</td>
<td>c_a = 1.79, c_v = 1.88</td>
</tr>
<tr>
<td>PL2</td>
<td>c = 17.01</td>
<td>c = 10.09</td>
<td>c = 10.09</td>
</tr>
<tr>
<td>PL2F</td>
<td>c_a = 1.87</td>
<td>c_a = 0.51</td>
<td>c_a = 1.51</td>
</tr>
</tbody>
</table>

Table 6.1.: Normalisation parameters values, found through a constrained particle swarm optimization.

It defines the Equation (6.3) as \( tfn_e = \frac{F_t \times (k_1 + 1)}{F_t + k_1 \times (1 + b \times \left( \frac{l_e}{l_{avg}} - 1 \right))} \), where \( l_e \) is the entity length of the entity \( e \), \( l_{avg} \) is the average of the entity length in the collection and \( b \) is a normalisation parameter.

**BM25F**

is defined in Equation (6.3). It considers documents as composed of fields, each field being a bag of words.

**PL2 [AVR02]**

considers the document as a simple bag of words. It is a model derived from the DFR framework, with the Equation (6.4) formulated as \( tfn_e = F_t \times \log_2 \left( 1 + c \times \frac{l_{avg}}{l_e} \right) \), where \( c \) is a normalisation parameter.

**PL2F**

is defined in Equation (6.8). It considers a document as a set of fields, each field being a bag of words.

The Table 6.1 reports the values of normalisation parameters for each ranking function. For a given function and dataset, the parameter value that maximises the Mean Average Precision (MAP) is found through a constrained particle swarm optimisation [HE02]. The optimisation is run on ranking functions that have all their weights equal to one, i.e., \( \alpha_e = \alpha_a = \alpha_v = 1 \). The reason is to limit what may impact the performance of a ranking function, thus invalidating the optimisation.

Using such parameters, we report in Figures 6.5a and 6.5b the performance of the ranking functions on the three datasets, for BM25MF and PL2MF respectively. The raw results of the bar charts are available in the Appendix section, in Table B.1.

Using the two-tailed Wilcoxon matched-pairs signed-ranks test [DJSH03, BCC10], the difference between a candidate ranking function and a MF extension is statistically significant at level 0.05 if the bar is displayed with a dot pattern; it is statistically significant at level 0.10 if displayed with an oblique lines pattern instead.

TF-IDF provides a clear-cut discrepancy between INEX09 and the datasets based on BTC, i.e., SS10 and SS11, the reason being it is not suited for heterogeneous datasets.

On SS10, BM25MF (resp., PL2MF) does not report a significant difference with BM25 (resp., PL2). On INEX09 and SS11, the MF extensions provide an increase of at least 10% in retrieval performance compared to BM25 and PL2.

On SS10 and SS11, the MF extensions provide better ranking performance with a significant difference than the field-based ranking functions with an increase of 15% at
Chapter 6. Directed Acyclic Graphs Ranking

Figure 6.5.: Comparison of state-of-the-art candidates against the MF generalizations. On the Y axis is reported the Mean Average Precision of the MF extensions and the other state-of-the-art candidates. Bars with a dot pattern indicate a statistically significant difference at level 0.05 compared to the MF extension; bars with an oblique lines pattern is at level 0.10 instead.

Overall, the experiments show that the MF model improves significantly the ranking effectiveness over traditional field-based methods. In addition, the experiments emphasize as well the fine tuning possible with the MF ranking model, allowing to better fit the ranking to a specific dataset.

6.6.4. Effectiveness of the Weights

In this section, we evaluate the impact of the presented weights over the MF ranking extensions. First, we discuss the impact of discarding the attribute label as a source of possible relevant terms on the ranking performance. Then, we evaluate the weights from Section 6.5 developed for the MF model.

Results of the evaluations are depicted in Figure 6.6. The PL2MF results are displayed on the left side of the figures, and on the right side the results of BM25MF. Using the two-tailed Wilcoxon matched-pairs signed-ranks test [DJSH03, BCC10], the MF extension with no added weight is used as the base of the comparison. Bars with a dot pattern indicate a statistically significant difference at level 0.05 compared to the MF extension; bars with an oblique lines pattern is at level 0.10 instead. Raw results are reported in Table B.2 of the Appendix section.

Impact of the Attribute Label

In this section, we investigate the consequence of not considering the attribute label as a source of potentially relevant terms, i.e., the attribute is not expanded into a leaf as
described in Section 6.4.

Figure 6.6a depicts the results, where we observe that removing the attribute label lowers the performance of the ranking with a statistical significance on INEX09 with BM25MF and PL2MF, and on SS11 with PL2MF only. This shows that the attribute labels can be a source of possible relevant terms.

**Query Coverage Weight**

In order to evaluate the benefit of the query coverage weight QC, we first analyse its effect separately when applied as an entity, an attribute or a value-specific weight. Then we study the consequence of applying it on all nodes at the same time.

We observe in Figure 6.6b that the QC weight improves the retrieval performance when applied on the attribute node, with a statistical significance on SS10 and SS11.

**Leaf Coverage Weight**

The evaluation of the leaf coverage weight LC investigates its efficiency with and without the control function (6.13), where we depict the results in Figure 6.6c.

We provide for each dataset the best performing parameters described in Section 6.5.2 for the control function. For the three datasets regardless of the MF extension, the values are as follows:

**INEX09:** $n = 1 \alpha = 0.7$;

**SS10:** $n = 2 \alpha = 0.4$; and

**SS11:** $n = 1 \alpha = 0.9$.

We can observe that the LC weight, with the control function (6.13) applied, improves slightly the retrieval performance on SS10 and SS11. The reason is that without this function, LC assigns a low weight to leaves containing many terms, even if they have occurrences of all query terms.

**Attribute and Entity Labels Weights**

For the weights that depend on the label of the entity or of an attribute, we use the following regular expressions to determine the value of the weight:

- 2 if the label matches ".*(label|name|title|sameas)";
- 0.5 if the label matches ".*(seealso|wikilinks)";
- 0.1 if the label matches "http://www.w3.org/1999/02/22-rdf-syntax-ns#_\d+"; and
- 1 otherwise.
Chapter 6. Directed Acyclic Graphs Ranking

For instance, if an attribute label is http://xmlns.com/foaf/0.1/name then a weight of 2 is assigned. The weight's values are set empirically, and were chosen in order to improve the task of entity search. The rationale is that a document where a query term occurs associated with an attribute matching "title" should be promoted.

The third regular expression matches an attribute URI defining items of a collection in RDF\(^6\). It is assigned a low weight of 0.1 to reduce the importance of terms occurring in each item of the collection.

We evaluate the Attribute and Entity Labels (AEL) weights first by considering the Attribute and the Entity Label weights separately, then both at the same time. Figure 6.6d depicts the results of applying such query-independent weights.

We note that the Attribute Label weight gives significant benefits to the ranking in SS10 and SS11, while it decreases the ranking performance in INEX09. This indicates that carefully defined weights for important and non-important attributes can contribute significantly to the effectiveness of the approach. We note also that the same can be seen with the Entity Label weight applied alone. The reason is similar to the Attribute Label weight.

Except for INEX09, using both weights at the same time increases the performance of MF ranking functions noticeably.

**Combination of Weights**

In this section, we investigate the retrieval performance when all four weights are used together. We depict results in Figure 6.6e, with the weights configuration

1. QC applied on the attribute node only;
2. LC with dataset-specific \(n\) and \(\alpha\) parameters for the control function; and
3. AEL weights.

The weights applied on a same node are combined by the multiplication of each weight value on that node. The attribute and entity label being leaves in the ranking model, we apply also the QC weight on those two in this experiment.

On INEX09 their combination decreases slightly the performance for PL2MF. On SS10 and SS11, although the QC and LC weights applied separately do not improve the effectiveness of the MF ranking functions by much, their combination with AEL increases the retrieval performance by at least 30% on SS10 and SS11.

6.7. Query-dependent Ranking of the Graph Summary

Graph summarisation is used for extracting the underlying structure of graph-shaped data. However, the size of a summary can still be an obstacle towards an effective browsing of the data.

\(^6\)RDF Container: http://www.w3.org/TR/rdf-schema/#ch_container
6.7. Query-dependent Ranking of the Graph Summary

(a) Without the attribute as a leaf

(b) Query coverage weight QC applied on the leaf, on the attribute, on the entity, and on all three levels

(c) Leaf coverage weight LC, tested with and without the control function in Equation (6.13)

(d) Entity and attribute labels weights

(e) All weights combined

Figure 6.6.: Evaluation of weights with the MF extensions. The PL2MF results are displayed on the left side of the figures, and the results of BM25MF are on the right side. Bars with a dot pattern indicate a statistically significant difference at level 0.05 compared to the MF extension; bars with a oblique lines pattern is at level 0.10 instead.
Yu and Yagadish propose in [YJ06] to “summarise” an existing database schema by showing only the important parts. This allows to get a succinct overview of the database. However, that work proposes an approach that is independent of the actual information need of a user.

In the case of a graph summary, we consider the information need of a user to be expressed as a graph-shaped query. Given a query over the summary, we propose to rank its solutions using the MF ranking model. In this way, the user will first browse subgraphs of the summary that are most relevant to his need. In this section, we evaluate the MF ranking model applied to a summary, showing the ability of the two levels of Normalisations in MF for fine-tuning of the ranking function.

6.7.1. Dataset

In order to evaluate the ranking on a graph summary, we use the USEWOD2013 dataset [LRBH13]. This dataset provides query logs from several SPARQL endpoints. In this evaluation, we focus on the logs from the DBpedia SPARQL endpoint. SPARQL [PS08] is the standard query language for RDF data. Given a graph-shaped query, it retrieves all subgraphs from a dataset which match its pattern.

Query Extraction

The graph summary highlights the structure of the graph it was created from. Therefore, we are only interested in queries that pertain to the structure of the graph. We present in this section a function that takes as input a SPARQL query, applies a set of rules over it, and outputs as result the transformed SPARQL query. The transformed query is then devoid of any specific details about entities.

Definition 6.7.1 (Query Extraction Function)

Let $I$ and $O$ be two SPARQL queries. Let $qe$ be a function that consists of the following rules:

1. retain only the basic graph patterns [PS08] (BGP);
2. transform CONSTRUCT and ASK queries [PS08] into wildcard select queries;
3. remove information specific to an entity; and
4. discard any solution modifiers.

The application of the function $qe$ on the query $I$ outputs the query $O$ that is independent of any entity.

In the following, we describe each operation of the function $qe$ and consider the query below as an ongoing example of the effect of an operation on the query:

1. **PREFIX dbr: <http://dbpedia.org/resource/>**
6.7. Query-dependent Ranking of the Graph Summary

&space;2
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

&space;3
ASK {
   dbr:Paris dbo:country "France" ;
   rdfs:label ?label .
   FILTER isLiteral(?label)
} ORDER BY ?label

**Operation 1.** A basic graph pattern (BGP) represents a set of triple patterns, which together form a graph pattern. A triple pattern represents an edge of a graph, where any of its components can be a variable.

**Definition 6.7.2 (Triple Pattern)**

Let \( G = \langle V, A, l_v \rangle \) be a graph, \( \mathcal{L} \) be the set of labels, and \( \text{Var} \) an infinite set of variables. A triple pattern is a tuple \( t = \langle s, p, o \rangle \) where \( t \in (\mathcal{L} \cup \text{Var}) \times (\mathcal{L} \cup \text{Var}) \times (\mathcal{L} \cup \text{Var}) \). The components of a triple pattern \( t \) are denoted subject\( (t) \), predicate\( (t) \) and object\( (t) \), respectively. An edge \( (x, \alpha, y) \in A \) of the graph \( G \) is a match for a triple pattern \( t \) if and only if the non-variable components of \( t \) are equal to those of the edge \( (x, \alpha, y) \).

