COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique et Architectures numériques*

Par
H. Ambre AYATS

Construction de graphes de connaissances à partir de textes avec une intelligence artificielle explicable et centrée-utilisateur·ice

Thèse présentée et soutenue à Rennes, le 21 décembre 2023
Unité de recherche : IRISA, CNRS, UMR 6074

Rapporteurs avant soutenance :

Sebastian RUDOLPH	Full Professor, Technische Universität Dresden
Fathia SAÏS	Professeure des Universités, Université Paris Saclay

Composition du Jury :

Président :	François GOASDOUÉ	Professeur des Universités, Université de Rennes
Rapporteurs :	Sebastian RUDOLPH	Full Professor, Technische Universität Dresden
	Fatiha SAÏS	Professeure des Universités, Université Paris Saclay
Examinateurs :	Marianne HUCHARD	Professeure des Universités, Université de Montpellier
	François GOASDOUÉ	Professeur des Universités, Université de Rennes
Dir. de thèse :	Sébastien FERRÉ	Professeur des Universités, Université de Rennes
Co-encadrante :	Peggy CELLIER	Maîtresse de Conférence, Université de Rennes

Acknowledgement

Since I started working on the subject of my thesis, late 2019, a lot happened in my life, and many people have been here to support me. It is close to my heart to thank all of them. Without them, this thesis would not exist, and I would not be who I am today.

First, I wish to thank Peggy and Sébastien for those four years of collaboration. I guess I have not been the easiest student to manage, but both of them supported me all along. Despite my depression, my anxiety disorders, my attention disorder, they never gave up on me. When I was not able to work, they respected that, and helped me to get back in track when I was feeling better, without judgement or excessive pressure.

I also thank Luis Galárraga Del Prado and Thierry Charnois, as members of my *comité de suivi individuel*, for having been friendly and of good advice.

I also wish to thank my thesis siblings, Francesco and Julie. First, Francesco, for having been a friend and mentor when I arrived in the lab. He has been simultaneously a technical and social support for my first steps as a researcher, a close friend, my cat's father-in-law, and a friendly professional contact during the lockdowns. And Julie, for being such a great friend and colleague, and for having been such a great ally at work when I started my gender transition.

More generally, I would like to thank all the members and ex-members of SemLIS and LACODAM. SemLIS has been a great place where to start my thesis, with a great scientific expertise and nice people. And LACODAM is such a friendly environment, with great scientists, a lot of good snacks and great board games pauses, and – more important – where my transness has been immediately accepted when I came out, without questioning it and without misplaced curiosity. Special shoutout to Isseïnie, Maiwenn, Gregory, Sarah and Yasmine, that are good friends in addition to being great colleagues.

I also thank Pierre, that was my intern for a few months in 2022. It was a pleasure to work with him, and it has been a excellent first experience as a supervisor.

In July 2023, Victor and I organized, with the help of several other colleagues, a conference on the ethics of scientific research. This has been a great, rewarding experience, and I am quite proud of the result. So thank you, Victor, for leading this initiative and for having brought me into this project. I would work again with you with great pleasure.

To finish with the professional environment, I thank all the queer folks from *Queerisa*, and all the people from *CGT Ferc-Sup Université de Rennes* and the *collectif doctorant·e·s CGT*. I have not been the most reliable militant because of all that happened in my life, but I am happy to have been useful.

In a more personal aspect, I would like to thank my family. My parents, of course, for always having been a great support. They always have been here when I needed them, and this is a lot, knowing what I have been through. I also wish to thank Beat, for hosting me all this time during the Covid-19 lockdowns and my depression following it, as well as Franz, for her moral and material support. I thank as well my two sisters, Margot and Camille, for being two great, supporting persons. And I thank Ori, of course.

I would like to thank all my friends that supported me during my depression, and helped me to get back on my feet afterward. This includes, amongst others, Arthur, Salammbô, John, Dao, Tori, Mathieu, and all the people around them.

