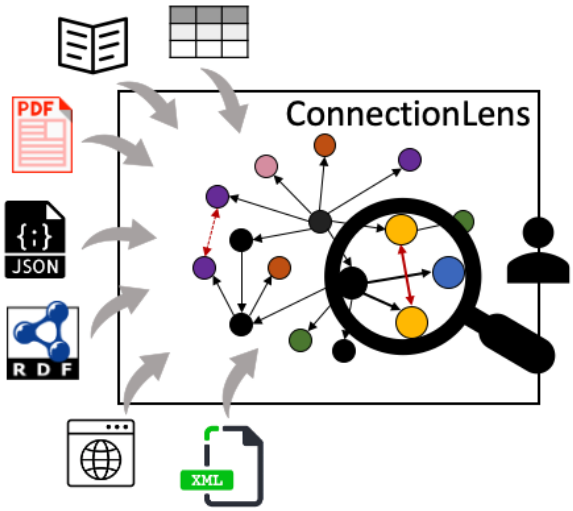
IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Inria

I.M.T. Nord Europe & Inria

Path-based ConnectionLens graph exploration

Engineering Assistant Internship (M1)



Antoine GAUQUIER - FISE 2023 - Academic year 2021/2022

Inria internship manager :
Ioana Manolescu

School internship manager :
Anas Adem

Acknowledgments

The success of this internship was only possible with the help of a certain number of people, whom I want to thank here.

I first would like to give my warnest thanks to my internship manager, Ioana Manolescu, research director at Inria and head of the CEDAR team. Indeed, she gave me the opportunity to carry out this internship and to trust me, both in my work, but also in my organization. She also always knew how to listen to my arguments and discuss our ideas objectively, allowing me to express myself. She has finally integrated me into the team and has always been available when I needed it.

Then I want to give a special thanks to Nelly Barret, a PhD student in the CEDAR team with whom I shared my office. She has always been available to discuss my problems or ideas, and has brought me a number of good practices and knowledge about the project, without which I certainly would not have been as successful in this internship.

I would also like to thank my I.M.T. Nord Europe internship manager, Anas Adem, who was always available if I ever felt the need. He was reactive at all times, and ensured that the internship was running smoothly.

This internship was also a success thanks to all the members of the CEDAR team, whom I would also like to thank. They gave me a good understanding of the context in which I had to work, and I learned a lot from their openmindness.

I would like to thank the administrative staff of Inria, with whom I was able to exchange during the entire integration process before the internship. These are always made available and very responsive, and have adapted to the modalities requested by I.M.T. Nord Europe.

Finally, I would also like to thank all those involved in the Inria Saclay Île-de-France centre, who have contributed directly or indirectly to the success of my internship.

Contents

List of Figures

Abstract and keywords

The purpose of this report is to summarize and present my internship, its context within the CEDAR team at INRIA's Saclay Île-de-France centre, the tasks I have carried out there and the learning and benefits I have gained from it. This internship is part of my second year of engineering at I.M.T. Nord Europe.

These four months of work have led to the realization of a research work allowing the exploration of large graphs of heterogeneous data, adopting an approach based on abstraction and extraction of a set of interesting paths. It is part of an application to investigative journalism, and has resulted in the creation of a functional application usable by them, without needing any particular a priori knowledge.

I also developed my knowledge of the world of research and its specificities, as well as scientific and technical knowledge in the field of artificial intelligence. This experience reinforces my desire to continue my studies and my career in the world of research in the field of artificial intelligence, by pursuing a PhD after my engineering studies.

Keywords: Inria - I.M.T. Nord Europe - Research internship - Investigative journalism - Artificial intelligence - Data graphs - Big Data

Ce rapport a pour but de synthétiser et présenter mon stage, son contexte de réalisation au sein de l'équipe CEDAR du centre Saclay Île-de-France de l'INRIA, les tâches que j'y ai effectuées ainsi que les acquis et bénéfices que je tire de ce dernier. Ce stage d'assistant ingénieur s'inscrit dans le cadre de ma deuxième année de cycle ingénieur à l'I.M.T. Nord Europe.

Ces quatre mois de travail ont abouti à la réalisation d'un travail de recherche permettant l'exploration de grands graphes de données hétérogènes, en adoptant une approche basée sur l'abstraction et sur l'extraction d'un ensemble de chemins intéressants. Il s'inscrit dans une application au journalisme d'investigation, et a aboutit à la création d'une application fonctionnelle utilisable par ceux-ci, sans avoir besoin de connaissances particulières a priori.

J'ai aussi développé ma connaissance du monde de la recherche et de ses spécificités, ainsi que des connaissances scientifiques et techniques dans le domaine de l'intelligence artificielle. Cette expérience conforte ma volonté de poursuivre mes études et ma carrière dans le monde de la recherche dans le domaine de l'intelligence artificielle, en faisant un doctorat à l'issue de mes études d'ingénieur.

Mots-clés : Inria - I.M.T. Nord Europe - Stage de recherche - Journalisme d'investigation - Intelligence artificielle - Graphe de données - Big Data

Introduction

As part of my second year of engineering studies at I.M.T. Nord Europe, I had the opportunity to do a four-month internship as an assistant engineer from May 9 to September 5, 2022, with the CEDAR team at Inria's Saclay Île-de-France centre.

This internship aims to develop our skills in our field of specialization in a context of taking responsibility and working independently. Specializing in the Digital field, and more particularly in artificial intelligence and data science, I wanted through this internship to discover the world of academic research and have a first experience of it. So I naturally turned to a research internship in this field. I then applied to several research laboratories, and Ioana Manolescu, Inria research director and head of the CEDAR team, gave me this opportunity.

The CEDAR team works, among other things, on the integration of heterogeneous data in large volumes in graphs. Several tools have already been developed in this regard. In particular, the ConnectionLens project, which has been in development for several years, allows the integration of various data sources, independently written, from different types of files, into a single graph. This tool is particularly interesting for the field of investigative journalism, where more and more data are available, making their analysis on a human scale only complicated. Thus, it is necessary, in addition to the generated graphs, to be able to explore them on the one hand, and to be able to automatically bring out the most relevant information on the other hand. Much work has already been done or is in the process of being done on the query of the ConnectionLens graphs, but few studies have been carried out to enable guided and automated exploration of these graphs.

The need to allow investigative journalists to analyse more and more data combined with the absence of a system allowing a semi-automated analysis of the exploration of ConnectionLens graphs led to the realization of this internship. In particular, the subject of my internship is as follows. Given a ConnectionLens data graph, and from the already existing literature on the exploration of heterogeneous data graphs, the aim of the internship is to develop a theoretical model and a tool for guided or automated exploration of these generated data graphs.

In order to meet this objective, this report will be divided into three parts. The first part will present the context of the internship, the host structure and the context of the ConnectionLens project. Secondly, I will describe the chosen approach to respond to the problem, the objectives and the organization of the work, and then its concrete implementation, step by step. Finally, I will summarize in a final part all the achievements and benefits I have gained from this experience, both technically and humanly. I will also describe the contribution of this experience in the construction of my professional project, and what I plan to do for the rest of my studies.

Chapter 1

Context in which the internship is carried out

This chapter aims to present the context in which this internship was carried out. More specifically, the first section of this chapter will present the working environment, from the research institute to the research team. The second section will present the background of the work of this internship.

1.1 Working environment

My internship took place within Inria, in the centre of Saclay Île-de-France, and more particularly within the CEDAR team. This section aims to present the environment and the context in which this work is taking place. Specifically, in the first section, we present what is the Inria. In the second one, we present the Saclay Île-de-France research center, and in the last one, the CEDAR team, in which I worked.

1.1.1 Presentation of Inria

Inria is a French national public research institute in digital science and technology. Composed of more than 3,900 engineers and researchers, the institute is divided into 10 research centres, which are divided into 215 project teams, joined by Inria and French higher education stakeholders (universities and engineering schools).

Inria therefore aims to develop and promote research in new technologies, but also has an important entrepreneurial role since the institute is at the origin of more than 200 start-ups. As a public institute, it also carries out public missions of national interest, such as the affirmation of national digital sovereignty, as well as the response to the challenge of digital transformation.

Inria focuses on twelve research themes: algorithms and quantum computing, high-performance computing, education and digital, artificial intelligence, the Internet of Things, software, modelling and simulation, frugal digital, robotics, digital health, data science and digital security.

Inria is also recognized for its operation in project teams. Indeed, each project-team is initially created in response to a real scientific need, whether it is industrial or academic. After approval by a commission, the project team is operational for four years. At the end of this period, a new commission meets to present the progress of its work and what it plans for the future. If it can convince the commission, the team is then renewed for a further four-year period. In total, a team can be renewed twice, and therefore has a maximum lifespan of twelve years.

This system makes it possible both to ensure that the research work of the team always meets a real scientific need, but it also makes it possible to reorient the research beam if necessary (either during a team

renewal or when a new team is created).

1.1.2 The Saclay Île-de-France research center

The Saclay Île-de-France research centre was established in 2008 and is one of Inria's 10 research centres. This centre is in partnership with two leading academic actors: the Université Paris-Saclay and the Institut Polytechnique de Paris (itself composed of five major engineering schools: Télécom Paris, Télécom Sud-Paris, ENSTA, ENSAE and École Polytechnique).

This centre led by Jean-Yves Berthou is composed of 34 project teams, mobilizing 600 people from 54 different nationalities.The organizational chart of the centre's management is shown in Figure 1.1. Its activities are structured around four scientific axes: digital health, artificial intelligence and data science, cybersecurity, evidence and verification, as well as quantum computing.

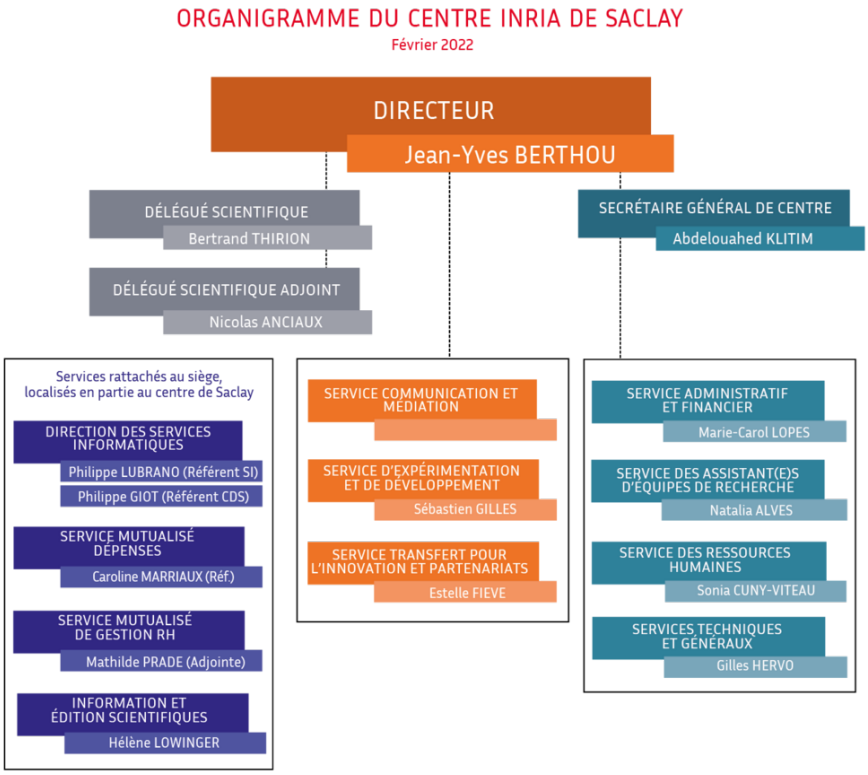


Figure 1.1: Organization chart of the Inria Saclay centre Île-de-France (source: Inria Web site)

The centre also has other strategic partnerships, which are: actors in public research, such as the CEA, CNRS, CNES, INRA, INSERM and IFPEN, private players such as Airbus, Boeing, Dassault Systèmes, EDF, Google, Microsoft, Thales, Orange and many others and international academic partners, such as Stanford University, UC Berkeley, Keio University, and others.