In this operation, we discard any component of the SPARQL query that is not a BGP, e.g., filters. With regards to the ongoing example, the effect of this operation removes the FILTER pattern in line 10.

# Same set of prefixes
ASK {
   dbr:Paris dbo:country "France" ;
   rdfs:label ?label .
} ORDER BY ?label

**Operation 2.** A SPARQL query can be either (i) a CONSTRUCT, i.e., a query that allows to create a graph from a solution; (ii) a ASK, i.e., a query that returns a boolean value that is true if there was at least one match, and false otherwise; or (iii) a SELECT, i.e., a query that returns a table, where a row represents a solution for the query and a column one of its variables.

In this experiment, we disregard the kind of output requested for a query. Also, the variables that may have been specifically selected in a SELECT query are not important here. Therefore, we view all queries as wildcard SELECT queries, i.e., the binding of every variable in a solution is returned.
Chapter 6. Directed Acyclic Graphs Ranking

With regards to the ongoing example, this operation results in the query being transformed into a SELECT query.

```sparql
# Same set of prefixes
SELECT * {
  dbr:Paris dbo:country "France" ;
  rdfs:label ?label .
}
ORDER BY ?label
```

**Operation 3.** An entity can be a person, a place, an abstract concept, etc. For this experiment, we are interested in queries that pertain to structure of the graph. Therefore, we replace with a unique variable any detail a query might have about a specific entity.

If the detail is either a URI such as `<http://dbpedia.org/resource/Paris>`, a literal such as "Paris", then it is replaced with a unique variable. In the case where a literal is a type (Section 2.3.1), it is then kept as is.

With regards to the ongoing example, we replace in this operation the URIs dbr:Paris and dbr:Eiffel_Tower with the variables `?uri_1` and `?uri_2`, respectively. Also, the literal "France" is replace with the variable `lit_1`.

```sparql
# Same set of prefixes
SELECT * {
  ?uri_1 dbo:country ?lit_1 ;
  rdfs:label ?label .
}
ORDER BY ?label
```

**Operation 4.** In SPARQL, one can use solution modifiers in order to alter the result set of the query. For instance, the keyword `LIMIT` is used for limiting the number of results up to a specified value. As for previous operations, a solution modifier does not impact on the graph structure; therefore, those are removed from the extracted queries.

Finally, this operation removes the solution modifier `ORDER BY` from the ongoing example query.

```sparql
# Same set of prefixes
SELECT * {
  ?uri_1 dbo:country ?lit_1 ;
  rdfs:label ?label .
}
```
Query Classification

We classify a query according to the complexity of a pattern. The complexity is based on two features of a query, which are (i) the number of “star-shaped” patterns; and (ii) the size of a star-shaped pattern.

A star-shaped pattern — or star in short — is a BGP where the subject of all triple patterns is the same. The size of such a pattern is determined by the number of triple patterns it contains. For instance, the BGP “?s rdfs:label ?label . ?s foaf:name ?name .” is a star of size 2, since the subject of both triple patterns is “?s”.

Such a classification scheme allows to capture two aspects of a graph summary that have an impact on its precision introduced in Section 5.2.2. The size of a star in a query pertains to the attribute and type error. In addition, the number of stars in a query pertains to the connectivity error.

Observations of Queries in the Dataset

We process the queries from the logs in the USEWOD2013 dataset using the function $qe$ from Definition (6.7.1). Once the operations from the function $qe$ are performed, we have then a collection of queries that are only concerned about the structure of the graph queried.

Table 6.2 reports general statistics about the queries extracted from the USEWOD2013 DBpedia 3.3 logs using the function $qe$. The full list of queries is available in the Appendix B.2. For each category of complexity as defined in the previous section, we report in the table (a) the number of queries; (b) the number of unique terms where a term is a URI or a literal; and (c) the distribution of queries across the categories as a percentage.

The queries are grouped by its complexity, where a group is identified with the a string having the following regular expression: "\d+(-\d+)*". For instance, the string “2-1” identifies a group of queries that have two star-shaped patterns, one with two triple patterns and the other with only one.

Using the process outlined above, a total of 293 queries spanning over 17 categories were extracted. We note that the distribution of queries follows a power law. Indeed, most of the queries are contained in the categories “2”, “3”, and “4”; those represent BGPs with only a single star-shaped pattern. Only 52 queries (18%) actually contain more than one star. This suggests that a majority of the extracted queries are intended at retrieving entities.

6.7.2. Graph Summary Ranking

While creating a summary of a given graph, one may accumulate several statistics. We generate a summary with the following statistics:

1. the number of nodes mapped to a sumnode;
2. the number of occurrences of an attribute associated to a specific sumnode;
3. the number of times an attribute connects two sumnodes; and
Table 6.2.: Queries extracted from the USEWOD2013 query logs

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of Queries</th>
<th>Number of Unique Terms</th>
<th>Query Distribution (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1-4</td>
<td>2</td>
<td>9</td>
<td>0.68</td>
</tr>
<tr>
<td>1-1-3</td>
<td>2</td>
<td>6</td>
<td>0.68</td>
</tr>
<tr>
<td>1-1-2</td>
<td>1</td>
<td>3</td>
<td>0.34</td>
</tr>
<tr>
<td>3-4</td>
<td>1</td>
<td>6</td>
<td>0.34</td>
</tr>
<tr>
<td>2-2</td>
<td>2</td>
<td>5</td>
<td>0.68</td>
</tr>
<tr>
<td>1-5</td>
<td>1</td>
<td>6</td>
<td>0.34</td>
</tr>
<tr>
<td>1-4</td>
<td>1</td>
<td>5</td>
<td>0.34</td>
</tr>
<tr>
<td>1-3</td>
<td>1</td>
<td>4</td>
<td>0.34</td>
</tr>
<tr>
<td>1-2</td>
<td>19</td>
<td>30</td>
<td>6.48</td>
</tr>
<tr>
<td>1-1</td>
<td>22</td>
<td>28</td>
<td>7.51</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>10</td>
<td>0.34</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>0.34</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>33</td>
<td>2.39</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>24</td>
<td>2.39</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>49</td>
<td>8.87</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td>87</td>
<td>22.18</td>
</tr>
<tr>
<td>2</td>
<td>134</td>
<td>112</td>
<td>45.73</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>293</strong></td>
<td><strong>201</strong></td>
<td><strong>100.00</strong></td>
</tr>
</tbody>
</table>

4. the number of types in a sumnode.

Those statistics are used in the ranking function that we present below.

As an example, we consider the graph in Figure 6.7 that is a solution of the query from the previous section over a graph summary. A number between parenthesis indicates a statistic associated with the node. For instance, there is one node from the entity graph that got mapped to the sumnode $s_1$.

**Ranking functions.** We present the basic and MF-based approaches, two ranking functions that can be applied over a graph summary.

**Basic approach.** A straightforward approach consists in summing up all the statistics available to a subgraph that matched a query. The rationale is that the greater the sum, the more relevant is that solution. The result is 5 in Figure 6.7.

**MF-based approach.** In this approach, we apply both normalisations available with the MF ranking model, i.e., Normalisations 1 and 2. The difference of this approach with previous is that the average cardinality of every node is considered.

### 6.7.3. Evaluation

In Section 6.7.1 we presented a dataset of queries that were extracted from logs of the DBpedia SPARQL endpoint. In Section 6.7.3 we introduced two ranking functions that we apply on sub-graphs of a graph summary which is, in this evaluation, that of DBpedia. In this experiment, we evaluate the performance of the MF-based approach at ranking subgraphs of a graph summary.
6.7. Query-dependent Ranking of the Graph Summary

Figure 6.7.: A solution to the SPARQL query of the running example. A number in parenthesis represents a statistic associated with that node.

We present first the ranking paradigm that we follow, before describing the experiment itself. Then, we describe the evaluation method for asserting the performance of the MF extension. Finally, we discuss the results of the experiment.

**Ranking paradigm.** Given a SPARQL query, its solutions retrieved from a graph summary can be erroneous as expressed in Chapter 5; indeed, a summary path from a solution might have no instance that form a path in the entity graph as well. A good solution is then a subgraph of the summary that does exist in the entity graph. Therefore, a ranking algorithm should rank first such solutions.

**Experimental setting.** Due to imprecisions in a summary as discussed in Chapter 5, a query can return solutions that do not have any actual summary instance in the entity graph. We present the following experimental setting in order to evaluate the performance of the MF extension with regards to the ranking paradigm.

Given a SPARQL query extracted from the USEWOD2013 query logs of DBpedia, we replace with a variable that we denote as “?POF” a term of the query, one at a time. A term is either a predicate or an object of a triple pattern. The resulting query is then executed over a graph summary of DBpedia.

In parallel, we run the query over the entity graph; this allows us to compare the bindings of the ?POF variables as returned from both graphs, i.e., the entity graph and its summary. The difference between the two sets of bindings is an indication of the summary’s precision for that query.

We keep only the queries that have a solution over the DBpedia dataset; this ensures that any invalid binding of the ?POF variable is due to the summary.
Example. Consider the SPARQL query taken as an example in the previous section. We replace the term rdfs:label with the ?POF variable. Let us consider as well that the terms rdfs:label and rdfs:comment are bindings for that variable as retrieved from the entity graph; however, bindings retrieved from the summary are rdfs:label, rdfs:comment and dc:title. The term dc:title is then invalid.

**Ranking performance measure.** We consider two aspects of the ranking in this evaluation:

1. whether a solution that provides bindings for the ?POF variable that are part of the error set (Section 5.2.2) is ranked lower than one that does not; and

2. the more a solution returns elements from the error set, the lower ranked it is.

In order to achieve this, we use the Mean Average Precision [MRS08] (MAP) as the measure of ranking performance. The average solution metric considers the order of a solution in the rank list, in addition to whether a solution is relevant or not. The MAP measure averages that metric over all the queries that were executed.

**Discussion.** We compare in this paragraph the performance of both ranking functions when applied over the Types summary of DBpedia. First, we evaluate the impact of the normalisation parameters on the MF-based ranking approach. Then, we discuss the ranking performance of the basic and MF-based approaches.

**Normalisation parameters impact.** Figures 6.8 depict the impact of the Normalisation 1 and 2 parameters on the ranking of graphs. On the X axis we vary the value of the parameter \( b_a \) for Normalisation 2. Each curve corresponds to a value of the parameter \( b_v \) for Normalisation 1. On the Y axis we report the MAP measurement for the MF-based ranking approach. The raw results of the curves are available in Appendix B.7.

In this experiment, we divided the queries into two categories: simple graphs and complex graphs. Simple graphs indicate queries that have only one star, i.e., the queries from the complexity categories 10, 9, 6, 5, 4, 3, and 2. Complex graphs represents queries that contain more than one star, i.e., the queries with complexity categories 1-1-4, 1-1-3, 1-1-2, 3-4, 2-2, 1-5, 1-4, 1-3, 1-2, and 1-1.

Figure 6.8a depicts the average of the MAP measurements from the seven queries classes of the simple category. Figure 6.8b depicts the average of the MAP measurements from the ten queries classes within the complex category.

We observe that with simple graphs in Figure 6.8a the parameter \( b_a \) for Normalisation 2 reported on the X axis has little effect on the ranking. On the contrary, we see it has a positive impact in Figure 6.8b for complex graphs. The parameter \( b_v \) for Normalisation 1 increases the performance of the ranking as its value reaches 1. This shows the normalisation parameters introduced with the MF ranking model allow a fine tuning of the ranking to the data at hand.
Basic and MF-based approaches comparison. Given the outcome of the previous experiment, we set to 1.0 the parameters of the Normalisations in the MF-based approach for the comparison between MF-based and basic ranking approaches of the graph summary.

Table 6.3 reports the MAP values of the basic and MF-based approaches over the Types summary of Dbpedia. We write the MAP value in bold for the approach that has the greatest. We observe that for a majority of the query complexities, the MF-based approach provides a better ranking than the basic approach.

This shows the benefits of the MF ranking model of normalising the data over two levels: the Normalisation 1 allows to compare several solutions to a query regardless of different statistics they might have; with the Normalisation 2, it is possible to rank solutions to a query that have different degree of a same attribute.

6.8. Conclusion and Future Work

In this chapter, we introduced “MF”, which is a novel ranking model for semi-structured data. We showed that MF is a generalisation of Field-based ranking models to a DAG model. This allows to apply the MF ranking model to a variety of data modelling: tree-shaped data as discussed in Section 6.6, or graph-shaped data as presented in Section 6.7.3.

The MF ranking model introduces two levels of normalisations; one over the length of a node, the other over the degree of a node. The latter is a new normalisation level when compared to field-based ranking models. Through a variety of experiments, these two levels have proved to increase the ranking performance significantly, since they allow a finer tuning of the ranking with regards to the data.
## Chapter 6. Directed Acyclic Graphs Ranking

<table>
<thead>
<tr>
<th></th>
<th>Basic Approach</th>
<th>MF-based Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1-4</td>
<td>0.8096</td>
<td>0.8102</td>
</tr>
<tr>
<td>1-1-3</td>
<td>0.8692</td>
<td>0.8698</td>
</tr>
<tr>
<td>1-1-2</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>3-4</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2-2</td>
<td>0.7503</td>
<td>0.7870</td>
</tr>
<tr>
<td>1-5</td>
<td>0.9784</td>
<td>0.9791</td>
</tr>
<tr>
<td>1-4</td>
<td>0.8793</td>
<td>0.8867</td>
</tr>
<tr>
<td>1-3</td>
<td>0.6730</td>
<td>0.6744</td>
</tr>
<tr>
<td>1-2</td>
<td>0.8795</td>
<td>0.8977</td>
</tr>
<tr>
<td>1-1</td>
<td>0.9381</td>
<td>0.9447</td>
</tr>
<tr>
<td>10</td>
<td>0.9351</td>
<td>0.9439</td>
</tr>
<tr>
<td>9</td>
<td><strong>0.9344</strong></td>
<td>0.9433</td>
</tr>
<tr>
<td>6</td>
<td>0.7877</td>
<td>0.8699</td>
</tr>
<tr>
<td>5</td>
<td>0.8475</td>
<td>0.9111</td>
</tr>
<tr>
<td>4</td>
<td>0.8685</td>
<td>0.9089</td>
</tr>
<tr>
<td>3</td>
<td>0.8763</td>
<td>0.9154</td>
</tr>
<tr>
<td>2</td>
<td>0.9320</td>
<td><strong>0.9389</strong></td>
</tr>
</tbody>
</table>

Table 6.3.: Comparison of the basic and MF-based ranking approaches on the Types summary of DBpedia 3.3
We have shown in the evaluations that MF ranking normalisation parameters are highly dependent on the datasets. MF ranking on a highly curated dataset requires different parameters than on a heterogeneous one. As future work, we will then pursue two directions of research. First, a dynamic approach for finding such parameters would improve the performance of the approach. Second, we will investigate the possibility of integrating statistics from a graph summary into the ranking, e.g., as a weight for an edge.
Part III.

Putting the Pieces Together
Chapter 7.

Graph Summary Applications

Graph summarisation allows to generate a much smaller structural replica of an entity graph. Therefore, it can benefit applications that require some knowledge about the structure of the graph. We present in this chapter some applications that are made possible thanks to the graph summary introduced in Section 4.

We introduce in Section 7.1 an RDF model for the output of a graph summarisation process. The existence of such a model enables querying the summary in order to inspect the entity structure of the graph. Combined with the entity graph, the RDFication of the summary ultimately holds the function of a schema for the graph. In Section 7.2 we present an application that leverage the RDF model in order to provide RDF terms recommendation to a user writing a SPARQL query. The graph summary is also used in the Web Data Inspector described in Section 7.3 as a mean for inspecting edges in a dataset and relationships between datasets.

7.1. Graph Summary for Web Data Management

The graph summary highlights the structure of a graph, which in the case of RDF data consists in the use of predicates, classes, and their relationships. As such, a graph summary exhibits a similar purpose as a schema in relational database management systems. In comparison to the VoID [ACHZ09] approach, the use of a graph summary as a schema provides additional information, e.g., which predicates co-occur, or how are the classes in a dataset inter-connected.

We define a graph schema model on top of a graph summary in Section 7.1.1. We describe an RDF model for a graph summary in Section 7.1.3.

7.1.1. Graph Schema

In order to use the information in a graph effectively, it is necessary to have a schema of that graph. The schema provides a description of the structure of the entity graph. Given a graph schema [BDFS97, Kun83], it is then possible to understand how the data is structured so that it is possible to perform meaningful operations such as querying, optimising query execution, or partitioning data.

The schema of the entity graph is itself a graph as in [Kun83], where a model for defining integrity constraints and similar rules as in a traditional relational database
Chapter 7. Graph Summary Applications

Figure 7.1.: Trade-offs between graph schema creation approaches.

schema is proposed. In this work, we focus on the structural description of the graph.

7.1.2. Overview

Knowledge is commonly represented using directed graphs, as it is expressive enough for modelling data with complex relationships. For example, semantic networks [Qui66] have been used in artificial intelligence and machine translation, with the Web of Data being a large scale instance.

Similarly to schemas in databases, there exists graph schemas which defines the structure of the graph. A graph schema can be pre-defined as in a top-down approach, or can be generated from the data itself in a bottom-up approach. Thanks to a graph schema, we suspect that the potential benefits are numerous: optimisation of query processing, data integration, data discovery, etc.

There exists two approaches for creating a graph schema, each sitting at opposite ends as depicted by Figure 7.1. On the right side, we have the top-down approaches where a person creates the schema manually. Such a hand-crafted schema ensures the data to be conform with it, but at the cost of less flexibility with regards to the evolution of the schema in time and its customisation to specific needs.

On the left side, we have the bottom-up approaches which rely only on the data. Such schemas offer more flexibility when manipulating the data, but they offer more heterogeneous information. Indeed, a hand-crafted schema is the result of a careful thought-process, while a generated (bottom-up approach) schema is susceptible to data heterogeneity.

The two approaches highlight a time dimension to the creation of a schema: in the top-down approach, the schema is first created and then the data is produced; in the bottom-up approach instead, this is the opposite: the data is first produced and then a schema is created from it.

Top-Down Graph Schema

The structure of the graph can be rigorously defined. In [Kun83] the schema of the database is itself represented as a graph, which facilitates the construction of the database as well as the query formulation. In [AVH04] the authors present the Web Ontology Language which represents the ontology layer in the Semantic Web stack. It allows to define classes, properties, and relationships between classes. Regarding XML data, a
graph schema can be expressed via XML Schemas\(^1\). In these methods, the graph data either has to be squeezed to fit the pre-defined schema, or the schema has to be updated to meet the new structure of the underlying data. Updating the schema is an expensive operation, since the graph schemas are generally complex.

**Bottom-Up Graph Schema**

In a bottom-up approach, the schema is generated from the data itself by using some features of the data, e.g., Attributes of a node. In order to generate a schema, the process must then preserve the structure of the graph. Buneman et. al. [BDFS97] defines that a graph dataset conforms to a graph schema if there exists a simulation from the dataset to the graph schema. Informally, whenever there is an edge in the dataset, there is a corresponding edge in the graph schema. This can be achieved by using a summarisation relation. Indeed, a graph summary is defined as a graph homomorphism in Section 4.1.1. As such, we can use a graph summary to fulfill the function of a schema.

**Graph Schema Granularity**

There is a need for schemas of varying granularity. It is common for database schemas to be large and therefore difficult for users to understand. The authors in [YJ06, YPS11] proposed methods for summarizing the information in schemas in order to highlight the “important” parts. This shows that applications having different degree of interactivity with the data require different parts to be focused, while others can be hidden. We call this notion the granularity level of the schema.

A varying level of granularity can be achieved by modifying appropriately the summarisation relation of a graph summary. By altering the mappings of the relation, the resulting schema is then more or less coarse, i.e., it carries more or less information about the data. The granularity can be varied through the use of the presented summarisation relations in Section 4.1.3. How coarse a granularity is can be measured thanks to the graph summary precision model introduced in Chapter 5.

**7.1.3. Representation of Graph Summaries**

In this section, we present an RDF model for describing a graph summary. This allows the summary to be queried as any other RDF graph, ultimately so it can be used in conjunction with the entity graph.

Viewing the graph summary as an RDF graph requires the creation of URIs for identifying sumnodes and sumedges. A URI for a sumnode can be created based on the features the summarisation relation it is based on, e.g., on the type information for the Types summary \(R_t\). The sumedge URI can be created by considering the source and target sumnodes, as well as the attribute.

\(^1\)XMLSchema: [http://www.w3.org/XML/Schema](http://www.w3.org/XML/Schema)
**Remark.** Although an edge is represented with a single statement in RDF, this is not the case for a sumedge of the summary if we need to associate with it metadata, e.g., statistics or the sumedge provenance. Therefore, we need to describe formally the sumedge.

For example, consider the sumedge “:Person :writes :Book.”, represented as a triple, which relates the sumnode “:Person” to the sumnode “:Book”. If we need to associate a statistic to that sumedge, we need to RDFify it with four triples:

1. :se :source :Person .
3. :se :label :writes .
4. :se :statistic "1" .

where “:se” is the URI for that sumedge, the first triple describes the source, the second the target, the third the label of the sumedge, and the fourth the statistic associated to that triple.

We depict in Figures 7.2a and 7.2b the ontology of the summary RDF model of the sumnode and sumedge respectively, where the label of a node indicates the type of that node. For example, the range of the origin predicate is xsd:anyURI. In Figure 7.2a the dot represents an intermediate node, which can be translated into a blank node in the RDF model. We describe in the Table 7.2c the vocabulary terms.

In the case where a summary describes a collection of inter-linked datasets, we record the provenance of the summarised edges at the sumnode (resp., sumedge) level using the predicate origin. The predicate label indicates the type or attribute associated to a sumnode (resp., sumedge). We record the features of the summarisation relation, e.g., a type for the Types summary $R_t$, using the predicate feature. In case of a summarisation relation that is based on the type feature, we record as well the attribute type that defined that type.

**Levels of Reification**

Depending on the use of the graph summary, a subset of the presented vocabulary is needed. For example, if no statistic was computed then the predicate statistic is not needed. In this section, we present two possible reifications. A lite reification which reifies only the structural information in the summary, and a full reification which captures all the information made available by the graph summary.