The centre is mainly located on the campuses of the Institut Polytechnique de Paris (Palaiseau), as well as on the campus of the Université Paris-Saclay (Gif-sur-Yvette). Its main address is the Alan Turing building, located 1 rue Honoré d'Estienne d'Orves (Palaiseau) on the campus of the Ecole Polytechnique. This building is also that of the LIX (computer lab of the Ecole Polytechnique), to which several project teams are attached to.

1.1.3 The CEDAR team

One of the 34 project teams in the Saclay Île-de-France centre is the CEDAR team, which stands for Rich Data Analytics at Cloud Scale. This is a joint project team between Inria Saclay Île-de-France and LIX, led by Ioana Manolescu, Inria senior researcher, who is my internship manager. The CEDAR Team visual identity is shown in Figure 1.2

The team's activities revolve around two main themes. The team's primary focus is on parallel data processing infrastructures for highly scalable, parallel Big Data storage and processing tools. The studied topics include highly efficient query processing, efficient query answering in the presence of ontologies, and algorithms for scalable, fast data analytics. The team also study new paradigms of user interaction with Big Data, based on exploratory querying, analytics for semantic graphs, and intuitive query tools over highly heterogeneous data.

Several academic and industrial partners are involved in the team's activities. There are other Inria project teams: ILDA, TYREX, Graphik and Zenith. Researchers from the Ecole Polytechnique, Télécom Paris, CentraleSupélec, INSA Lyon and the University of Rennes collaborate with the team. The team also works with industrial and non-academic partners, such as Amazon for the part of the team working on optimization topics, or Le Monde and Radio France for the part of the team working on the heterogeneous integration of large volumes of data, applied to investigative journalism.

The different research activities take the form of different projects. The work done during this internship is a continuation of one of these projects, called ConectionLens. This project is itself a component of another larger project, which is SourcesSay. SourcesSay is an AI chair funded by the Agence Nationale de la Recherche (ANR) and the Direction Générale de l'Armement (DGA), in 2020-2024. The aim of the project is to interconnect data sources of all kinds in digital arenas. In an arena, a data set is stored, analyzed, enriched and connected, by doing chart extraction, machine learning and visualization techniques, to build powerful data analysis tools.

The ConnectionLens project aims to integrate several heterogeneous, independently authored data sources in a single graph. It is particularly suited workloads that explore connections across the data sources, across different data formats, such as data journalism projects. To discover connections across data sources and enhance their value for the user, ConnectionLens leverages Information Extraction (Named Entity Recognition) and Named Entity Disambiguation (this will be explained later).

Figure 1.2: CEDAR team visual identity (source: CEDAR Web site)

1.2 Background of the work

In this section we lay out the context in which this work took place and describe the state of the system before my internship, and the objectives to fulfill starting from this system. Specifically, in the first section, we present and explain what are ConnectionLens graphs. In the second one, we precise how we can retrieve information out of these graphs.

1.2.1 ConnectionLens graphs

As lightly described above, the ConnectionLens project aims to generate graphs builded from several heterogeneous data sources. These graphs are builded as described in [ABC+21].

First, ConnectionLens is able to integrate structured data sources (CSV - Comma-Separated Values - files, SQL - Structured Query Language - databases), semi-structured (RDF - Resource Description Framework -, JSON - Javascript Object Notation- , XML - Extensible Markup Language - or HTML - HyperText Markup Language - files), and non-structured (raw text).

Then, each data source allows the generation of a graph of its own, and more precisely, of a tree whose root corresponds to the data source. Then, depending on the nature of the integrated data, branches develop from this root. For structured and semi-structured data, a data tree already exists and can therefore be integrated into the graph. For other data sources, machine learning modules allow the extraction of named entities: this is referred to as Named-Entity Recognition (NER) tasks.

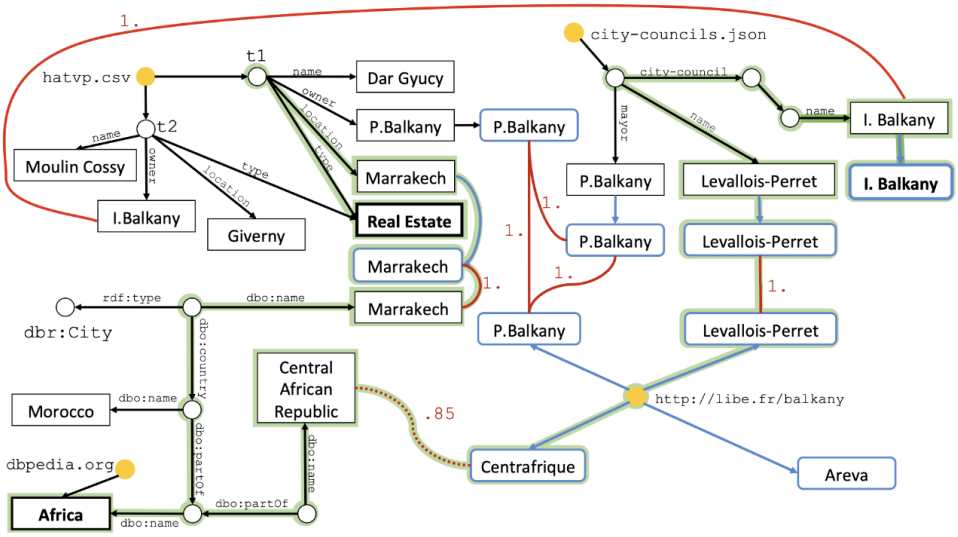


Figure 1.3: Example of ConnectionLens graph generated from four different data sources (from [ABC+21])

These modules make it possible to analyze texts and deduce from them entities such as people, locations, organizations, email addresses and others, but also to generate links between these named entities extracted. Whether they are used on raw texts or on already semi-structured data, their use allows the creation of entity nodes. These entity nodes are essential because they are the ones that allow to link each graph associated with a data source in the same common graph: this is what ConnectionLens is generating. Sometimes the named entities extracted do not have the same name (we speak of a label), but they have similarities. In this case, a connection is still generated, but a similarity score is associated with the connection to characterize its confidence. An example of such a graph created from four different data sources is shown in Figure 1.3.

In the example shown in Figure 1.3, we can clearly see the four data sources represented by yellow dots. From these, depending on the type of data source and the NER result, several trees are obtained. These trees

are then linked together by the named entities extracted. For example, the data sources "*hatvp.csv*", "*city-councils.json*" and "*http://libe.fr/balkany*" are linked by a named entity extracted, which is "*P.Balkany*", and which is a person. On the other hand, the data sources "*dbpedia.org*" and "*hatvp.csv*" are linked by the entity named extracted "*Marrakech*", which is a city and therefore a location.

We observe that thanks to these different connections between the entities named extracted, the graph generated by ConnectionLens integrates these different data sources.

1.2.2 Getting information out of ConnectionLens graphs

There have been several efforts in the team with the aim of finding interesting information within ConnectionLens graphs. In particular, there are two main ways to do so. The first sub-subsection presents the querying of ConnectionLens based on keywords, and the enumeration of interesting paths sets in the second one.

ConnectionLens querying based on keywords

Although ConnectionLens is able to generate graphs from different data sources, these cannot really be apprehended by a human in the state, especially because of their rapidly growing volume.

A keyword query system has therefore been set up, so that an algorithm can return much smaller sub-graphs of the ConnectionLens graph, and especially containing information relevant to the user's input. These sub-graphs are actually trees that are called answer trees.

We can build in Figure 1.3 several examples of answer trees from different keywords. By choosing the keywords "I. Balkany", "Africa" and "Estate", we get an answer tree highlighted in green on the graph.

A keyword query many lead to answers, and the system must then decide which answers are *the most interesting (the best)*. This is usually done with the help of a **score function**. ConnectionLens describes one in [ABC$^+$21] which is implemented, and is currently the default.

In general, the problem of keyword search in databases has been studied in a very large number of works. An 2021 intern (Lucas Maia Morais) has carried a survey, described in his report [Mor21]. Each of the 30+ surveyed works claims that their score function is better than the previous ones, at least in some cases. A difficulty that is still left for users is that there are many score functions, each with its own advantages and disadvantages. It is hard for a user (especially the non-computer scientists) to pick the function that corresponds to his need.

This has brought the idea that maybe we can **learn the right score function from some examples provided by users**, among the answers to a given query. An intern had worked on this problem in 2020, and his work is described in the report [Zha21]. He experimented with a relatively large RDF graph and an early version of ConnectionLens, and in the end, he loaded the graph in memory using a Python library. The most advanced results that were obtained show that an SVM classifier can learn (a) a specificity-based score and (b) a PageRank-based score.

Enumerating interesting path sets

A different approach consists of not only comparing a set of paths that answer to a given query, but characterizing the whole graph through its paths, in particular those that we find interesting (worth mentioning). Given the data journalism applications we consider, the interesting elements are people, places, and organizations that appear in the data. Therefore, one may be interested to find the possible paths that connect an entity pair (i.e., two extracted named-entity) having certain types. For instance, entities of type Person with entities of type Organization. In turn, this can be done in two different ways.

Firstly, we could characterize such paths directly from the graphs generated by ConnectionLens, as in Xin Zhang's report. But as already explained, the graphs generated by ConnectionLens are very quickly extremely voluminous. Thus, even if we limit ourselves to the exhaustive enumeration of the paths that bind a given entity pair, we would be limited at the same time by a concern for the performance of the system itself that would have too many paths to consider, but we would also be limited by the interpretability of the results by a human being, as we would quickly be faced with millions or even billions of paths.

Therefore, to obtain better performance, instead of manipulating the whole graph, we could work instead with a (much smaller) *summary* or *abstraction* of the graph. A graph summary is a compact description of a graph, oftentimes many orders of magnitude smaller than the graph itself. Usually, every path that exists in the data, also exists in the summary (and usually only once). Enumerating the paths on the summary is thus likely to be faster. The CEDAR team has developed RDFQuotient, a tool for summarizing RDF graphs, that can be found here : `https://rdfquotient.inria.fr/`. More recently, as part of her PhD `https://team.inria.fr/cedar/projects/abstra/`, Nelly Barret has already done some work on summarization and abstraction of CollectionLens graphs.

She has implemented a simple summary for a simple ConnectionLens graphs that consists of *exactly one XML dataset*, and similarly, for a graph that consists of *exactly one JSON dataset*. She also has adapted RDFQuotient so that it can be used to compute a summary of a ConnectionLens graph *that consists of exactly one RDF dataset*. Hence, Abstra, the name of the project corresponds to this abstraction and summarization, is capable like ConnectionLens of managing different types of data sources. This project views any of these summaries as a *collection graph*. This means that we could start from the collection graph to enumerate the paths. An example of a collecton graph is shown in Figure 1.4.

This second approach (path enumeration based on the summary) is the one we follow in this work. This is the best choice because it has several advantages. First, it solves the performance concerns of a path enumeration directly on the CollectionLens graph. Then, enumerating paths directly on a collection graph allows to present a number of results apprehendable by a human. And above all, this choice allows to work in a second time on a family of paths that has similar characteristics.

For instance, we can use Figure 1.3 to link the Balkany family to real estate they own in Africa. Thanks to an approach based on a collection graph, it would be possible, for example, to link any deputy or mayor to his or her real estate holdings abroad (of course, having the data sources to allow this beforehand).

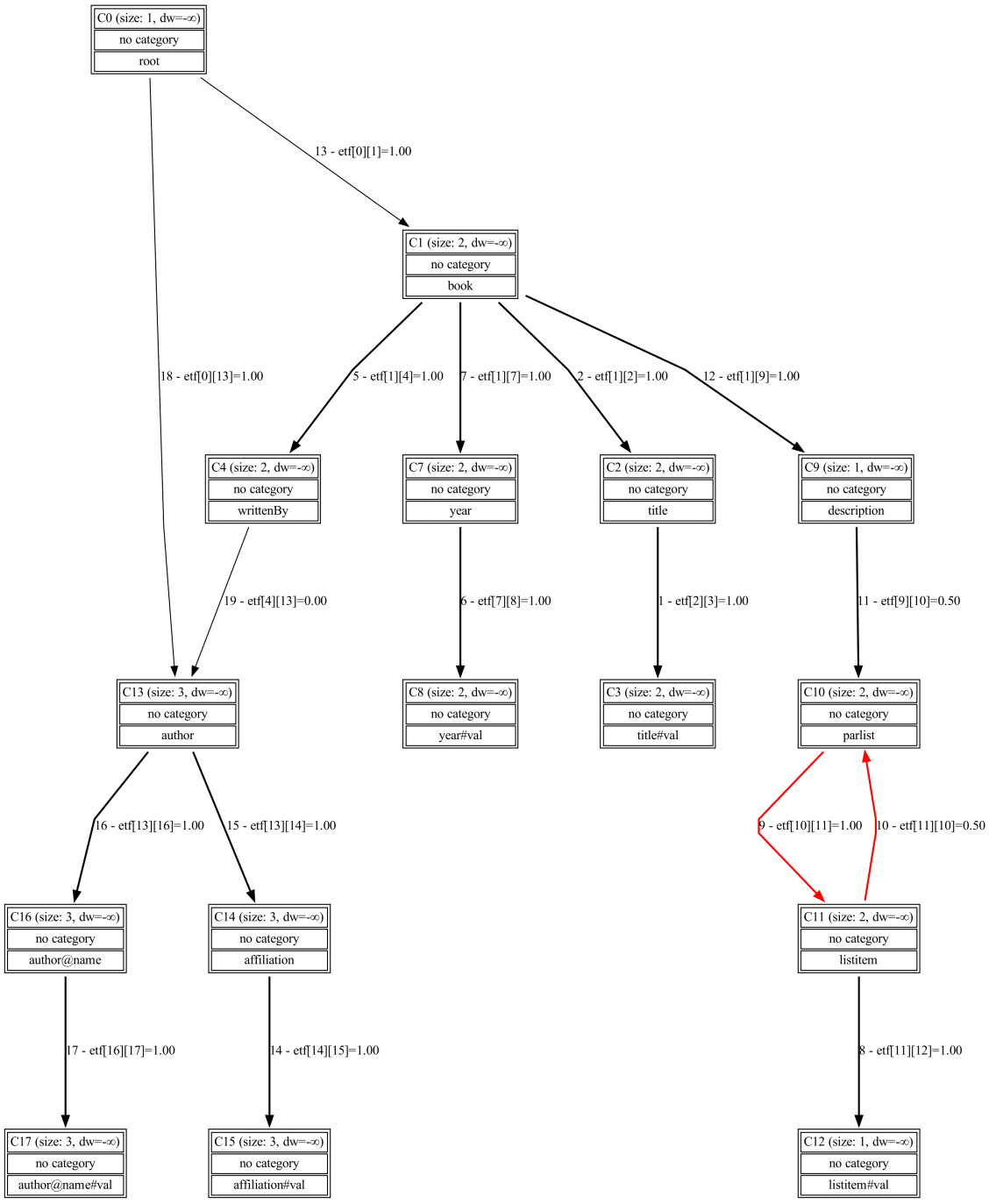


Figure 1.4: Example of a collection graph from Abstra, describing data about books. Image automatically generated by the Abstra tool.

Chapter 2

Approach, objectives and realization of the work

In this chapter, we present the scientific reasoning that led to the completion of the work done during the internship. The first section presents the approach used, the objectives pursued and the organisation. The second section presents the execution of the work step by step, as defined in the first section.

2.1 Approach, organization and objectives

In this section, we discuss the formalisation of the problem to be solved, the objectives to be achieved and the organization put in place to achieve them. Specifically, we formalize a problem statement in the first sub-section, then we set the objectives and constraints on the work on second one. The third one presents a synthesis of the state of the art's study, and the main ideas we kept. Finally in the last one, we describe the organization of the work.

2.1.1 Problem statement

The problem to be solved during this internship can therefore be formulated as follows: given a collection graph constructed from graphs of real data from ConnectionLens, we want to determine a set of paths that characterize this collection graph.

More precisely, we want these paths to be "the most interesting" for the person who explores the graph data (which we will call the user below, in the sense of user of the application that allows to obtain this set of paths). We have identified two ways to define the interestingness. Either the user wants to have a synthesis of the collection graph and/or the data graph, and is interested in the most frequent paths in the graph, or on the contrary the user wants to bring out from the graph the least usual connections, namely the rarest.

This second case is the most recurrent in the field of application that is investigative journalism. For instance, if one looks at the data of papers and their authors in the biomedical field, a connection between an author who declares not having a conflict of interest and a pharmaceutical company will be very rare, but this is what the journalist will look for. In this work we will therefore focus on this paradigm.

2.1.2 Objectives and constraints

The primary purpose of this work is to advance research in this particular area. Determining the set of paths that characterize a heterogeneous data plot constructed from various sources is a recurrent problem, and approaching it using an abstraction plot is a new approach. But beyond research, this work must allow the creation of an application within the framework of investigative journalism. At the end of the internship,

this application will be used by journalists as part of their activities to assist them in their work.

These requirements lead to a certain number of constraints, which we will have to follow throughout this work.

- First, it is necessary to ensure that the interface and the results displayed by the application will be understandable for a non-expert user (i.e., a user who is neither an expert in computer science nor in the field of research associated with the work carried out).
- Second, given the nature of the data being processed, the application must be able to ingest large amounts of data in a reasonable amount of time.
- Another constraint that will have to be followed throughout this work is to ensure that the temporal complexity is kept as low as possible. A reasonable goal is to maintain a linear complexity at each stage of the algorithm.

2.1.3 Summary of the study of the state of the art and main ideas

In order to be able to carry out this work effectively and within the time allowed, developing the project required a preliminary study of the state of the art.

This not only allows us to have a global vision of what has already been done in the areas of research associated with our work, but above all to have starting points on which to build in order to start thinking. A synthesis of these observations should then be made and the most relevant ones selected in order to draw inspiration from them to develop the project, while adding our own ideas.

Several articles have been read and synthesized for this study; here, we outline the main ideas that emerged from them. Later in this report, we will explain how these ideas and the technical adaptations that were necessary to move from the idea to the concrete have been put in place. All references to the work studied in this context are available in the bibliography of this report [ABC+21], [BCL15], [DAB16], [LMPV20] [LF06], [LPHM20], [YLW+22], [Des20].

First, although working on a collection graph drastically reduces the number of existing paths in the graph, it is still too computationally expensive to simply list all the paths linking named extracted entities. Therefore, we have to find criteria for exclusion, in order to reduce this number of paths. But since we want to make sure that the extracted paths are as relevant as possible to the user, regardless of what the data is about, we cannot simply set general exclusion criteria. Instead, we find in the literature a great use of user interaction, which allows to put the user at the heart of the process of exclusion of the least interesting paths.

However, given the size of the data processed, it is necessary to use interaction sparingly in order not to drown the user of questions. That is why we must first make sure that we pre-treat our exclusion proposals by classifying them according to the ones that we think are the most relevant, and then propose them to the user. We will still give the user the freedom to choose the number of answers he wishes to give, for two purposes. First in order to be as precise as possible if he wishes (indeed, since we are going to classify the proposals according to a heuristic, we cannot ensure that they are indeed the best), but also for the purpose of choosing how far he wishes to exclude possibilities.

We also observe, through the literature but also from what has already been done for ConnectionLens, that making an exhaustive enumeration and then applying a number of exclusion criteria will require too many resources, as mentioned above. To solve this problem, we have to think in the other direction: we will first, using the information available in the collection graph at a summary level (that is, without getting into the data level, that corresponds to the paths), perform user interaction to define the exclusion criteria, then in a second step, create an enumeration algorithm that takes these criteria into account in order to directly do the exclusion during enumeration.

2.1.4 Organization of the work

Thus, we can organize the execution of our work in different stages, according to the observations that have been made previously :

1. Implementation of exclusion criteria and the paradigm of the "rarest paths".
2. Use of user interaction, and establishment of a user interface usable by non-experts.
3. Path enumeration algorithm (taking into account the exclusion criteria), and calculating path supports (defined later).
4. Performance of the work through scale-up experiments and time-complexity calculations.

2.2 Realization of the work step by step

In this section we discuss the realization of the work, namely the development of the ideas explained in the previous chapter, and the technical realization of these ideas. This section is composed of four sub-sections, corresponding to the four steps presented above. The first sub-section presents the exclusion criteria and the use of the "rarest connections" paradigm. The second one presents the application's interface and how we integrated the use of user feedback. Then, the third one presents the path enumeration algorithm and the computation of path supports. Finally, the last one presents the experimental evaluations and the performance analysis.

2.2.1 Exclusion criteria and use of the "rarest connections" paradigm

The first step of this work is to develop exclusion criteria that have the objective, as explained in the previous chapter, to reduce the number of paths to be listed while ensuring to meet the needs of the user.

In this subsection, we present the four criteria we set up. The first sub-subsection presents the maximum path length and the number of paths limitation, and the second one the choice of a pair of named entity types. The third one is about the exclusion of intermediate collections, and the last one about the restriction of paths inversions.

Maximum path length and number of paths

In order to limit the number of paths generated by the enumeration algorithm, we give the user the freedom to choose, in advance, the number of paths he wants to obtain at the end of the enumeration. Concretely and technically, it just need to add a condition in the enumeration algorithm to stop it if the maximum path number has been reached.

Still with the idea of limiting the number of enumerated paths, we also allow the user to choose the maximum size of the enumerated paths. This size corresponds to the number of edges present in the path. Therefore, if the user chooses a path size of n, then the enumerated paths will contain between 2 and $n+1$ collection graph nodes. Indeed, since an edge links two nodes, n edges lead to $n+1$ nodes. By extension, a path containing only one edge will contain two nodes.

Choice of a pair of named entity types

Still with the aim of proposing paths that are best adapted to the needs of the user, we decided to restrict the enumeration to linking two types of named entities forming a pair. In particular, this choice is all the more relevant as this application will be used by journalists, who often have an idea of what they are looking for. Indeed, if we take the example used in the previous chapter on the data of scientific articles in the bio-medical field, the journalist would only have to select to link the types *ENTITY_PERSON* and *ENTITY_ORGANIZATION*, which would give paths linking collections whose associated data are (at least partially) associated with entities named as *ENTITY_PERSON* to collections whose data are associated

with entities of type *ENTITY_ORGANIZATION*. Of course, not all paths linking these collections would necessarily be displayed, based on the other exclusion criteria chosen by the user.