**Full Reification**

In this level of reification, we use all vocabulary terms presented in the previous section. We note nonetheless that the origin predicate is optional, since it is not needed for summarising a single dataset.
7.1. Graph Summary for Web Data Management

Figure 7.2.: RDF reification of the graph summary
Due to the RDF reification of the summary, there exists a case which might cause the summary to be larger than the original. Indeed, if the summarisation relation is a one-to-one mapping from the entity graph to the summary, the number of RDF statements used to describe the summary might be greater than for the entity graph. A single statement in the entity graph requires six statements for describing the corresponding sumedge, i.e., the five triples depicted in the Figure 7.2b plus one for the type attribute.

In addition to those six statements, we must also account for the statements describing the adjacent sumnodes. Depending on the application of the graph summary, the full description of the graph summary can be excessive.

Table 7.1a reports the Types summarisation of a simple entity graph for which the summarisation has a one-to-one mapping. We observe that the full reification of the summary consists of 18 triples, against 3 for the entity graph.

**Lite Reification**

For this reification, we don’t retain information such as provenance or statistics. Instead, we are only concerned with the structural description of the graph. A direct consequence is that the size of the RDF summary is reduced significantly. In order to do so, we only need to create a URI for the sumnodes through object invention [HS89].

Although this reification is not as insightful as the previous one, it provides nonetheless a succinct description of the structure of the graph. Table 7.1b reports the lite reification of the graph.

**7.2. SPARQLed: Query Recommendation**

When a user formulates a SPARQL query, the user is in fact trying to summarise his information needs. However, this task can be very difficult and time consuming as it requires the user to have a good knowledge of the structure and vocabulary of the dataset he is trying to query. This task becomes even more complex if the user is trying to formulate a query across multiple data sources. In order to gain such knowledge, the user must explore and investigate the data itself before querying it.

To save the user from such a tedious work we present the *Assisted SPARQL Editor*, an application that leverages the graph summary to help the user into effectively formulating complex SPARQL queries, even without prior knowledge about the structure and vocabulary of the data sources.

In this section, we first give an overview of the possible recommendations supported by the SPARQL editor, and then, we introduce some of the main concepts in SPARQL. The summary is RDFied using the model presented in Section 7.1 so that it can be queried: we describe how the current state of the SPARQL query is used to query the graph summary. The *Full* reification of the summary is used. The results from the summary are then recommended to the user as possible structural query elements.
7.2. SPARQLed: Query Recommendation

Table 7.1.: Full and lite reification of a Types summary $R_t$. The summarisation relation $R_t$ has with the example data a one-to-one mapping from the graph $G$ to the summary $S_t$. The node $p1$ is mapped to the sumnode $t1$ and $d1$ to $t2$.

7.2.1. Overview of SPARQL Recommendations

The Assisted SPARQL Editor supports four kinds of recommendations: class, predicate, relationships between variables and named graphs. Examples of such recommendations are pictured in Figures 7.3. Recommendations of entity node labels as well as literal node labels are not supported since such content data is discarded from the graph summary.

During query formulation, the assisted editor provides one of these four different types of recommendations to the user based on the state of the edited query. The state of the edited query is composed of an incomplete graph pattern and the cursor position. The cursor position materialises the Point Of Focus (POF), i.e., the unknown element of the graph pattern for which the user requests recommendations. The POF is depicted in Figures 7.3 with an angle bracket.

**Class recommendation.** Figure 7.3a depicts the recommendations of possible classes for a variable. Given that the variable `?Article` is associated to a predicate `akt:hasAuthor`, the system will only recommend classes that are mentioned with this predicate in the graph.

**Predicate recommendation.** Figure 7.3b depicts the recommendation of additional predicates for the class `akt:Article-Reference` which co-occur with the property `akt:hasAuthor`.

(a) Full reification. The edge $(p1, author, d1)$ is mapped to the sumedge that is reified with $e1$. 

(b) Lite reification.
Figure 7.3.: Overview of the possible recommendations for the http://www.rkbexplorer.com dataset depending on the Point of Focus. The Point of Focus is displayed by the green angle bracket <.

**Relationships recommendation.** Figure 7.3c depicts recommendations of possible relationships between two variables. The only predicates that are returned as a recommendation are those that connect an author to an organisation.

**Named graph recommendation.** In Figure 7.3d, possible named graphs are recommended to the user. The proposed named graphs are those that contain at least one solution for the graph pattern.

### 7.2.2. SPARQL Graph Pattern

In this section, we introduce the main concepts of SPARQL that are used later in the description of our recommendation engine. SPARQL is the standard query language for RDF data and is based around graph pattern matching. Triple Pattern described in Definition (6.7.2) is the building block in SPARQL.
7.2. SPARQLed: Query Recommendation

Triple patterns can be combined into a Basic Graph Pattern (BGP) [PS08]. More complex graph patterns can be formed by combining BGPs in different ways [PS08]: Group Graph Pattern, Optional Graph Pattern, Alternative Graph Pattern and Patterns on Named Graphs.

A SPARQL query can be translated into an Abstract Syntax Tree (AST). The AST is a tree structure composed of all the logical operators of the query and where leaf nodes are triple patterns to be evaluated. Such an AST is the data structure used by our system to translate the current user need into possible recommendations. In our implementation, the AST may contain no more than one incomplete triple pattern and a special symbol ‘<’ to indicate the Point of Focus. A triple pattern is incomplete if all three components are not present.

7.2.3. Definitions

We introduce in this section definitions that are needed for describing our approach for SPARQL query recommendation.

Similar to triple patterns in SPARQL, summary patterns are the building blocks to construct a graph summary query.

Definition 7.2.1 (Summary Pattern)

Let $G = \langle V, A, l_V \rangle$ be a graph, $S = \langle W, B, l_W \rangle$ be the summary of $G$, and $\mathcal{L}$ be the set of labels, and $\text{Var}$ an infinite set of variables.

A summary pattern is a tuple $t = \langle s, p, o \rangle$ where $t \in (\mathcal{L} \cup \text{Var}) \times (\mathcal{L} \cup \text{Var}) \times (\mathcal{L} \cup \text{Var})$.

A sumedge $(x, \alpha, y) \in B$ of the summary $S$ is a match for a summary pattern $t$ if and only if the non-variable components of $t$ are equal to those of the sumedge $(x, \alpha, y)$.

We call class triple pattern (CTP) a triple pattern which predicate is a type attribute (Section 2.3.1).

Definition 7.2.2 (Class Triple Pattern)

Let $G = \langle V, A, l_V \rangle$ be a graph, $\mathcal{L}$ be the set of labels, and $\text{Var}$ an infinite set of variables. A class triple pattern (CTP) is a triple $t = \langle s, \tau, o \rangle$ where $t \in (\mathcal{L} \cup \text{Var}) \times T \times (\mathcal{L} \cup \text{Var})$ where predicate($t$) is a type attribute.

Similarly to triple patterns, summary patterns may be used to build more complex pattern, such as BGPs, as described in the previous section.

7.2.4. From Entity Graph to Graph Summary

In order to suggest the possible structural elements to the user with respect to the current state of his query, we need to normalise the AST of the query to match the RDF data model of the summary introduced in Section 7.1. The normalised AST is then evaluated on the graph summary and the possible structural elements are retrieved and presented to the user.
Chapter 7. Graph Summary Applications

The AST normalisation is performed in three steps:

1. transformation of the POF symbol ‘<’ into a variable to project as the query solution;
2. removal of content elements from the AST; and
3. mapping of triple patterns into summary patterns.

Each step is part of the normalisation sequence: an operation is performed on a step, which result becomes then the input of the next step. In the following section we describe in detail the normalisation of a SPARQL query.

Abstract Syntax Tree Normalisation

In this section, we describe the algorithm for normalising the AST of a SPARQL query into another AST that can be executed over a graph summary. We illustrate each step thanks to the following SPARQL query over a fictional entity graph:

```
ASK WHERE {
    :article1 a :Article ;
        :title ?t .

    ?i a :Institute ;

    ?p :name "Renaud" ;
        <                # POF
}
```

**Step 1 — Projection of the POF.** The first step consists in defining the variable to project as the query solution. In the triple pattern containing the POF, we transform the POF symbol into a variable ?POF and complete that triple pattern with a wildcard variable if needed. We denote as a wildcard variable a variable that is unique in the query. The initial Query Form [PS08] of the AST is replaced by a projection of the POF variable using the SELECT form.

For example in the ongoing SPARQL query, the POF symbol ‘<’ at line 9 is translated into the variable ?POF, and the wildcard variable ?_z2 is added to complete that triple pattern. The result of that step on the query is depicted below:

```
SELECT ?POF WHERE {
    # Projection of the POF
    :article1 a :Article ;
        :title ?t .
}
```
### Step 2 — Removal of content elements.

The second step consists in removing all content elements from the AST. A content element is an element that describes a specific aspect of an entity — thus it does not inform about its structure.

Literals and URIs that appear in a triple pattern at a subject or object position are replaced with a wildcard variable. For instance, the literal “Renaud” in line 8 is replaced by the variable \(?_z1\) in the SPARQL query below.

If the triple pattern is a CTP, then only the element at the subject position is replaced with a wildcard variable. The triple pattern on line 2 of the ongoing SPARQL example is a CTP. Therefore, the object URI is left as is, but the subject URI \(\text{:article1}\) is replaced with the variable \(?_z0\).

```sparql
SELECT ?POF WHERE {
    ?_z0 a :Article ;
    :title ?t .
}
```

### Step 3 — Mapping.

The third step consists of mapping each triple pattern to a summary pattern according to the following two rules. The RDF data model of the summary this operation maps to is described in Section 7.1.

1. If the triple pattern \(\langle ?s, p, o \rangle\) is a CTP, then it is replaced with the following BGP:
   a) A triple pattern \(\langle ?s, :feature, :b0 \rangle\) is created in order to describe the CTP;
   b) A triple pattern \(\langle :b0, :label, o \rangle\) is created to set the label of the type, e.g., :Article; and
   c) A triple pattern \(\langle :b0, :type, p \rangle\) is created to set the label of the type attribute, e.g., rdf:type.

\[\text{SELECT } ?POF \text{ WHERE } \{
\begin{align*}
?_z0 & \text{ a :Article ;} & \text{# The variable } ?_z0 \text{ replaces the URI :article1} \\
& \text{ :title } ?t .
\end{align*}
\]
Chapter 7. Graph Summary Applications

2. Otherwise, the triple pattern \( \langle ?s, p, ?o \rangle \) is replaced with the following BGP. We remark that in this case, both subject and object elements are variables due to the previous step.

   a) A new wildcard variable \(?x\) representing a sumedge \(x\) is created;
   
   b) A triple pattern \( \langle ?x, :source, ?s \rangle \) is created to set the source of the sumedge;
   
   c) A triple pattern \( \langle ?x, :target, ?o \rangle \) is created to set the target of the sumedge; and
   
   d) A triple pattern \( \langle ?x, :label, p \rangle \) is created to set the label of the sumedge.

For example, if we apply these mapping rules on the ongoing SPARQL query, the first triple pattern \( \langle ?_z0, a, Article \rangle \) is translated into the summary pattern displayed on line 3 below since it is a CTP. The second triple pattern \( \langle _z0, :title, ?t \rangle \) is translated into the BGP on line 5 of the query below.

```sparql
1 SELECT ?POF WHERE {
2   # Mapping of the triple pattern ?_z0 a :Article
4   # Mapping of the triple pattern ?_z0 :title ?t
5   ?se0 :source ?_z0; :target ?t; :label :title .
6
7   # Mapping of the triple pattern ?i a :Institute
9   # Mapping of the triple pattern ?i :employs ?p
11
12  # Mapping of the triple pattern ?p :name ?_z1
14  # Mapping of the triple pattern ?p ?POF ?_z2
16 }
```

**Graph graph pattern.** A *Graph Graph Pattern* [PS08] is an operator in SPARQL that indicates which dataset the associated graph patterns should be queried on. For example, in the SPARQL query below the first triple pattern is queried against the dataset :dbpedia. In the second graph graph pattern, the user seeks recommendations on the possible datasets where the triple pattern “?movie :title "Terminator"” might occur in.

```sparql
SELECT * WHERE {
   GRAPH :dbpedia {
      ?character :name "John Connor"
   }
```

138
7.2. SPARQLed: Query Recommendation

We need to retain the desired dataset thanks to the vocabulary term :origin in order to provide correct recommendations to the user. The query above is then normalised to the query below:

```
SELECT ?POF WHERE {
  # Mapping of the first graph graph pattern
  ?se0 :source ?character; :target ?_z0; :label :name, :origin :dbpedia .
  # Mapping of the second graph graph pattern
}
```

7.2.5. User Interaction

The purpose of the Assisted SPARQL query editor is to assist a user in writing a query. Therefore, there is a need to consider how the user interacts with the application. In this section, we present two angles into improving the interaction. First, we discuss the ranking of recommendations. Next, we introduce the notion of scope for a recommendation.

**Recommendation Ranking**

In Section 6.7, we discussed the ranking of solutions to a SPARQL query executed on a graph summary. For the purpose of improving the user interaction, it is possible to rank the recommendations using the MF ranking model. As reported by the experiment in Section 6.7.3, the MF model provides noticeable benefits over a simple ranking.

Applying the MF model to rank recommendations allow us to present first to the user the recommendations that are most likely not part of the error set. Thus, we reduce the chances of a user to be frustrated over a recommendation that yields no result.

**Recommendation Scope**

During the formulation of a query, the query may contain multiple BGPs where some do not return any results. The system will therefore no longer produce recommendations, as the evaluation of the graph summary query will also not produce any results. However, this can be interpreted incorrectly by the user since he might believe that the dataset does not contain any other information. In order to minimise this issue, we introduce the notion of recommendation scope.
Chapter 7. Graph Summary Applications

The recommendation scope helps to reduce the extent of the area that is relevant for the recommendation. Instead of taking into account the full SPARQL query, the recommendation engine will take only a relevant subset.

The recommendation scope is defined recursively by including all the triple patterns with a path to the POF variable. A breadth-first search algorithm on the query, starting on the POF node, is performed in order to find all the graph components that are connected to the POF. All the graph components that are not connected to the POF are removed. This prevents non-relevant (to the POF) triple patterns from limiting the recommendations.

For example, we take the SPARQL query that we used as illustration in the previous section.

```sparql
ASK WHERE {
  :article1 a :Article ; :title ?t .
  ?p :name "Renaud" ; <
}
```

The first BGP on line 2 is not connected to the BGP that contains the POF on line 4. Indeed, neither the URI :article1 nor the variable ?t are referenced in the other two BGPs. Therefore, that first BGP is outside of the recommendation scope. After normalising the SPARQL query, we remove that BGP from the query to be run on the graph summary, as depicted below:

```sparql
SELECT ?POF WHERE {
  # Mapping of the triple pattern ?i a :Institute
  ?i :feature [ 
    :label :Institute ;
    :type rdf:type
  ] .
  # Mapping of the triple pattern ?i :employs ?p
  ?se1 :source ?i; 
    :target ?p; 
    :label :employs .
  # Mapping of the triple pattern ?p :name ?_z1
  ?se2 :source ?p; 
    :target ?_z1; 
    :label :name .
  # Mapping of the triple pattern ?p ?POF ?_z2
  ?se3 :source ?p; 
    :target ?_z2;
}
```
7.3. Web Data Inspector

Web Data is a collection of heterogeneous datasets, coming from a variety of sources and spanning over many diverse domains. Some datasets are created by integrating others. Datasets owners expose links between other datasets and their own in order to improve their connectivity to the rest of Web Data. This thoughtful process, which is not an objective per se, allows more relevant information to be accessible for the user. As a user trying to retrieve some information from this heterogeneous collection of datasets, it is necessary to have sufficient insight into the dataset of interest in order to formulate queries: the user needs to understand how the data is structured, how a dataset is connected to others, and to assess in some way the quality of the data.

In this section, we present the Web Data Inspector as a tool for inspecting the edges of a dataset through the use of graph summaries.

7.3.1. Views into a Dataset

Based on the graph summary, the Web Data Inspector presents different views of a dataset. This tool was part of the Sindice [ODC+08] project; its home page, not anymore accessible with the end of the Sindice project, is shown in Figure 7.4 and provides the user with a text box where he can enter the dataset’s name of interest — here, the domain name of the web site. By clicking the “Check” button, the user will be provided with several views useful in exploring the content and the structure of the chosen dataset.

Also, from the home page, the user has the possibility to choose an RDF class or an RDF property and then, upon clicking the “Check” button, the user is shown the provenance datasets of such RDF element (classes or properties) as well as its relationships with other RDF elements. In order to provide such information, a summary of the dataset collection is generated and is presented thanks to the RDF data model described in Section 7.1.

In the next subsections, we describe in more detail the different views that the user is provided with when exploring a dataset using the Web Data Inspector.

General View

The general view tab as depicted in Figure 7.5 shows a summary of some of the particular dataset’s properties and statistics. This includes the domain address of the dataset and the size in terms of triples and documents. Further details and analysis is found in the other tabbed sections: “Inside this dataset”, “To/from datasets”, and “Third party”.

Chapter 7. Graph Summary Applications

Figure 7.4.: Web Data Inspector - home page

Figure 7.5.: Web Data Inspector - general view
Inside This Dataset

The “Inside this dataset” tab as depicted in Figure 7.6 delves deeper into the dataset to explore and present information about the RDF properties and classes used within. This view shows the links that are classified as internal only to the dataset. From this view, a user may learn more about the structure and content of the dataset.

The first two boxes present to the user the top properties and classes used in the dataset. The next two present to the user a property, and when selecting a specific one, the classes that are connected by that property. The last two boxes report the opposite: when selecting a specific pair of classes, the properties connecting the two are presented.

We present below a SPARQL query that is used for retrieving the top properties and classes occurring in a dataset. The query is executed over the graph summary of the inspected data collection. The dataset that is being inspected is denoted as “MYGRAPH” in the following query. The output of this query is used for populating the first two boxes on the top in Figure 7.6.

```sparql
SELECT ?type ?label (SUM(?nb) AS ?total) WHERE {
    BIND ("class" AS ?type)
    ?n a :SNode;
    :origin <MYGRAPH>;
    :statistic ?nb;
} UNION {
    BIND ("property" AS ?type)
    ?e a :SEdge;
    :origin <MYGRAPH>;
    :statistic ?nb;
    :label ?label .
} GROUP BY ?type ?label
```

In the following SPARQL query, we describe how the statistics for the last four boxes of Figure 7.6 are retrieved from the graph summary. The query returns (a) the class of the subject component; (b) the class of the object component; and (c) the total number of links that connects the previous two classes for a specific property.

```sparql
SELECT ?sourceClass ?targetClass ?property (SUM(?nb) AS ?total) WHERE {
    ?e a :SEdge;
    :origin MYGRAPH ;
```
Chapter 7. Graph Summary Applications

To/From This Dataset

The “To/from datasets” tab as depicted in Figure 7.7 presents the existing relationships this dataset has with other datasets. From here, exploration can span across the many domains that the particular dataset is linked with.

The first two boxes present the datasets the current dataset being inspected connects to. Instead, the next two boxes report on the datasets that link to the current dataset.

We show below a SPARQL query that is used for retrieving the data necessary for the boxes depicted in Figure 7.7. We denote with “MYGRAPH” the dataset being explored, e.g., bbc.co.uk in the figure. The query retrieves links that connects to and from the “MYGRAPH” dataset; “to” the dataset is shown in the first block of the UNION, while “from” the dataset is shown in the second block.

Third-Party Links

The edge authority as defined in Definition (4.2.3) helps dataset owners to distinguish another important aspect: third-party links. These represent links that do not have
7.3. Web Data Inspector

Figure 7.6.: Web Data Inspector - inside this dataset
Chapter 7. Graph Summary Applications

Figure 7.7.: Web Data Inspector - to/from this dataset
authority; the ability to differentiate a link as being a third-party link helps to discover if the link is incorrect or specify relationship that the owner does not explicitly agree with. In some cases, these links can be connotative to the idea of e-mail spam.

The “Third-party links” tab as depicted in Figure 7.8 shows to the user information about third-party links, inferred from the graph summary. Statistics reported in this section differ from those reported in the “To/From This Dataset” by filtering links which were not published in the inspected dataset, e.g., bbc.co.uk in the figure.

```
SELECT ?PublishedOn ?sourceDomain ?targetDomain
    ?property (SUM(?nb) AS ?total) WHERE {
            ?e a :SEdge;
            :origin ?PublishedOn;
            :label ?property;
            :statistics ?nb .

            {
                BIND (<MYGRAPH> AS ?sourceDomain)
                ?e :source [ :origin <MYGRAPH> ];
            } UNION {
                BIND (<MYGRAPH> AS ?targetDomain)
                ?e :source [ :origin ?sourceDomain ];
            }
            FILTER ?PublishedOn != <MYGRAPH>
    }
GROUP BY ?PublishedOn ?sourceDomain ?targetDomain ?property
```

### 7.3.2. Evaluation

In this section, we show how the Web Data Inspector tool works across some notable datasets: bbc.co.uk, sws.gonames.org, and www.rottentomatoes.com. Statistics reported in this evaluation date from April 2012.

**bbc.co.uk**

The dataset “bbc.co.uk” uses RDFa to annotate their “Programmes” and “Music” websites. As depicted in Figure 7.5, there are over 6M triples that cover brands, series (seasons), episodes, artists, broadcast events, broadcast services, etc. The top three most used classes in the “bbc.co.uk” dataset are: “article” which appears 44.12K times, “mo:MusicArtist” 24.18K times and “vcard:Name” 20.48K times. The top three most popular properties are “po:episode” with 2.67M occurrences, “rdfs:label” with 216.78K and “wo:adaptation” with 139.14K, as depicted in Figure 7.6.
The “bbc.co.uk” dataset includes links to further URIs, allowing the user to discover more data, e.g. about members of a band or about a certain series episode. It also includes a “owl:sameAs” link with 21.61K occurrences to the corresponding DBpedia resource, allowing the user to aggregate more data about that band, extracted from Wikipedia’s infoboxes. From the “To/from dataset” view, a user may note that there are about 3K links that link “bbc.co.uk” to well known datasets such as “dbpedia.org”, “myspace.com”, and “imdb.com” as reported by the “outgoing links” box. On the incoming links side, the user can observe that few datasets are linked to “bbc.co.uk” as there are only 275 links that link other datasets like “freebase.com” and “dbpedia.org” to “bbc.co.uk”.

**sws.geonames.org**

The dataset “sws.geonames.org” is a dump dataset rather than a website which provides over 129M triples, as depicted in Figure 7.5, about countries and geographical locations. The user can see what classes occur in that dataset such as “geonames:Feature” which appears 7.85M times, “wgs:Point” 4.36K times and “foaf:Agent” with 2.41K occurrences and properties such as “geonames:nearbyFeatures” with 14.28M occurrences, “geonames:name” with 7.89M occurrences and “geonames:parentFeature” with 7.88M occurrences.