In order to choose a pair of types, they must first be identified. This involves identifying the collections in whose values (strings), named entities have been extracted. This will then be referred to as a *collection signature*.

Signature retrieval has already been done by Nelly Barret as part of her Abstra project[1]. All that remains is to recover the signatures, and reassign them one by one to each collection to which it belongs. Sometimes the same collection contains data associated with different types of extracted entities. This case is particularly common for collections that group data related to free text, such as a description for example, in which we can find names, locations, emails, etc.

Since we want to enumerate on the collection graph and stay as close as possible to a summary level, we cannot check, for each collection graph path linking collections associated with several types, what type of entity among those identified at the collection level is actually associated with the data path. Therefore, we have decided to consider that a collection can have several types, and therefore, for two pairs of different entities, it is possible that one finds the same path. In order to better understand this principle, an explanatory diagram is shown in Figure 2.1.

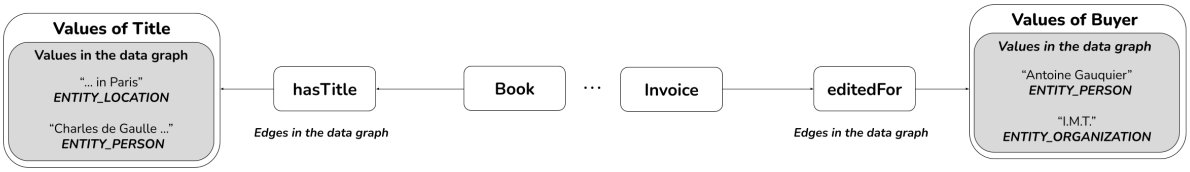


Figure 2.1: Example of two target collections associated to different types of named-entities in a path

As shown in Figure 2.1, we have a path that is linking two collections. The one called " *Values of Title*" is associated with the types *ENTITY_PERSON* and *ENTITY_LOCATION*, whereas the one called " *Values of Buyer*" is associated with the types *ENTITY_PERSON* and *ENTITY_ORGANIZATION*. Therefore, we would find this path for different pairs of types : Persons and Persons, Persons and Organizations, Locations and Persons, Locations and Organizations.

Exclusion of intermediate collections

Now that it is possible to limit the length of the paths, their numbers and the types of entities they link, we can think about exclusion criteria relative to the inside of the paths, and therefore relative to the collections they contain.

A simple solution to reducing the enumeration is to make a list of prohibited collections, namely collections that you will not want to find in the paths. But we cannot randomly exclude collections, and this is why we decided to focus on prioritizing the paths that, in the data level, are the rarest, and therefore the most interesting for journalists.

As we seek to exclude collections while keeping the rarest paths in the data graph, it is relevant to exclude collections that are associated (in the data graph) with edges that are the most common. By this, we mean the edges through which a large number of paths of the data graph go through. By doing this, one will exclude by extension the most common data paths, which will necessarily bring out the rarest paths.

[1]Project Web site: `https://team.inria.fr/cedar/projects/abstra`

To successfully identify collections with these properties, it is necessary to be able to quantify whether the edges of the associated data graph are involved in many paths or not. To do this, a metric is already defined on the edges by ConnectionLens [ABC+21]: this is the *edge specificity*. For a given node n and label l, let $N^l_{\rightarrow n}$ be the number of l-labeled edges entering n, and $N^l_{n\rightarrow}$ the number of l-labeled edges exiting n. The *specificity* of an edge $e = n_1 \xrightarrow{l} n_2$ is defined as:

$$s(e) = 2/(N^l_{n_1\rightarrow} + N^l_{\rightarrow n_2}).$$

To give an example of the use of this formula, let us consider all the edges labeled *hasWritten*. Let us consider that this edge is incoming for the nodes *Author1*, *Author2* and *Author3* and *Author4*, and that it is outgoing for the nodes *Book1*, *Book2* and *Book3*. In this case, we will therefore count 4 incoming and 3 outgoing edges for all the edges labeled "hasWritten". So we would have a specificity of $s(e) = \frac{1}{3+4} \approx 0.143$. Thus, the more an edge set has incoming and outgoing edges, the lower the edge specificity, and therefore the more common the edge set is in the data graph. More precisely and in theory, this edge specificity is between 0 (excluded) and 1. But in practice, it will be a maximum of 0.5, because each edge of a set of edges has an incoming and an outgoing representative, since it links two nodes in the data graph.

We can therefore use this metric to identify the least interesting collections. To do this, we have decided, for each collection whose associated data are edges, to take the edge with the weakest specificity as the representative edge specificity for the collection. It is then sufficient to rank these collections by increasing specificity to obtain the collections that seem to be the least interesting in our exploration of the collection graph. An example is shown in Figure 2.2.

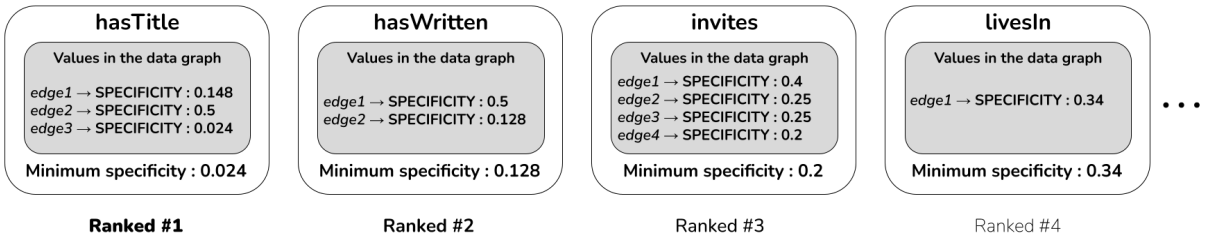


Figure 2.2: Example of edge-specificity usage in collections exclusion

Again, we won't do an exclusion without involving the user. The purpose here is to propose to him the collections that seem to us to be the least interesting. User interaction will be discussed later in this report.

Inversions in paths

Inversion in paths is something that is not considered in the data graphs generated by ConnectionLens, as it has no interest, and overloads in a path several times the same information.

However, we cannot make the same conclusion for collection graphs, as a collection actually contains several data that represent different information. This can be understood through a very simple example: let us consider the path shown in Figure 2.3, coming from a collection graph.

As we can see in (a), we included in the collection graph's path an inversion : we go twice through the collection "*hasWritten*" (which is a collection coming from edges in the data graph), to twice reach the collection "*Paper*". But this is only a path in the collection graph, and what really matters is what can be found at the data level. Therefore, if we take a look at the corresponding possible paths at this data level, we see the path that is shown in (b). This example shows that, at the data level, we do not have an inversion anymore, since different representatives of the collection "*Paper*" were found. Concretely, this example, took in the context of scientific papers, corresponds to the example where a same author wrote two different papers.

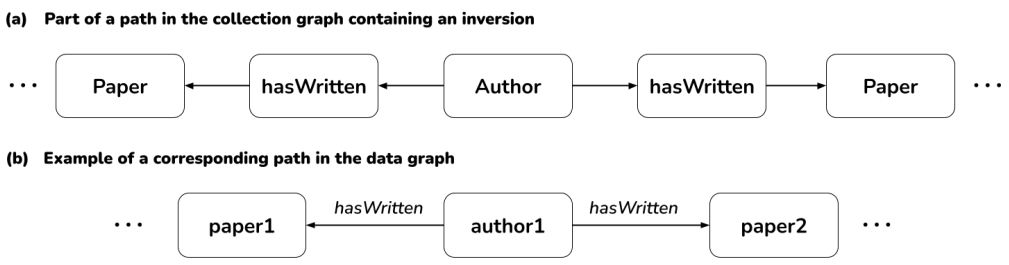


Figure 2.3: Example of part of a path with an inversion in a collection graph (a), and part of one of its associated path in the data graph (b)

The inversions in the paths are therefore relevant for the collection graph. However, allowing inversions wherever possible in the graph will be too expensive, as it will result in the generation of too many paths, which we want to avoid.

Thus, instead of allowing all of them or even making a blacklist, we will only allow certain inversions that are consistent and that seem to be the most interesting (via a white-list therefore). Thus, we must consider two criteria in order to accept a reversal. First of all, it has to be consistent, that is to say that, technically, inversion is possible. We have seen in Figure 2.3 an example of potential inversion, which makes sense in the collection graph. However, if there are not two data from the "*Paper*" collection that are accessible from a common data from the "*Author*" collection, then the inversion no longer makes sense, as it will not correspond to any situation at the data level.

Once this property is checked, it is necessary to select, among these inversions, those that seem most interesting for the user, always according to the paradigm of rarity defined above in this report. In the same way as for the exclusion of collections, we will rank the inversions so that those that seem to us most relevant are the first ones proposed to the user, but only his intervention will allow the authorization of an inversion.

To do this, we use the same principle as for the exclusion of collections, but this time via a slightly different metric. Inversion is always done around collections that come from edges. Let's go back to the example in Figure 2.3: the "*hasWritten*" collection is a collection of edges that allows us to make an inversion around the "*Author*" and "*Paper*" collections. In order to rank the edge collections, and by extension the resulting inversions, we decided to count, for each of these collections, the number of corresponding edges in the data graph. The larger the number, the more common is to have a path going through such a collection, and the less interesting the collection. We therefore first expose the user to inversions with a collection of edges with the lowest number of edges at the data level, making them rarer and therefore more interesting.

2.2.2 Use of user feedback and application's interface

We have seen the different exclusion criteria to reduce the number of paths to be enumerated while preserving the interests of the user. We must now be able to interact with him so that we can make suggestions and ask him questions, and especially so that we can record his answers in order to set up the enumeration that will result.

This requires on the one hand to use the input and output of the program to communicate, presented in the first sub-subsection, but also to create an interface clear enough for a non-expert user to use it without problem, explained in the second one.

Exchange of information with the user regarding the exclusion criteria

In this sub-subsection, we will review each of the exclusion criteria considered in section 8.1, and explain how we interacted with the user to integrate his feedback. We will also present an example of a concrete

interaction in the application.

Since we are dealing with professional and conscientious users, we have considered throughout our work that users respect the rules defined in the statements that precede the feedback. Thus, no protection mechanism against entries that do not comply with the rules is put in place. Now considering the security of entries against potential attacks, the only data processed by the algorithms are those provided by the user, and they are stored locally, and also overwritten with each new use of the application. There are no risks at that level.

- Maximum path length and number of paths

In order to retrieve the maximum number of paths and the maximum length of the paths desired by the user, it is enough to write in output the question asked, and to wait in input the integer number corresponding to the user's choice. Specifically, a natural integer is expected, and there is no high limit or maximum value. Thus, if a user enters 0 to at least one of the two criteria, no path will be enumerated. On the other hand, the user can enter an arbitrarily large number if he wishes respectively not to limit the number of paths or their maximum size. An example of feedback is shown in Figure 2.4 for these two criteria.

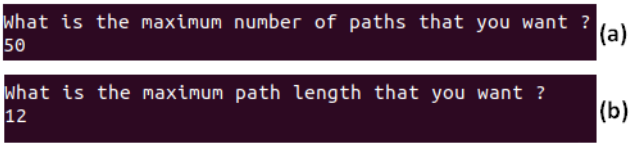


Figure 2.4: Example of a feedback for number of paths limitation (a), and for path length limitation (b)

- Choice of a pair of named entity types

Thanks to the work of identifying the named entity types associated with the collections present in the graph carried out previously, we already have at our disposal the list of entity types. All we have to do is display it to the user, then ask him to return the entity pair he wishes to see linked by the paths, so to choose two entities from this list. To do this, the list is displayed in such a way that each type of entity is presented with a number. Then, two successive questions are given to the user, who successively ask him the number of the first type of entity and the second type of entity. It is important to note that the order in which the types are chosen does not matter because it is a pair. The user also has the option to choose the same entity type twice if desired. An example of this feedback is shown in Figure 2.5.