The “sws.geonames.org” dataset includes over 4M links to resources from other datasets like “identi.ca”, “nytimes.com”, or “dbpedia.org” as depicted by the outgoing links box in Figure 7.7. This dataset is also well linked to by other datasets like “naplesplus.us”, “dbpedia.org” or “geospecies.org” with over 274K incoming links.
7.4. Conclusion

Rotten Tomatoes is a film review aggregator which provides reviews, information, and news of films. It contains over 11M triples, as depicted in Figure 7.5, and it uses several proprietary vocabularies. Among the most popular classes, a user may note the classes “actor” which appears 147.19K times, “sorg:Person” 31.00K times, “sorg:AggregateRating” 2.94K times and “video.movie” 2.77K times.

The linkage for this dataset with others is very poor as reported to the user by the number 0 for the outgoing/incoming links box in Figure 7.7. A possible improvement of the dataset might be to increase the linkage of the dataset with links to well known datasets like “dbpedia.org” or specific datasets like “imdb.com”.

7.3.3. Dataset Improvements thanks to Web Data Inspector

In this section we show how the results of the Web Data Inspector can be used as suggestions for improving one’s dataset. Being able to see the classes and the properties of his dataset, the dataset owner is able to have a deep understanding of his dataset. The user can determine if the dataset graph is shaped as planned.

For example, a user may assume that his dataset contains the “foaf:Person” class which has, among others, the “foaf:name” and “foaf:homepage” properties. From the number of the occurrences of these properties, a dataset owner can decide if his dataset is as intended: if it is expected that most people mentioned in the dataset have an homepage, then this should be reflected in similar numbers for the occurrences of the “foaf:name” and “foaf:homepage” properties.

Also, a dataset owner can identify possible mistakes such as typos in the classes/properties names. For example, it is well known that “foaf:name” is a property of the FOAF vocabulary but that “foaf:naem” is not.

7.4. Conclusion

We have presented in this chapter several applications of the graph summary and of the MF ranking model. We introduced an RDF model of a graph summary; this allows a summary to perform the function of a schema for the entity graph. Then we have described an application of the summary which provides recommendations to a user writing a SPARQL query. These recommendations are dependent on the current state of the query, and so only contain elements that may be used at a specific position in the SPARQL query. In addition, we discussed how the MF ranking model can be used in the latter application in order to improve the ranking of recommendations. Finally, we presented another application of graph summaries with the Web Data Inspector, which allows users to delve into a dataset in order to better understand its structure; it can further benefit dataset owners by highlighting incorrect or suspicious parts of the graph.

As future work, we plan to investigate the use of a summary for the purpose of optimising query execution by leveraging the statistics available with the summary. In addition,
Chapter 7. Graph Summary Applications

we will study the application of a summary to improve data partitioning, by relying on the knowledge of the structure of the graph. Furthermore, we want to improve the Web Data Inspector so that it can better assess the quality of a dataset.
Chapter 8.
Conclusions

In the last decade, the Internet has evolved into a global database. The rise of many standards for structured data and the ease of sharing information over the Internet provoked a deluge of (semi-)structured data. In order to bring this into being, the focus was put on the ease of representing and sharing knowledge; engaging people into publishing information was key as illustrated by the 5-stars rewarding system [BL06] proposed by Tim Berners-Lee. However, the lack of insight into the structure of the data prevents the effective use of this inter-connected data.

We have proposed in this thesis a framework for generating summaries of a graph directly from the data itself. A generated summary is we believe the answer for providing insights into one’s data. In this chapter, we summarize the research work conducted in this Ph.D. project, we recall the main contributions achieved, and discuss some possible future works.

8.1. Summary of the Thesis

In the Internet, information about a certain concept can be found within several datasets. Thus, a user is faced with the challenge of integrating the information from a range of sources that may not model their information in the same way. This lack of visibility into the structure of the data impede as well tasks such as discovery and browsing; such tasks require a user to be aware of the structure of the data in order to write queries.

In this thesis, we have presented a methodology that enables users to achieve such and similar tasks:

**Graph Summarisation:** We have introduced a generic framework for summarising the graph structure. By specifying a summarisation relation, a user is able replicate the structure of a graph into a smaller, more manageable graph. That summary retains the structure of the original graph and, therefore, may be used for tasks that would too costly to perform on the original graph. With a summary, one can answer quickly if a certain graph pattern occurs without having to process the original graph. A summary may then hold the function of a schema for the graph.

**Summary Precision:** We have presented a model for measuring the precision of a summary. This models relies on a precise graph summary, such as generated using the
Chapter 8. Conclusions

bисимulation relation, in order to compute which edges of the summary are false positives. This model allows to grasp the impact of a summarisation over two properties of a graph: its schema and its connectivity. We have evaluated in experiments over real-world datasets the tradeoffs between computational complexity and precision of several summarisation relations; they have shown the applicability and advantages of a summary.

Graph Ranking: We developed an extension of Field-based ranking models to support arbitrary directed acyclic graphs. Our model named “MF” provides additional ranking parameters which allows fine tuning for a specific dataset. In addition to the parameter for normalising field length, our approach offers the possibility to account for fields having a variable number of values. We have shown through several experiments that our MF ranking model outperforms traditional field-based ranking models.

Finally, we have presented several applications of graph summarisation and ranking. We have developed an RDF model for a summary, so that it may hold the function of a schema. We have shown how to leverage a summary for providing recommendation of terms when writing a query.

8.2. Directions for Future Research

Graph summarisation offers a solution to the problem of data heterogeneity and unknown structure of the graph. This opens a wide range of potential applications. In the following, we present a list of possible future directions of research related to the problem of graph summarisation:

Graph Summary Updates As soon as a dataset is updated, i.e., either an edge is added, removed, or modified, the summary becomes out-dated. In order for the summary to reflect the current structure of a graph, we need to update the summary. However, the proposed framework does not propose an efficient mechanism for keeping a summary fresh without having to generate the summary anew. An approach worth investigating is to leverage the existing summary in order to compute what part of the graph changed, and how those changes impacted on the connected component of the graph.

Data Quality Thanks to the summary reflecting the structure of the graph, it is possible to assess the quality of the data from a modeling perspective. One is able to identify structural inconsistencies with the data, such as (a) missing attributes; (b) invalid combination of attributes; or (c) incorrect type usage. Using such quality features of the data, one can improve the data modeling in order to better fit an application. How a summary may be used for such a purpose is an open question that would require further study.
8.3. Current Limitations

**Data Integration** The task of integrating data is often challenging because of datasets having varying, loose-defined or extremely complex, schemas. With Web Data, this task is made even more difficult by the lack of schema in general to begin with. How can we improve this task in the context of Web Data through the use of graph summaries?

**Approximate Summary** Given the heterogeneity of Web Data, we discussed that approximate graph summaries are the only viable option in many use cases. Also, we can assess the quality of a summary thanks to the summary precision model. Using this model as a guide, there is a need to develop approximate summarisation relations that have less impact on the schema and connectivity precisions of the generated summary.

**Entity Graph Ranking** We generalised Field-based ranking models to directed acyclic graphs with the introduction of our MF ranking model. Entities found in Web Data are modeled differently from one dataset to the other. Indeed, an entity may be represented as a star-graph in a dataset, while showing in another a more complex structure with a graph having few hops. With the help of graph summaries of the datasets to try and reconcile different modeling, is it then possible to rank entities having different representations?

It is our belief that the work achieved in this thesis answers a growing need; that is, to get a better understanding of datasets structure. This knowledge is key for we assume a large number of applications such as data discovery, query optimisation, and data integration. The concept of graph summarisation presented in this thesis applied to Web Data opens the door to better and new applications, and raises as well novel challenges as outlined in this section.

8.3. Current Limitations

We acknowledge the following limitations with regards to the work done in this thesis:

**Expressivity of Summarisation Relations** The graph summarisation framework proposed in Chapter 4 assumes that the assignment of a node in the graph to a node of the summary is deterministic. Therefore, graph summarisations that have an undeterministic assignment, e.g., subject to an heuristic, as the one proposed in [NRS08] cannot be expressed in our framework.

**Approximate Summaries** The loss of connectivity precision by approximate summaries outlined in Section 4.1.3 would most impact applications that rely heavily on the exactness of paths information. Such applications would need to rely on a summarisation relation that iterates over the graph (e.g., Bisimulation summary), which is however a costly operation on large graphs within shared-nothing environments (e.g., Hadoop\(^1\)).

Part IV.

Appendices
Appendix A.

Data Tables of the Graph Summarisation Evaluation

In this Appendix, we report the raw results of the evaluation conducted on the graph summarisation as presented in Section 5.3. Table A.1 reports some statistics about the datasets used in the evaluation. In Table A.2 we report performance results of computing summaries over the datasets. We report in Tables A.3 and A.4 the connectivity and schema precisions measurements of several summaries generated from the datasets.
Appendix A. Data Tables of the Graph Summarisation Evaluation

### Table A.1.: Size and order of graph summaries.

| Dataset     | $|\mathcal{W}_u|$ | $|\mathcal{R}_u|$ | $|\mathcal{W}_l|$ | $|\mathcal{R}_l|$ | $|\mathcal{W}_a|$ | $|\mathcal{R}_a|$ | $|\mathcal{W}_{ioa}|$ | $|\mathcal{R}_{ioa}|$ | $|\mathcal{W}_{ioat}|$ | $|\mathcal{R}_{ioat}|$ |
|-------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| dbpedia     | 288524         | 18104329       | 1450687        | 19725084       | 21825          | 1903078        | 1728460        | 22430463       | 55296          | 2565315        |
| twc-log     | 450            | 30480          | 1190           | 29377          | 1309           | 44174          | 2265           | 60104          | 1805           | 58251          |
| enipedia    | 128            | 2420           | 163            | 2533           | 536            | 13001          | 755            | 16143          | 810            | 18308          |
| b3kat       | 20             | 716            | 233            | 5064           | 72835          | 1028849        | 154266         | 2087985        | 74553          | 1048702        |
| lobid       | 19             | 679            | 86             | 311            | 812            | 13391          | 1142           | 17809          | 1305           | 16147          |
| bnb         | 27             | 297            | 46             | 365            | 14214          | 223979         | 10502          | 231727         | 14730          | 230299         |
| datos       | 23             | 428            | 25             | 372            | 76995          | 1200563        | 77095          | 1209914        | 77616          | 1204300        |
| gnd         | 22             | 293            | 22             | 293            | 3120           | 32718          | 3003           | 36713          | 6889           | 70787          |
| eures       | 18             | 89             | 18             | 61             | 2009           | 35529          | 2022           | 35552          | 2037           | 35622          |
| europeana   | 5              | 66             | 5              | 66             | 466            | 5235           | 466            | 5235           | 466            | 5235           |
| wb          | 4              | 216            | 3              | 183            | 271            | 6360           | 271            | 6360           | 293            | 6987           |
| cordis      | 7              | 92             | 7              | 92             | 1497           | 28607          | 1500           | 28610          | 1725           | 31727          |
| ny-times    | 2              | 34             | 2              | 34             | 57             | 683            | 57             | 683            | 65             | 794            |

### Table A.2.: Performance comparison. We report the CPU time in ms of the edges step in the graph summarisation computation. For each category of dataset complexity, we report the mean $\mu$ of the CPU time.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$R_u$</th>
<th>$R_l$</th>
<th>$R_a$</th>
<th>$R_{ioa}$</th>
<th>$R_{ioat}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia</td>
<td>7746385</td>
<td>52608760</td>
<td>9983935</td>
<td>104255540</td>
<td>105685365</td>
</tr>
<tr>
<td>twc-logd</td>
<td>6974425</td>
<td>4213100</td>
<td>6542530</td>
<td>6557955</td>
<td>6519540</td>
</tr>
<tr>
<td>enipedia</td>
<td>1518265</td>
<td>1314045</td>
<td>1381875</td>
<td>1327200</td>
<td>1400110</td>
</tr>
<tr>
<td>$\mu_H$</td>
<td>4246345</td>
<td>2763572</td>
<td>396202</td>
<td>3943077</td>
<td>3989825</td>
</tr>
<tr>
<td>b3kat</td>
<td>137235825</td>
<td>128601525</td>
<td>135458935</td>
<td>135080175</td>
<td>136010320</td>
</tr>
<tr>
<td>ecs</td>
<td>946635</td>
<td>812700</td>
<td>983015</td>
<td>921185</td>
<td>946450</td>
</tr>
<tr>
<td>lobid</td>
<td>187415830</td>
<td>161954355</td>
<td>204759505</td>
<td>205217415</td>
<td>204657490</td>
</tr>
<tr>
<td>bnb</td>
<td>25310415</td>
<td>19193380</td>
<td>21927855</td>
<td>21776020</td>
<td>21757485</td>
</tr>
<tr>
<td>datos</td>
<td>8271535</td>
<td>8215710</td>
<td>8387465</td>
<td>8389250</td>
<td>8478590</td>
</tr>
<tr>
<td>gnd</td>
<td>1510455</td>
<td>1596060</td>
<td>1569565</td>
<td>1563460</td>
<td>1580720</td>
</tr>
<tr>
<td>eures</td>
<td>1378185</td>
<td>1223370</td>
<td>1448975</td>
<td>1468045</td>
<td>1472430</td>
</tr>
<tr>
<td>$\mu_M$</td>
<td>51724125</td>
<td>45942442</td>
<td>53505035</td>
<td>53487935</td>
<td>53557640</td>
</tr>
<tr>
<td>europeana</td>
<td>7995115</td>
<td>7871920</td>
<td>8327500</td>
<td>8362750</td>
<td>8498575</td>
</tr>
<tr>
<td>wb</td>
<td>10508190</td>
<td>10592170</td>
<td>10340025</td>
<td>10412450</td>
<td>10291955</td>
</tr>
<tr>
<td>cordis</td>
<td>1642030</td>
<td>1536155</td>
<td>1620345</td>
<td>1605850</td>
<td>1711790</td>
</tr>
<tr>
<td>ny-times</td>
<td>594285</td>
<td>549680</td>
<td>563200</td>
<td>566740</td>
<td>563110</td>
</tr>
<tr>
<td>$\mu_L$</td>
<td>5181727</td>
<td>5149088</td>
<td>5208950</td>
<td>5227250</td>
<td>5254718</td>
</tr>
</tbody>
</table>

$\mu$: Mean CPU time in ms.
### Table A.3.: Connectivity precision comparison. For each category of dataset complexity, we report the mean $\mu$ of the connectivity precision.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{Err}(R_{a})_{\text{con}}$</th>
<th>$\text{Err}(R_{b})_{\text{con}}$</th>
<th>$\text{Err}(R_{a})_{\text{con}}$</th>
<th>$\text{Err}(R_{b})_{\text{con}}$</th>
<th>$\text{Err}(R_{a})_{\text{con}}$</th>
<th>$\text{Err}(R_{b})_{\text{con}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>P2</td>
<td>P1</td>
<td>P2</td>
<td>P1</td>
<td>P2</td>
<td>P1</td>
</tr>
<tr>
<td>twc-logd</td>
<td>0.0107</td>
<td>0.0003</td>
<td>0.1846</td>
<td>0.0002</td>
<td>0.1444</td>
<td>0.0175</td>
</tr>
<tr>
<td>enipedia</td>
<td>0.0059</td>
<td>0.0005</td>
<td>0.2982</td>
<td>0.0046</td>
<td>0.0632</td>
<td>0.283</td>
</tr>
<tr>
<td>µH</td>
<td>0.0083</td>
<td>0.0004</td>
<td>0.2414</td>
<td>0.0001</td>
<td>0.0091</td>
<td>0.0013</td>
</tr>
<tr>
<td>b3kat</td>
<td>0.3162</td>
<td>0.0000</td>
<td>0.2304</td>
<td>0.0001</td>
<td>0.2480</td>
<td>0.0035</td>
</tr>
<tr>
<td>ecos</td>
<td>0.0806</td>
<td>0.0000</td>
<td>0.0779</td>
<td>0.0001</td>
<td>0.0320</td>
<td>0.0048</td>
</tr>
<tr>
<td>µM</td>
<td>0.0121</td>
<td>0.0001</td>
<td>0.0262</td>
<td>0.0001</td>
<td>0.0091</td>
<td>0.0013</td>
</tr>
<tr>
<td>µL</td>
<td>0.0757</td>
<td>0.0003</td>
<td>0.1190</td>
<td>0.0002</td>
<td>0.1145</td>
<td>0.0181</td>
</tr>
<tr>
<td>europeana</td>
<td>0.3641</td>
<td>0.0345</td>
<td>0.3641</td>
<td>0.3944</td>
<td>0.5429</td>
<td>0.3944</td>
</tr>
<tr>
<td>vw</td>
<td>0.6935</td>
<td>0.0476</td>
<td>0.6320</td>
<td>0.0476</td>
<td>0.9142</td>
<td>0.8479</td>
</tr>
<tr>
<td>cordis</td>
<td>0.0064</td>
<td>0.0094</td>
<td>0.0064</td>
<td>0.0094</td>
<td>0.1619</td>
<td>0.0347</td>
</tr>
<tr>
<td>ny-times</td>
<td>0.3830</td>
<td>0.0798</td>
<td>0.3830</td>
<td>0.0798</td>
<td>0.6126</td>
<td>0.6813</td>
</tr>
<tr>
<td>µL</td>
<td>0.3617</td>
<td>0.0128</td>
<td>0.3617</td>
<td>0.1928</td>
<td>0.5579</td>
<td>0.4896</td>
</tr>
</tbody>
</table>

### Table A.4.: Schema precision comparison.
Appendix B.

Data Tables of the DAG Ranking Evaluation

B.1. Entity Ranking

Table B.1 reports the evaluation results of comparing MF extensions BM25MF and PL2MF against several state-of-the-art ranking functions. The p-Value is computed with the two-tailed Wilcoxon matched-pairs signed-ranks test [DJSH03, BCC10], where a statistically significant difference at level 0.10 is marked with one star * and at level 0.05 with two stars **. BM25MF and PL2MF are used as a baseline in this test. Δ% indicates the difference in percentage between the two MAP values compared in that test.

Table B.2 reports the MAP scores of BM25MF and PL2MF combined with each weight individually and using the normalisation parameters values from the Table 6.1. Apart from the row “Without Attribute Label”, all runs consider the attribute label as an additional value as in the previous experiments.

B.2. Graph Summary Ranking

B.2.1. Queries

The queries are grouped by its complexity, as outlined in Section 6.7.1. A group is identified with the a string having the following regular expression: "\d+(-\d+)*". For instance, the string “2-1” identifies a group of queries that have two star-shaped patterns, one with two triple patterns and the other with only one.

The SPARQL queries below are extracted from the logs available in the USEWOD2013 dataset [LRBH13]. We make use of the following prefixes:

dbo: http://dbpedia.org/ontology/
dbp: http://dbpedia.org/property/
foaf: http://xmlns.com/foaf/0.1/
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
rdfs: http://www.w3.org/2000/01/rdf-schema#
Table B.1.: Comparison of state-of-the-art candidates against the MF generalizations. 

Mean Average Precision (MAP) and the Precision at 10 (P@10) scores of PL2MF and BM25MF and the other state-of-the-art candidates; a $p$-Value is computed using the two-tailed Wilcoxon matched-pairs signed-ranks test, where one star * marks statistically significant difference at level 0.10, and two stars ** at level 0.05, with BM25MF (resp., PL2MF) used as a baseline; $\Delta \%$ indicates the difference in percentage between the two MAP values compared in that test.

<table>
<thead>
<tr>
<th>Method</th>
<th>INEX09 MAP</th>
<th>P@10</th>
<th>p-Value</th>
<th>$\Delta %$</th>
<th>SS10 MAP</th>
<th>P@10</th>
<th>p-Value</th>
<th>$\Delta %$</th>
<th>SS11 MAP</th>
<th>P@10</th>
<th>p-Value</th>
<th>$\Delta %$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25MF</td>
<td>0.1593</td>
<td>0.3982</td>
<td>-</td>
<td>-</td>
<td>0.1303</td>
<td>0.3783</td>
<td>-</td>
<td>-</td>
<td>0.1811</td>
<td>0.1880</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>With Attribute Label</td>
<td>0.1484</td>
<td>0.3800</td>
<td>6.9e-03</td>
<td>-10.84</td>
<td>0.1241</td>
<td>0.3783</td>
<td>4.5e-01</td>
<td>-10.66</td>
<td>0.1763</td>
<td>0.1940</td>
<td>8.6e-01</td>
<td>-10.66</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>0.1474</td>
<td>0.3109</td>
<td>3.2e-03</td>
<td>-10.78</td>
<td>0.1058</td>
<td>0.2304</td>
<td>2.1e-04</td>
<td>-10.55</td>
<td>0.0655</td>
<td>0.1040</td>
<td>1.4e-04</td>
<td>-10.45</td>
</tr>
<tr>
<td>With Function (6.13)</td>
<td>0.1594</td>
<td>0.3982</td>
<td>-</td>
<td>-</td>
<td>0.1303</td>
<td>0.3783</td>
<td>-</td>
<td>-</td>
<td>0.1811</td>
<td>0.1880</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B.2.: Evaluation of the weights effectiveness on PL2MF and BM25MF.
In Tables B.3, B.4, B.5, and B.6 we report the SPARQL queries grouped by complexity.

**B.2. Normalisation Parameters Effect**

We divided the queries into two categories: *simple graphs* and *complex graphs*. Simple graphs indicate queries that have only one star, i.e., the queries from the complexity categories 10, 9, 6, 5, 4, 3, and 2. Complex graphs represents queries that contain more than one star, i.e., the queries with complexity categories 1-1-4, 1-1-3, 1-1-2, 3-4, 2-2, 1-5, 1-4, 1-3, 1-2, and 1-1. We report in Table B.7 the average across complexity categories of MAP values. The results are reported in the form of a matrix, where a row represents the parameter $b_v$ of the Normalisation 1, and a column the parameter $b_a$ of the Normalisation 2.
### Appendix B. Data Tables of the DAG Ranking Evaluation

<table>
<thead>
<tr>
<th>Category</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td><code>?v0 a dbo:Film . ?v0 dbo:director ?v1 .</code></td>
</tr>
</tbody>
</table>

Table B.3.: Queries of complexity 1-1-2, 1-1-3, 1-1-4, 1-1, 1-2, 1-3, 1-4, 1-5, 2-2, 3-4
Table B.4.: Queries of complexity 2
## Appendix B. Data Tables of the DAG Ranking Evaluation

### Table B.5: Queries of complexity 3

<table>
<thead>
<tr>
<th>Category</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Query</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 4        | \(?v0 \text{ a dbo:Country} . \?v0 \text{ rdfs:label } \?v1 . \?v0 \text{ dbo:countryName } \?v2 . \?v0 \text{ dbp:long } \?v3 . \?v0 \text{ dbp:lat } \?v4 . \?v0 \text{ dbp:timezone } \?v5 . \?v0 \text{ dbp:countryName } \?v6 . \?v0 \text{ dbp:long } \?v7 . \?v0 \text{ dbp:lat } \?v8 . \?v0 \text{ dbp:timezone } \?v9 . \?v0 \text{ rdfs:label } \?v10 . \?v0 \text{ dbo:countryName } \?v11 . \?v0 \text{ dbp:long } \?v12 . \?v0 \text{ dbp:lat } \?v13 . \?v0 \text{ dbp:timezone } \?v14 . \?v0 \text{ rdfs:label } \?v15 . \?v0 \text{ dbo:countryName } \?v16 . \?v0 \text{ dbp:long } \?v17 . \?v0 \text{ dbp:lat } \?v18 . \?v0 \text{ dbp:timezone } \?v19 . \?v0 \text{ rdfs:label } \?v20 . \?v0 \text{ dbo:countryName } \?v21 . \?v0 \text{ dbp:long } \?v22 . \?v0 \text{ dbp:lat } \?v23 . \?v0 \text{ dbp:timezone } \?