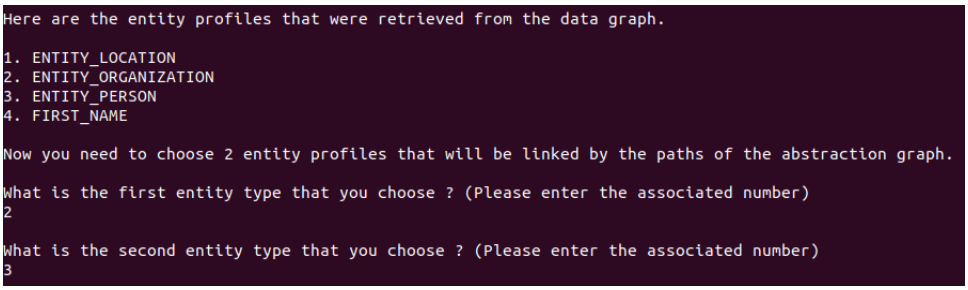


Figure 2.5: Example of a feedback for the choice of the pair of named entity types

- Exclusion of intermediate collections

User interaction becomes a bit more complex for the retrieving of collections to be excluded. We already know how to retrieve them and to rank them according to the criteria that have been defined previously. So it remains to expose them to the user, and to collect his wishes. To do so, we start by asking him how many collections he wants to label. This allows both to make a precision sorting if desired, and to gauge

the degree of exclusion. The user is asked a first question, asking him how many collections he wishes to label. A natural integer is expected in return. If the user enters 0, then no collection will be excluded and the enumeration will be done on the whole graph. If he or she enters a greater number than the number of collections retrieved by the algorithm that can be labeled, then the number entered by the user will be replaced by the maximum number of collections that can be labeled. Let us denote this number n.

Now, the user will have to answer n questions, which will correspond to the exclusion (or not) of the most interesting n collections, as defined earlier. For each question, the name of the collection is given to the user, and he must enter 1 if he wants the collection to be found in the paths, and 0 otherwise. It is more natural to describe things this way, instead of asking to enter 1 if he wants the collection to be excluded and 0 otherwise, which could lead to confusion. In any case, we then process the response as follows: if the user does not wish to keep the collection, then it is added to the list of collections to be excluded, otherwise it is added to the list of intermediate collections. The process is repeated for the n collections to be labeled. An example of feedback for this exclusion is shown in Figure 2.6.

```
Do you want paths that go through a collection labeled "invites" ? (1 for yes, 0 for no)
1

Do you want paths that go through a collection labeled "gender" ? (1 for yes, 0 for no)
1

Do you want paths that go through a collection labeled "hasWritten" ? (1 for yes, 0 for no)
1

Do you want paths that go through a collection labeled "honorificPrefix" ? (1 for yes, 0 for no)
1

Do you want paths that go through a collection labeled "publishedIn" ? (1 for yes, 0 for no)
1
```

Figure 2.6: Example of a feedback for the exclusion of intermediate collections

- Inversions in paths

The user interaction regarding the inversion of paths works on the same principle as for the exclusion of collections. We begin by asking the user the number of inversions that he wants to label, with the aim of offering him as many, ranked in an order already explained above. The difference is that not all inversions are consistent, and some should not be proposed if they do not exist in the data graph (please refer to section 8.1.4 for more details). Thus, we proceed as following. We start by taking the collection at the top of the ranking, then we build the first possible inversion from it. We then check whether it is consistent. If so, we propose it to the user. If not, we do nothing, and we build the second possible inversion. Once again, we check whether it is consistent, and if so, we propose it. This is repeated until one of the following two conditions is met: either the user has been offered as many inversions as requested, or all possible inversions have been tested. This second scenario occurs when, of all the existing inversions, there are not enough consistencies to offer the user as many as he has requested. An example of such a feedback is shown in Figure 2.7.

```
Do you want the following sub-path to be included in the paths :
Paper (first occurrence)      --      publishedIn      --      Conference      --      publishedIn      --      Paper (second occurrence)

1

Do you want the following sub-path to be included in the paths :
Paper (first occurrence)      --      city      --      Value of {place, city}      --      city      --      Paper (second occurrence)

1

Do you want the following sub-path to be included in the paths :
Value of {place, city} (first occurrence)      --      city      --      Paper      --      city      --      Value of {place, city} (second occurrence)

1

Do you want the following sub-path to be included in the paths :
Author (first occurrence)      --      honorificPrefix      --      Value of {honorificPrefix}      --      honorificPrefix      --      Author (second occurrence)

1

Do you want the following sub-path to be included in the paths :
Author (first occurrence)      --      birthDate      --      Value of {birthDate}      --      birthDate      --      Author (second occurrence)

1
```

Figure 2.7: Example of a feedback for the selection of inversions

Creation of a rudimentary but effective and understandable interface

This section presents the interface that was created to interact with the user. Since this work is primarily a research work, it has not been discussed here to develop an advanced interface. Besides, the aim was to develop the simplest multi-platform interface possible, while remaining understandable and usable by a non-expert user.

So we made the choice to make a command line interface, directly in the terminal. Thus, it is possible to use the application on both Linux and Mac OS systems. Windows users can also use the application without problems, by installing upstream any terminal emulator of one of these two operating systems (the Bash shell for instance).

We will not discuss here to the user interaction as part of the exclusion presented in section 8.2.1. In this section, we will focus on two aspects: the header summarizing at each stage of the application the status of the choices made and of the enumeration, as well as the display of the paths once enumerated, although details on the enumeration will be given in section 8.3.

Before each new step of interaction with the user, whether to ask him questions in the context of the exclusion or to display him information or the enumerated paths, the screen of the application is erased, and a header summarizing the progress of the process is displayed. This header, once the enumeration is complete, is composed of the following information: maximum number of paths, maximum size of paths, pair of entity types chosen, number of labeled collections, number of authorised inversions, and number of paths enumerated with respect to all parameters. An example of a complete header is shown in Figure 2.8.

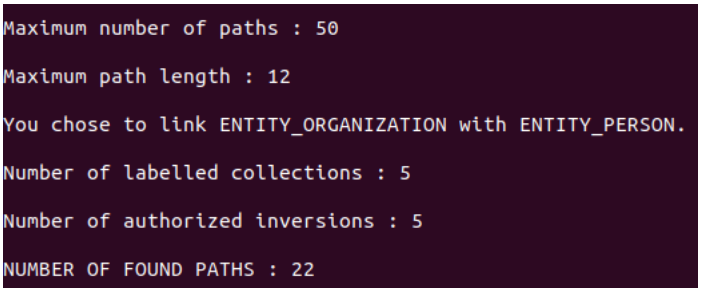


Figure 2.8: Example of a complete header, at the end of the execution

Once the enumeration is complete, the paths are displayed (with their support, and their length). They are delimited by an opening bracket ([), and by a closing bracket (]). Then we find the names of the two types of named entities that the path links, then the different collections that make up this path, with the orientation of the edges between them. We also find edges between collections that are not oriented, and that have the mention "*SHARED ENTITY*" : it corresponds to collections that are sharing (at least) one common data between each other. It also provides an example of such a common entity. An example of a path containing all these properties is shown in Figure 2.9.

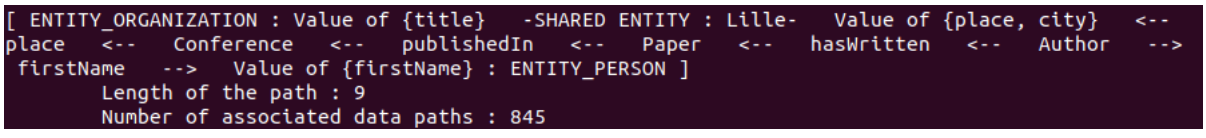


Figure 2.9: Example of the display of an enumerated path

2.2.3 Path enumeration and path supports

Now that we have access to user-made decisions, we can move on to enumerating paths. In this sub-section, we will present the algorithm of the path enumeration in the first subsection, and how we compute the supports of the enumerated path in the second one.

Path enumeration algorithm

The first observation to make is that thanks to the pair of named entity types, we can access the list of target collections, namely those associated with (at least) one of the two types of the pair. Since we want to build paths that exclusively link these collections, then it is wise to start exploring the graph for the enumeration by them. We will do this by recurrence: we will first build paths of size 1, then, until we reach some criteria that we will define later, build all paths of size n from paths of size $n-1$.

Our algorithm will therefore be broken down into two major parts: the initialization, then the recurrence.

- Initialization

The pseudo-code of the initialization part of the enumeration algorithm can be found in the Appendix A of this work. We will go into the details of this algorithm. Basically, what we want to do here is to build all the length 1 paths starting from the target collections. We first create two lists: a list of complete paths, which therefore link two collections target, and which paths respect all the criteria defined by the user, and a list of incomplete but valid paths (meeting the criteria), that will allow us to build n size paths from $n-1$ size paths.

In order to obtain these length 1 paths, we will systematically go through each of these collections. For each of them, we will see if it is possible to join others via an edge of the collection graph. If not, then we'll just move on to the next collection. Otherwise, for each edge found, a potential path of size one can be added to the list of incomplete paths, but we must ensure that it meets our criteria. There are three of them at this stage. Indeed, the new collection that we want to add mustn't be a collection excluded by the user, and it mustn't already be in the path: if inversions are allowed, they will be managed later. Finally, the size of the new path (here which will be equal to 1) must be well below or equal to the maximum size defined by the user. If these conditions are met, then the path can be added to the list of incomplete paths. However, it is possible that the path we have just created is not an incomplete path, but a complete one, in we have that the second collection is a target collection, and if the types of the two collections of the formed path constitute the defined pair. If so, then we add the path to the list of complete paths, and remove it from the list of incomplete paths.

We're adding two features to that. The first is to see if other incomplete paths in the list could be merged to get a new complete path. To do this, we look at whether there are other paths that have their second collection in common with the actual incomplete path, and if this is the case and that the two target collections fit with the pair, then we merge these two paths into a third complete path that we add to the corresponding list. Finally, we check if there are possible inversions from the second collection of the incomplete path. If this is the case, we create a new path from the combination of incomplete path and inversion, and add it to the list of incomplete paths.

By doing so, we have all paths of length 1 that are valid : the complete ones in a dedicated list, and the incomplete ones in another list. We will now see how to retrieve all the other existing complete paths from this list.

- Recurrence

We now consider the case where we have paths of length $n-1$ in order to build paths of length n. So, starting from our list of incomplete paths of length 1, we will generate all valid paths of length 2. Those that are complete will be put in the list of complete paths, and those that are not will go in a new list of incomplete paths of length 2, which will overwrite all incomplete paths of size 1 now processed, allowing then to get valid paths of length 3, and so on. We will repeat this operation several times, until we meet a

stop condition. This one should stop us from looking for new paths in two cases: either we have reached the maximum path length, or the maximum number of paths, both defined by the user. Let's now describe how to build all valid paths of length n from incomplete valid paths of length $n-1$.

We first define a list of new incomplete paths in which we will add all new valid incomplete paths of n. Then, we iterate over the list of incomplete paths (of length $n-1$). For each of these paths, we take the last added collection, and see if there are any edges leading to other collections from it. If that's not the case, then nothing is done. Otherwise, for each of these edges, we have to check the validity conditions. These are the same as for the initialization part of the algorithm, and there are 3: we check that the outgoing collection of the edge is not already in the path, that it is not forbidden, and that if we add this new edge to the path, we do not exceed the maximum length. If these conditions are met, then we add this path to the new list of (valid) incomplete paths (those of length n therefore).

Still following the same schema as for initialization, now we check if the newly obtained valid path is complete. To do this, we see if it links the two types of entities of the pair, if it is not already contained in the list of valid complete paths, and if its new size does not exceed the maximum size. If this is the case, then we add the path to the list of complete paths, and remove it from the list of incomplete paths of n.

Again, we check if it is possible to merge two incomplete paths sharing the same last collection (and not sharing other collections), in which case we add the path in the list of complete paths. We also check if an inversion is possible, by seeing if inversions are associated with the last collection added to the incomplete path. If this is the case, then merge the path with the inversion and add it to the list of incomplete paths of n, for each possible inversion. However, merging a length $n-1$ path with an inversion that is a length 5 path gives a length $n-1+5 = n+4$ ($\neq n$) path. Thus, in reality, our list of incomplete paths of length n will be a list of incomplete paths of at least length n, built on the basis of paths already explored of at least length $n-1$. In practice, this poses no problem, as the construction is done from close to close, increasing the length of each path by 1, regardless of its initial length.

The pseudo-code of the recurrent part of the enumeration algorithm is available in Appendex B.

Computation of paths support

Since the list is given in the previous subsection, we now have access to the list of paths corresponding to the user-defined parameters. It is now interesting to determine concretely, how many (different) paths of the data graph exist for each path of the listed collection graph. This will, among other things, determine which of the listed paths are the rarest in the collection graph.

To do this, we will need to query the data graph. Until now, we have had very little need to go to the data level, and rightly so: we have chosen to go through the collection graph instead of the data graph to gain efficiency, So there was no question of going back through the data graph to do so. The data graph was used only twice: to calculate *edge-specificities* for the exclusion of intermediate collections, and for checking the consistency of inversions (section 2.2.1). However, these calls were limited and inexpensive.

However, in order to calculate the support of a path, we have no choice but to use this data graph. Technically, given its large size, the graph is stored on disk, not in memory. As a result, it is more time-consuming to access (more details in section 2.2.4). Thus, since the data graph is stored in a database, it is necessary to query this database, so that it returns to us the path support of the collection graph. To do this, a module has already been developed in Abstra, in order to directly query the database during execution, using the appropriate class in the code. This greatly facilitates our work, since we can use what we have already calculated that is in memory (the paths and the collection graph) in order to build a query that we will then send to the disk (in particular, to PostgreSQL, which is the database management system used by Abstra and ConnectionLens). Our work is therefore limited to creating a query, opening a connection to the database (DB), sending it the query, and waiting for its response (the support) which we recover in memory and which we can use. The construction of the query is based on several information: the paths

themselves, especially the collections that they contain, their length. We also need the collection graph, to retrieve the ID of the collections, and the connections between them; We finally need the names of the tables in the dataset, and their corresponding standard name defined by Abstra. A detailed explanatory diagram is shown in Figure 2.10.

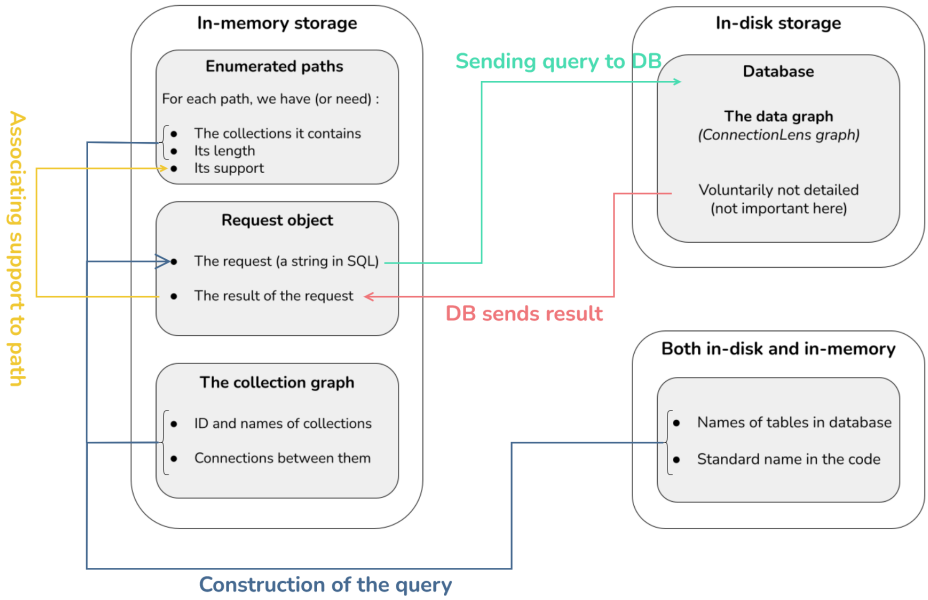


Figure 2.10: Explanatory diagram of path support computation

2.2.4 Experimental evaluations and performance analysis

Now that we have completed the work and that the application is functional and ready for users, it is necessary to analyze its performance, through different experiments.

In the first sub-subsection, we will present the three datasets used to conduct these experiments. In the second one, we will present the results obtained and the trend curves that we were able to compute. Finally, in the last, we will critically judge these results and compare them to what we should expect in theory.

Presentation of the three datasets

Before describing the datasets themselves, it is important to justify their choice. Indeed, as we have seen several times, the ConnectionLens and Abstra environment aims to integrate large volumes of data. Thus, even if we have everything in place to ensure that our work remains functional despite a large volume of data, we must ensure that the scaling is effective, so that by increasing (gradually) the amount of data processed, The processing time remains acceptable (called tractability). As a result, we selected three data sets, which are growing in size.

The first dataset is 21KB, and describes data related to scientific research, in particular links between papers, authors and conferences. The second dataset is about a thousand times larger than the first, and is 18MB. It describes a data set made available by NASA on missions, vehicles, astronauts, equipment and many others. Finally, the last dataset is about ten times larger than the previous one, and makes 168MB. It describes data from foodista.com, which are mainly recipes, which are associated with ingredients, utensils, times, and many more.

File sizes were not randomly selected either. The last file is large enough to correspond to more than 1.2 million data in the data plot, which can be characterized as a large volume of data, and would confirm

the scaling. However, it would have been difficult to consider more data sets in order to better scale up the intermediate volumes. Indeed, at each volume level, it is necessary to find a dataset that corresponds more or less to the desired volume, that it contains data with types of entities named real (necessary to find real life datasets), and that they are complex enough for the collection graph to be rich.

That's why we took a really small data set and then moved very quickly to a fairly comprehensive data set. This makes it possible to quickly pass stages that in any case would make the ladder transition impossible. Finally, the last pass makes it possible to really check the scale change and the evolution of time with the number of data.

Empirical evaluation and trend curves

In order to make the evaluation, we will calculate the time taken to perform certain steps of the algorithm. In particular, we are interested in the processing time for the exclusion of collections and the processing time for the authorization of inversions, without taking into account the time it takes the user to respond to us. We are also interested in the time to do the enumeration of the paths, then finally time to calculate the support of all the enumerated paths. All the times are expressed in milliseconds (ms).

In addition, we will evaluate two scale-up experiments. That of the data volume, therefore with fixed parameters, and that of the number of paths, with fixed parameters except for the number of paths.

- Scale-up experiment on the data volume

This evaluation consists in studying the evolution, for the four treatments mentioned above, of time according to the size of the datasets (and more precisely, the number of nodes present in the data graph). We set the same parameters for the 3 experiments : we fix the maximum number of paths to 50, we don't put a limitation of the maximum path length (we set a veary high number), we decide to label 5 collections (but not to exclude some of them), and to label 5 inversions. With these parameters, we ensured ourselves that we actually do have 50 paths (i.e., the maximum number of paths we set).

Let's start with the time taken to calculate the potential collections to exclude. The time measurement results as well as the trend curve that presents time as a function of the number of nodes in the data graph is presented in Figure2.11. It turns out that we observe a quasi-linearity between time and the number of nodes. The trend curve equation is $y = 0.003x + 346.7$.

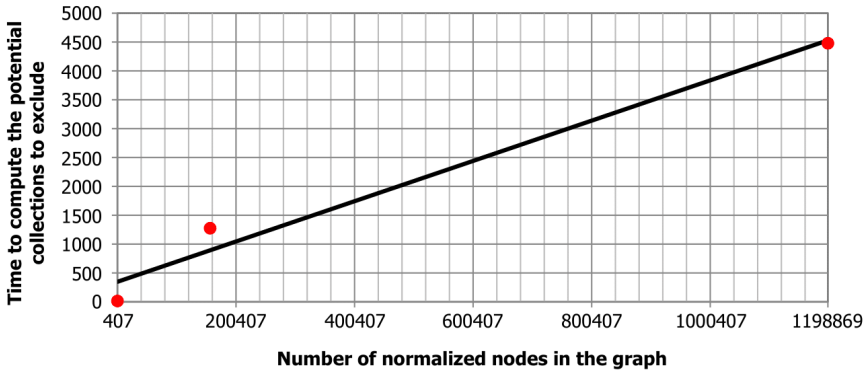


Figure 2.11: Trend curve of the time to compute candidate collections to exclusion in function of the number of nodes in the data graph

We then have the time taken to calculate the inversions proposed to the user. The time measurement results as well as the trend curve that presents time as a function of the number of nodes in the data graph is presented in Figure 2.12. In this case, we see that, given the Figure 2.12 y axis's scale, we don't really have a linear relation between time and the number of nodes. The trend curve is $y = 0001x - 432.3$.

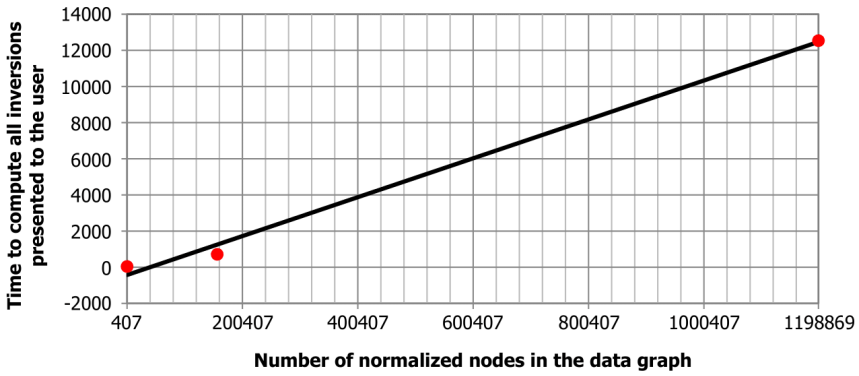


Figure 2.12: Trend curve of the time to compute the inversions proposed to the user in function of the number of nodes in the data graph

Now, regarding the time taken to calculate the support of the 50 listed paths. The time measurement results as well as the trend curve that presents time as a function of the number of nodes in the data graph is presented in Figure 2.13. In this case, we have a great linear relation between the time taken to compute the supports and the number of nodes in the data graph. The trend curve is $y = 0.207x + 5959.7$.