Table B.6.: Queries of complexity 4, 5, 6, 9, and 10
### Appendix B. Data Tables of the DAG Ranking Evaluation

<table>
<thead>
<tr>
<th>$b_v$</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Graphs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.8832</td>
<td>0.8834</td>
<td>0.8834</td>
<td>0.8834</td>
<td>0.8833</td>
<td>0.8833</td>
<td>0.8832</td>
<td>0.8832</td>
<td>0.8831</td>
<td>0.8831</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.8929</td>
<td>0.8927</td>
<td>0.8927</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.8925</td>
<td>0.8924</td>
<td>0.8924</td>
<td>0.8923</td>
<td>0.8923</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.8977</td>
<td>0.8977</td>
<td>0.8975</td>
<td>0.8975</td>
<td>0.8974</td>
<td>0.8973</td>
<td>0.8972</td>
<td>0.8971</td>
<td>0.8970</td>
<td>0.8968</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.9009</td>
<td>0.9009</td>
<td>0.9008</td>
<td>0.9008</td>
<td>0.9007</td>
<td>0.9007</td>
<td>0.9006</td>
<td>0.9005</td>
<td>0.9004</td>
<td>0.9001</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.9033</td>
<td>0.9033</td>
<td>0.9033</td>
<td>0.9033</td>
<td>0.9032</td>
<td>0.9032</td>
<td>0.9031</td>
<td>0.9031</td>
<td>0.9030</td>
<td>0.9029</td>
<td>0.9027</td>
</tr>
<tr>
<td>0.5</td>
<td>0.9059</td>
<td>0.9059</td>
<td>0.9058</td>
<td>0.9058</td>
<td>0.9057</td>
<td>0.9057</td>
<td>0.9056</td>
<td>0.9056</td>
<td>0.9054</td>
<td>0.9053</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.9085</td>
<td>0.9084</td>
<td>0.9084</td>
<td>0.9083</td>
<td>0.9082</td>
<td>0.9082</td>
<td>0.9081</td>
<td>0.9080</td>
<td>0.9079</td>
<td>0.9077</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>0.9111</td>
<td>0.9110</td>
<td>0.9109</td>
<td>0.9108</td>
<td>0.9108</td>
<td>0.9107</td>
<td>0.9106</td>
<td>0.9104</td>
<td>0.9103</td>
<td>0.9101</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.9130</td>
<td>0.9129</td>
<td>0.9128</td>
<td>0.9127</td>
<td>0.9126</td>
<td>0.9125</td>
<td>0.9124</td>
<td>0.9123</td>
<td>0.9121</td>
<td>0.9119</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>0.9151</td>
<td>0.9150</td>
<td>0.9150</td>
<td>0.9149</td>
<td>0.9148</td>
<td>0.9147</td>
<td>0.9146</td>
<td>0.9145</td>
<td>0.9142</td>
<td>0.9140</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.9191</td>
<td>0.9191</td>
<td>0.9190</td>
<td>0.9190</td>
<td>0.9190</td>
<td>0.9190</td>
<td>0.9189</td>
<td>0.9189</td>
<td>0.9189</td>
<td>0.9187</td>
<td>0.9185</td>
</tr>
</tbody>
</table>

| **Complex Graphs** |     |     |     |     |     |     |     |     |     |     |     |
| 0.0   | 0.8781 | 0.8789 | 0.8792 | 0.8795 | 0.8800 | 0.8804 | 0.8810 | 0.8815 | 0.8820 | 0.8829 | 0.8842 |
| 0.1   | 0.8800 | 0.8806 | 0.8810 | 0.8815 | 0.8820 | 0.8825 | 0.8830 | 0.8835 | 0.8841 | 0.8846 | 0.8854 |
| 0.2   | 0.8803 | 0.8809 | 0.8812 | 0.8816 | 0.8820 | 0.8825 | 0.8830 | 0.8835 | 0.8839 | 0.8844 | 0.8850 |
| 0.3   | 0.8806 | 0.8811 | 0.8814 | 0.8818 | 0.8821 | 0.8825 | 0.8829 | 0.8834 | 0.8838 | 0.8842 | 0.8848 |
| 0.4   | 0.8808 | 0.8813 | 0.8815 | 0.8819 | 0.8821 | 0.8825 | 0.8829 | 0.8833 | 0.8838 | 0.8843 | 0.8847 |
| 0.5   | 0.8810 | 0.8815 | 0.8817 | 0.8820 | 0.8823 | 0.8827 | 0.8831 | 0.8834 | 0.8838 | 0.8844 | 0.8846 |
| 0.6   | 0.8812 | 0.8816 | 0.8819 | 0.8821 | 0.8824 | 0.8828 | 0.8831 | 0.8834 | 0.8838 | 0.8842 | 0.8845 |
| 0.7   | 0.8814 | 0.8818 | 0.8820 | 0.8822 | 0.8825 | 0.8829 | 0.8833 | 0.8836 | 0.8839 | 0.8843 | 0.8847 |
| 0.8   | 0.8819 | 0.8821 | 0.8822 | 0.8824 | 0.8826 | 0.8829 | 0.8833 | 0.8836 | 0.8839 | 0.8843 | 0.8846 |
| 0.9   | 0.8821 | 0.8823 | 0.8825 | 0.8827 | 0.8829 | 0.8833 | 0.8836 | 0.8839 | 0.8842 | 0.8846 | 0.8849 |
| 1.0   | 0.8822 | 0.8826 | 0.8828 | 0.8830 | 0.8833 | 0.8836 | 0.8839 | 0.8841 | 0.8843 | 0.8846 | 0.8850 |

Table B.7.: Average MAP values over simple and complex graphs
Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


[ZLXH11] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph Cube: On Warehousing and OLAP Multidimensional Networks. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD ’11, pages 853–864, New York, NY, USA, 2011. ACM.

Glossary

**Attribute** $(\alpha)$ the label of an edge. 16

**Type attribute** $(\tau)$ an edge label that defines the type of the source node, e.g., rdf:type. 17, 18, 40, 43, 45, 47–49, 51, 55, 56, 60, 65, 67, 76, 79, 81, 131, 133, 135, 180

**Attributes** set of edge labels that are connected to a same node. 16, 35, 37, 41, 42, 48–51, 56, 69, 75–79, 129

**Incoming attributes** set of edge labels that have the same target node. 16, 84, 85

**Dataset label** $(l_G)$ function that assigns a label to a graph $G$. 19, 57, 59

**Entity description** $(\mathcal{E})$ the set of edges that describe an entity. 19, 52–54, 58, 59, 64, 65, 67, 93, 106

**Field** in the context of ranking, a field represents a characteristic portion of an entity. 8, 92, 93, 97, 121, 152, 153, see MF

**Graph summary** product of the graph summarisation. 180

**Attributes & Types summary** $(S_{at})$ graph summarisation based on the outgoing attribute and type features, taking the set of outgoing attributes and types for defining the summarisation relation. 49, 52, 71, 83, 84, 86, 87

**Attributes summary** $(S_a)$ graph summarisation based on the outgoing attribute feature, taking the set of attributes for defining the summarisation relation. 37, 48, 52, 69, 75, 79, 81, 83, 84, 86, 87

**Bisimulation summary** $(S_{fbt})$ graph summarisation which summarisation relation is based on the f&b-bisimulation relation taking also into consideration the set of types. 44, 72–78, 81, 83, 153

**FT-Bisimulation summary** $(S_{ft})$ graph summarisation which summarisation relation is based on the bisimulation relation taking also into consideration the set of types. 44, 45, 78, 82

**IO Attributes & Types summary** $(S_{ioat})$ graph summarisation based on the incoming/outgoing attributes and type features, taking the set of incoming/outgoing attributes and types for defining the summarisation relation. 51, 52, 71, 83, 84, 86, 87

**IO Attributes summary** $(S_{ioa})$ graph summarisation based on the incoming and outgoing attributes features, taking the set of incoming/outgoing attributes for defining the summarisation relation. 49, 50, 52, 81, 83, 84, 86, 87

**Types summary** $(S_t)$ graph summarisation based on the type feature, taking the set of type for defining the summarisation relation. vii, ix, 48, 52, 69, 71, 73–75, 81, 83, 84, 86, 87, 120–122, 133
Glossary

**Unique Type summary** \((S_{ut})\) graph summarisation based on the type feature, taking a single type for defining the summarisation relation. viii, 47, 52, 79, 81, 83, 84, 86, 87

**MF ranking model** generalisation of field-based ranking models for semi-structured data. 97–100, 103–114, 118–123, 139, 149, 152, 153, 161, 162
- **BM25MF** MF extension of the field-based ranking function BM25F. 100–103, 107, 109–111, 113, 161, 162
- **PL2MF** MF extension of the field-based ranking function PL2F. 100, 102, 103, 107, 109–113, 161, 162

**Node ownership** \((d)\) function that defines the ownership of an node by assigning a dataset to it. 57, 59

**Point Of Focus** the POF indicates the position in a SPARQL query for retrieving recommendations. 133, 136, 140

**Precision model** by comparing a Graph summary against a gold-standard summary, the precision of the former summary is measured. 69, 78

**Classification of errors** different kinds of misinformation deduced from the Graph summary due to the graph summarisation process.
- **attribute** category of error of a Graph summary that pertains to the attribute information of the original graph. 75, 84, 180
- **connectivity** category of error of a Graph summary that pertains to the original graph traversal. viii, ix, 75, 84, 86, 89, 90, 117, 152, 153, 157, 159
- **schema** errors of type and attribute categories. viii, ix, 89, 90, 152, 153, 157, 159
- **type** category of error of a Graph summary that pertains to the type information of the original graph. 75, 84, 180

**Lattice** ordering of the graph summaries based on a Partial order. 32, 70

**Partial order** \((\sqsubseteq)\) binary relation on the set of graph summaries. 70, 78, 80, 180

**summarisation relation** \((R)\) the relation that maps a node of the graph to a node of the summary. vii, viii, 39, 40, 44, 47, 51–54, 56, 58, 59, 61, 64–66, 69–72, 75, 77–81, 84, 85, 88, 89, 129–132, 151–153, 179, 180

**Types** set of labels of nodes that have the same source node and which edge label is a Type attribute. 18, 35, 38, 39, 42, 47–51, 53, 69, 78, 79, 92