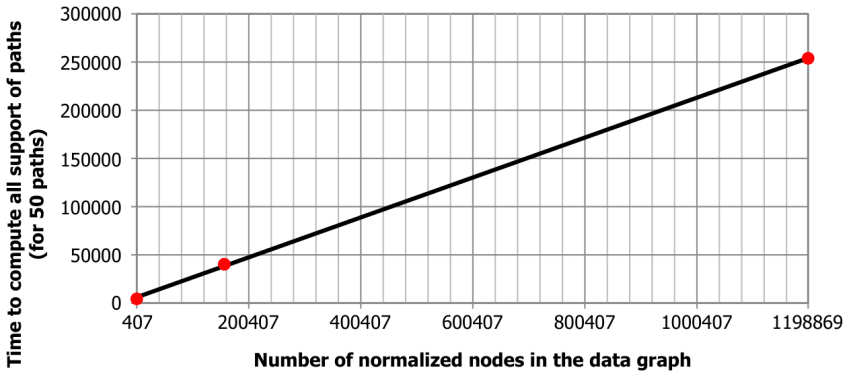


Figure 2.13: Trend curve of the time to compute the (50) paths supports in function of the number of nodes in the data graph

Finally, it remains to look at the time taken to list the 50 paths. It turns out that the enumeration takes the same time for the three datasets, with a few milliseconds. Thus it is not relevant to define a trend curve, because it seems that the time taken to list the paths is independent with the number of nodes (we will come back to this during the analysis).

- Scale-up experiment on the number of paths

Even tough we don't have a particular relation between the time taken to compute the supports and the number of nodes, it is interesting to look at what happens when, for the same dataset, we compute the same times but with a different number of enumerated paths. We therefore use the same parameters as for previous experiment, but this time with an evolving number of paths.

We decided to use the NASA dataset to do these experiments, since it is an intermediate dataset in terms of size (even though we don't consider the size anymore here). We conducted three computations : a first one with 100 paths to be enumerated, the second one with 200, and the third one with 300. We will only compute two times : the time taken to compute the support of the paths, and the time taken to make the enumeration. In fact, since the computation of the collections to be excluded and of the inversions only depends on the dataset itself and not on the enumeration, then we would end up having the exact same time

for each (or practically the same).

Let's start with the time taken to make the path enumeration. The time measurement results as well as the trend curve that presents time as a function of the number of nodes in the data graph is presented in Figure 2.14. It turns out that we find a quite good linear relation between the time taken to make the path enumeration and the number of paths. The trend curve is $y = 1.29x - 22$. We can also, for each number of path, the mean time to enumerate one path. We end up having 1.120 ms/path for 100 paths, 1.130 ms/mpath for 200 path and 1.233 ms/path for 300 path, which also supports the linear trend of the relationship.

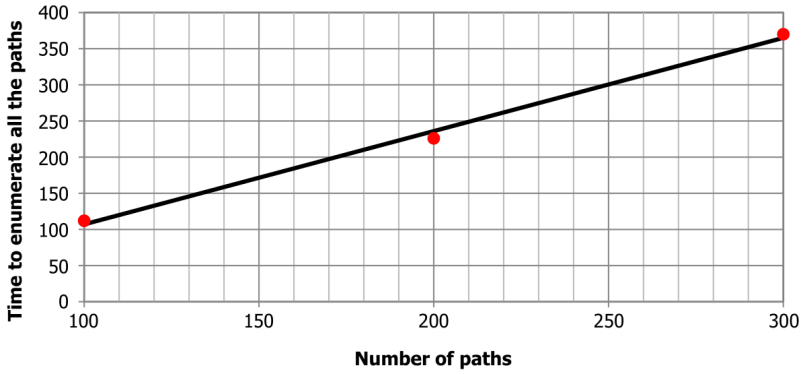


Figure 2.14: Trend curve of the time to compute enumeration of all paths in function of the number of paths

Then, we have the time taken to compute the support of all paths. The time measurement results as well as the trend curve that presents time as a function of the number of nodes in the data graph is presented in Figure 2.15. It turns out that we also find a good linear relation between the time taken to compute all path supports and the number of paths. The trend curve is $y = 1228.5x - 30221.7$. We can also, for each number of path, the mean time to enumerate one path. We end up having 1.480 s/support for 100 paths, 1.431 s/support for 200 path and 1.312 s/support for 300 path, which also supports the linear trend of the relationship.

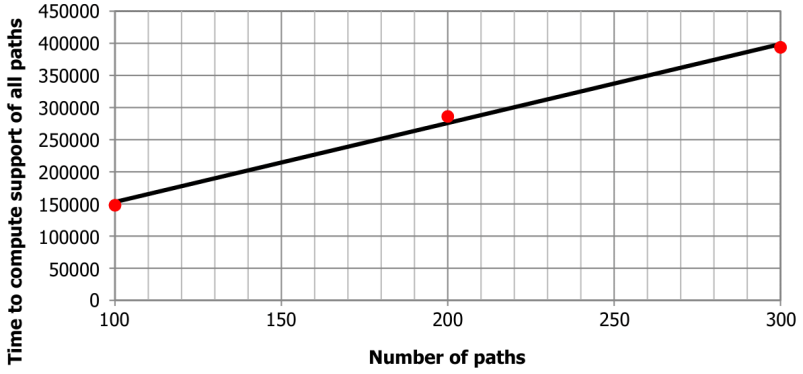


Figure 2.15: Trend curve of the time to compute all paths supports in function of the number of paths

Analysis of the results and comparison with the theory

The last step is to analyze the results obtained. We will go through the six results obtained through the two experiments.

- Scale-up experiment on the data volume

Concerning the first experiment, let us first focus on the results concerning the exclusion of collections and the calculation of inversions. For collections, we've seen a near-linearity with the increase in the number of data. That's good results. In fact, theoretically, the best temporal complexity is linear, because the increase in the number of nodes in the data graph necessarily increases the number of edges in the same graph. Thus, the number of edges evolving linearly, the number of edges to be recovered from the data plot will be linear, and the number of *edge-specificities* will also be linear in order to find their minimum. Thus, we cannot hope for better in terms of order of magnitude (on the other hand, the coefficients associated with the linearity could differ). Secondly, regarding inversions, we could not observe a linear relationship with the number of data. However, for the same reasons as for the collections, we should observe one, since we rely on the data plot to verify the validity of an inversion. There is a justification for that. Indeed, in the context of inversions, we only present them for labeling if they are consistent. This has the consequence that it may happen that we have to prepare more inversions, and therefore check the validity of more inversions, in order to still present to the user the number of inversions he has requested. We can take a simple example: the user asks to label four inversions. The first two are consistent and therefore presented, but it turns out that the next two are not. We must then continue to search until we find four consistent ones: let's say the following two are. Thus, we had to consider six inversions to present only four. Therefore, even in theory, the inversions do not evolve (in the middle case) in a linear temporal complexity, since it also depends on the articulation of the dataset.

Now about the computation of paths supports, results showed a quite good linear relation with the number of data. As for the collections, this is the best (in terms of order of magnitude) that we can have. Again, we need to access the data graph to check how many paths correspond to a given collection graph path. Therefore, the more the data in the data graph, the more path will be found (or at least, the more part of paths will be checked), and therefore the more time will be taken, and this in a linearly way. Finally, regarding the time taken to enumerate all paths, we observed an independence with the number of data in the data graph. In fact, that is completely normal: the paths are enumerated on the collection graph, and this collection graph is provided by Abstra, and is therefore stored in-memory (as showed in Figure 2.10). Thus, we do not need to dive into the data level, and the complexity of a collection graph does not depend on the number of data, but more on their articulation and on how heterogeneous they are. But in every case, their complexity remains reasonable, and that explains why we find the same times (within milliseconds), regardless of the size of the datasets.

- Scale-up experiment on the number of paths

We still have two set of results to analyze: the ones that are related to the number of paths. We already explained why we don't focus on the times for collections exclusion and inversions, thus we will skip them. For both computations of paths supports and enumeration, we empirically identify a linear relation between these and the number of paths.

In fact, it appears that this is the best that we could expect theoretically, in terms of order of magnitude. For the enumeration part, we could have ended up having worse results, since increasing linearly the number of paths to enumerate does not necessarily increase linearly the number of incomplete paths to explore. But since we get rid of incomplete paths that cannot be extended as we go along, and merge the paths that can be, we manage to reach (at least empirically) a linearity that corresponds to the best case in theory. Finally, regarding the paths supports, we have already seen that, for a fixed number of paths, that we have a linear relation between the time taken to compute them and the size of the dataset. As a consequence, changing the number of paths will not change this complexity, since the dataset will remain the same and therefore the size of the dataset will remain constant. Therefore, there is theoretically a linear relation between the number of paths and the time taken to compute their supports. That leads to the fact that we can characterize the time-complexity function for paths supports completion : $T_{supports}(numberOfPaths, sizeOfDataset) = \mathcal{O}(numberOfPaths * sizeOfDataset)$.

Chapter 3

Analysis of achievements and development of the career plan

Beyond the work that has been done, it is important to focus on the personal contributions of this internship.

Specifically, we describe in the first section of this chapter the skills acquired during this internship, both technical and human. Then in the second section, we describe the participation of this experience in the construction of the professional project, especially for the final internship that will take place next year.

3.1 Achievements and acquired skills

n this section, we will detail the skills developed or reinforced during this experience. In the first sub-section, we will focus on technical and scientific knowledge acquired, and in the second sub-section, on human and relational skills acquired.

3.1.1 Technical and scientific knowledge

This internship of high scientific intensity necessarily allowed the acquisition and strengthening of technical and necessarily scientific skills. In this subsection, we first develop the skills acquired in the research dimension of the internship, in the first sub-subsection. In the second, we describe the theoretical knowledge I have acquired, and in the third, the technical skills in computer development.

Research skills

First, this internship was primarily a research internship, and as such, I was able to develop many knowledge and skills in this field. This allowed me to learn how to read and analyze scientific papers properly. Indeed, these papers have a particular vocabulary and synthesis, and I was able to take the time to adapt. The study of these papers also taught me to make a study and a synthesis of the state of the art of a domain. Working in this environment has also allowed me to develop a systematic approach to solving scientific problems, to organize a reflection and to draw rational conclusions. I have also, thanks to my tutor and my colleagues, been able to follow the progress of several research projects, and in particular to learn more about the functioning of a PhD, scientific conferences and the organization of a research team.

Theoretical skills

Then, the study of these papers and the research work in the field of graphs and processing large volumes of data allowed me to acquire theoretical knowledge in these fields. Whether in graph theory, evaluation functions, man-machine interaction, natural language processing or even database management, I have been able to solidify knowledge that will serve me later in my professional life, but also in the rest of my studies.

Technical skills

Finally, I also learned a lot in computer development. First in knowledge of the Java language, where I learned to work on the basis of a large-scale project (Abstra), which required using Gitlab (a Git platform of Inria, Git being a version management system) in order to follow the progress of my work and manage its evolution with the evolution of Abstra, and thus learn to use Gitlab. I have also developed my IT-language skills in itself, including learning to choose the right data structures based on our needs. Finally, I also had the opportunity to learn a number of good practices in programming, and was able to gain in efficiency and speed in needs analysis and problem solving.

3.1.2 Human and relational skills

This internship was also an opportunity to develop a number of human and relational skills. In this subsection, we will describe the relational and human skills acquired within the collective in the first sub-section, then the personal ones in the second.

Collective skills

First, the work environment itself has brought me great learning in communication and interculturality. Indeed, the team was composed of less than 20 people, but was composed of at least 7 different nationalities. This means that it is necessary to communicate in English at all times, which is the working language. I already had good skills in this language, but less in a scientific context, where a specific vocabulary is required, and where one must expose ideas and scientific reasoning sometimes abstract and difficult to illustrate simply. I was also confronted with the difficulty of accents and expressions used by different nationalities. But the frequent team meetings, the communication with colleagues on various problems encountered, the reading of the literature and the writing of this report have allowed me to develop this skill and to better and more successfully communicate. Thus, I also had the opportunity to work as a team and learn to present my work.

Personal skills

Beyond the skills related to the collective, I also developed personal skills. Indeed, this internship required a great deal of autonomy in the work, both in order to be able to advance alone, but also in decision-making when it was necessary to favour one idea over another, or to discard a technical solution in favour of another. This internship also allowed me to develop my self-learning, especially in the context of the study of the state of art, where I had to, by myself, understand new abstract concepts, or to apprehend technologies still unknown. In my view, this skill is particularly essential to the work of engineers and even researchers, and will serve me well in the future. The organisation was also a key point of this internship. Whether it's to ensure a rigorous follow-up of my progress in terms of code via Gitlab, but also the follow-up of the decisions taken, the reasons that led to dismiss certain hypotheses, and the reasoning that led to new ideas. This resulted in the use at least weekly of a document shared with my tutor, where I wrote down all this information. Finally, I also had the opportunity to develop my critical mind, both on the work of others (especially in the study of the state of art), but also on my own work. Taking a step back often makes it possible to see a problem from several angles and sometimes to be aware of errors or incompleteness.

3.2 Development of the career plan

This experience was also an opportunity to reflect on my further studies and on my professional project. Indeed, I already had the idea, before this internship, to pursue at least part of my career in research.

This internship has fully supported this desire. I had the opportunity to see what the research work represented in concrete terms, and to experience it in concrete terms, having the opportunity to do so in a cutting-edge environment. Beyond the problems of the internship, I also discuss with my tutor and a PhD

student the reality of the research profession, and how to go about guiding my profile and my career in this direction. That is why, as soon as I finish my engineering studies, I plan to do a PhD. In addition to this, I would like to do so within Inria, whose working environment and team-project operation is very pleasant, because often associated with research work with concrete applications in society, whether for industry or even for public institutions. In particular, I loved working at the Inria Saclay Île-de-France centre, which works on exciting topics, with passionate people and with great means.

In addition, I was admitted for my last year of engineering studies at Télécom Paris to complete my specialization. This year includes a master's research and innovation project, which will be another opportunity to be confronted with research, or research and development in industry. The topics covered in my internship also interested me a lot, I decided to take courses to deepen the notions that I could meet but on which I could not work (knowledge bases, logical systems, architecture for big data and natural language processing among others). In addition to these courses, I have also chosen to take courses in the fields of statistical learning, machine learning, deep learning and optimization, which are subjects that are also of great interest to me, and that I will have the opportunity to explore further during this year.

Thus, my goal is to do my internship again in a research team, in the same field as the CEDAR team or in another team more oriented in the fields mentioned above. I would like this internship to be an opportunity to prove myself and thus have the opportunity to do a PhD. I ideally want to do this in one of the teams associated with the schools of the Institut Polytechnique de Paris, where I did my M1 internship, as well as my future year of M2, which is a state-of-the-art institute at the centre of major research issues in artificial intelligence.

Conclusion and perspectives

So, after four months in the CEDAR team at Inria's Saclay Île-de-France centre, I was able to develop a tool for exploring graphs generated by ConnectionLens using a path-based approach. In particular, I had the opportunity to create an interactive application, which will be used to facilitate work in investigative journalism. I had to mobilize both theoretical and practical knowledge in the fields of data processing and graph exploration in order to carry out this work.

The creation of the application as well as the development of new ideas in the field of the exploration of large graphs of heterogeneous data is a definite contribution. First, because the application will make it possible to facilitate the work of investigative journalists, and thus by extension access to information for the population. Second, this work contributes to the advancement of the team's overall research work, and more specifically, it will serve as the basis for the doctoral work of a student starting at the beginning of the 2022 academic year.

More personally, this internship allowed me to develop strong scientific skills in the field of graph theory, artificial intelligence, database systems and IT development and its best practices. Moving into the research world has also brought me scientific analytical skills, but also strong human and relational qualities, such as teamwork in an intercultural context, autonomy, organization and scientific communication in English.

I chose this internship to discover the field of artificial intelligence research, and this experience reinforced my desire to continue my studies in this direction. That's why my goal is to do my M2 internship with an artificial intelligence research team, so that I can then pursue doctoral studies once I graduate as an engineer.

It is also important to analyse the limitations and potential evolutions of this work. Indeed, although we have done a performance analysis that shows a good result, the work done could become widespread. In particular, ConnectionLens graphs are intended to integrate different types of data files, and as part of this work, we have relied on properties owned by the standard graph model describing web resources and their metadata, The RDF (Resource Description Framework). However, some types of files do not have exactly the same properties (XML files for Extensible Markup Language for example), and as such cannot be processed as a result of the work done. In reality, it is mostly that changes should be made (for example, using other metrics to classify collections or considering other types of inversions). But on the principle and the ideas developed, we made sure to remain sufficiently generalist. Thus, a good improvement of this work would be to generalize its operation to a greater number of types of data files.

Finally, a last good improvement would be to create an Graphical User Interface (GUI) instead of the actual apllications's interface, which would be less brief and more pleasant for the user. Especially, it would be interesting to create a web interface, through which it would be much easier to integrate components and a display that will greatly improve the user experience in this direction.

Bibliography

[ABC+21] Angelos Christos Anadiotis, Oana Balalau, Catarina Conceicao, Helena Galhardas, Mhd Yamen Haddad, Ioana Manolescu, Tayeb Merabti, and Jingmao You. Graph integration of structured, semistructured and unstructured data for data journalism. *Information Systems*, page 42, July 2021.

[BCL15] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. Learning Path Queries on Graph Databases. In *18th International Conference on Extending Database Technology (EDBT)*, Bruxelles, Belgium, March 2015.

[DAB16] Gonzalo Diaz, Marcelo Arenas, and Michael Benedikt. SPARQLByE: Querying RDF Data by Example. *Proc. VLDB Endow.*, 9(13):1533–1536, sep 2016.

[Des20] Marie Destandau. *Path-Based Interactive Visual Exploration of Knowledge Graphs.* Theses, Université Paris-Saclay, December 2020.

[LF06] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 631–636, New York, NY, USA, 2006. Association for Computing Machinery.

[LMPV20] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. Graph-query suggestions for knowledge graph exploration. In *Proceedings of The Web Conference 2020*, WWW '20, page 2549–2555, New York, NY, USA, 2020. Association for Computing Machinery.

[LPHM20] Matteo Lissandrini, Torben Bach Pedersen, Katja Hose, and Davide Mottin. Knowledge graph exploration: Where are we and where are we going? *SIGWEB Newsl.*, (Summer 2020), jul 2020.

[Mor21] Lucas Maia Morais. Query result scores and quality in ConnectionLens. M1 Internship Report, Ecole Polytechnique, 2021.

[YLW+22] Yi Yang, Meng Li, Jian Wang, Weixing Huang, and Yun Wang. Entity recommendation with negative feedback memory networks for topic-oriented knowledge graph exploration. *IEEE Transactions on Reliability*, 71(2):788–802, 2022.

[Zha21] Xin Zhang. Learning to rank answer trees in ConnectionLens graphs. M2 Internship Report, Télécom Paris, 2021.

Appendix A

Initialization part of the enumeration algorithm

Algorithm 1 Enumeration algorithm : Initialization

Variables
    maxPathLength: maximum length of enumerated paths.
    maxNumberOfPaths: maximum length of enumerated paths.
    targetCollectionsToLink : The list of all target collections to be linked.
    forbiddenCollections : list of excluded collections.
allPathsBetweenTargetCollections: the list of all (valid) and complete paths (enumerated paths).
    entityTypesToLink : The pair of entity types that the paths must associate.
    incompletePaths: 2D array of lists of paths, the (valid) incomplete paths.
    authorizedInversions : All the authorized inversions.
end Variables
for each collection c in *targetCollectionsToLink* **do**
    for each outgoing edge oe of collection c **do**
        Create a new path p consisting of the only edge oe
        if *incompletePaths[p.first][p.last]* **does not** contain p **and** p.last **is not in** *forbiddenCollections* **then**
            Add p to *incompletePaths[p.first][p.last]*
        end if
        if p is linking the two entity types in *entityTypesToLink* **then**
            Remove p from *incompletePaths[p.first][p.last]*
            if *allPathsBetweenCollections* **does not** contain p **and** *allPathsBetweenCollections*.size $<$ *maxNumberOfPaths* **then**
                Add p to *allPathsBetweenTargetCollections*
            end if
        end if
        if p.last is in keySet of *authorizedInversions* **then**
            for each path p' in *authorizedInversions*.get(p.last) **do**
                if p'.size $+ 1 <$ *maxPathLength* **then**
                    Create a new path $p_{withInv} = p + p'$
                    Add $p_{withInv}$ to *incompletePaths[$p_{withInv}$.first][$p_{withInv}$.last]*
                end if
            end for
        end if
    end for
end for

Appendix B

Recurrence part of the enumeration algorithm

Algorithm 2 Enumeration algorithm : Recurrence

Variables
All the ones from Algorithm 1
newIncompletePaths : 2D array of list of paths, containing all incomplete paths of length greater or equal that n based on all incomplete paths of length greater or equal than $n-1$.
end Variables
$currentLength \leftarrow 1$
while $currentLength + 1 < maxPathLength$ **and** $allPathsBetweenCollections$.size $< maxNumberOfPaths$ **do**
  $newIncompletePaths \leftarrow$ empty 2D array of lists of paths
  for each path list lop in $incompletePaths$ **do**
    for each path p in lop **do**
      $candidateEdges \leftarrow$ List of all outgoing edges of p.last
      for each edge e in $candidateEdges$ **do**
        if the candidate collection to be added (associated to e) **is neither** already in p, **nor** in $forbiddenCollections$ **and** p.size $+ 1 \leq maxPathLength$ **then**
          Create a new path $p' = p + e$
          Add p' to $incompletePaths[p'.first][p'.last]$
          if p' is linking the two entity types in $entityTypesToLink$ **then**
            Remove p' from $incompletePaths[p'.first][p'.last]$
            if $AllPathsBetweenTargetCollections$ **does not** contain p' **and** $allPathsBetweenCollections$.size $< maxNumberOfPaths$ **then**
              Add p' to $AllPathsBetweenTargetCollections$
            end if
          else
            if sizeOf(p') $* 2 \leq maxPathLength$ **then**
              for each path p'' in $incompletePaths$ such that p''.last=p'.last **do**
                if the merge of p' and p'' is possible **then**
                  Create $\bar{p''}$, the inverted path of p''
                  Complete p' with $\bar{p''}(p' + \bar{p''})$
                  if p' **is not already in** $allPathsBetweenTargetCollections$ **then**
                    Add p' to $allPathsBetweenTargetCollections$
                  end if
                end if
              end for
            end if
            if p'.last is in keySet of $authorizedInversions$ **then**
              for each path p'' in $authorizedInversions$.get(p'.last) **do**
                if p''.size $+ p'$.size $< maxPathLength$ **then**
                  Create a new path $p'_{withInv} = p' + p''$
                  Add $p'_{withInv}$ to $incompletePaths[p'_{withInv}.first][p'_{withInv}.last]$
                end if
              end for
            end if
          end if
        end if
      end for
    end for
  end for
  $incompletePaths \leftarrow newIncompletePaths$
  $currentLength \leftarrow currentLength + 1$
end while