I would like to thank Sasha and Lynn. Both of them have been a huge support when I realized my transness, came out and started my transition, and I do not know how I would have lived through it without both of them. I wish all the best to both of you.

I also would like to thank two online communities, for being a source of friendship, laugh and great discussions: *Musick Has The Right To Children* and *Ostinautoscope*.

I would like to thank my partners, Nath and Arthur. Even if our relationships are quite young, you bring me love and stability in a moment where I need this. I am so proud of being part of your life, thank you so much. I also wish to thank Enka, for being a good friend and such an inspiring scientist, and Jaime, for being such a great person. And thanks to all the Dareau Bog, you make me happy.

I would like to thank all the members of my jury, hoping you find my work interesting.

Finally, I would like to thank you, who read this thesis, for being interested in my work.

H. Ambre Ayats

Résumé en français

L'intelligence artificielle est généralement définie comme la tentative de reproduire artificiellement un équivalent de la cognition humaine. Cette idée peut être retrouvée dans la célèbre question posée par Allan Turing dans son article fondateur [Tur50] : *« Les machines peuvent-elles penser ? »*. Cette définition, vague et théorique, masque en réalité un terme générique. En pratique, l'intelligence artificielle regroupe un ensemble de techniques, copiant plus ou moins directement la cognition humaine, qui visent à automatiser des tâches habituellement considérées comme intellectuelles – c'est-à-dire faisant apparemment usage de processus cognitifs de haut niveau plutôt que de processus mécaniques – et qui étaient donc considérées comme difficilement automatisables avant la naissance de l'informatique. On peut souligner que la frontière entre les tâches intellectuelles et les tâches manuelles est floue, car la plupart des tâches traditionnellement réalisées par l'humain font appel à des processus cognitifs de haut niveau ; par conséquent, la différence entre l'automatisation mécanique et l'intelligence artificielle est en grande partie arbitraire. Cependant, cela souligne le fait que l'IA, plus qu'un nouveau paradigme dans la production humaine, est globalement une extension naturelle, provoquée par l'invention des technologies informatiques, du processus global d'automatisation de la production – dont les origines ne peuvent pas être correctement retracées, mais qui s'est accéléré et qui est devenu prédominant depuis la révolution industrielle. Cet aspect de l'IA est devenu particulièrement clair au cours de la dernière décennie, avec l'explosion de l'apprentissage profond qui, malgré l'inspiration initiale des réseaux neuronaux en tant que copie des neurones biologiques, n'a pas pour objectif principal de reproduire ou d'imiter la cognition humaine, mais de résoudre des tâches en apprenant des jeux de données massifs d'entrée-sortie [1]. Cependant, cette distanciation avec la cognition naturelle pose un problème majeur, comme montré par Rudin [Rud19] et diverses controverses [2] : comment peut-on faire confiance à un résultat si les étapes intermédiaires pour obtenir ce résultat ne peuvent pas être comprises et si l'utilisateur·ice n'a aucun contrôle sur le résultat produit ? La

1. C'est pourquoi le terme *apprentissage machine* est parfois préféré au terme *intelligence artificielle* pour décrire ce type de technologie.

2. La plus connue étant sans doute le cas de *COMPAS*, un modèle utilisé dans certains tribunaux aux États-Unis pour prédire les risques de récidive, et qui s'est avéré avoir un biais raciste.

question de l'explicabilité est devenue un domaine prédominant de l'IA ces dernières années [BH21] alors que, malgré le développement de travaux sur les systèmes interactifs, la question du contrôle humain et de l'IA centrée sur l'humain est encore peu étudiée.

Le point de vue le plus classique sur la question du rapport entre l'automatisation informatique et le contrôle humain est, tel qu'il a été formalisé entre autres par Parasuraman *et al.* [PSW00], le modèle de compromis entre ces deux aspects. Dans cette conception, l'automatisation et le contrôle humain sont antagonistes : un contrôle humain élevé implique une faible automatisation, et une automatisation élevée implique un faible contrôle humain. Par exemple, Parasuraman propose une échelle pour évaluer le niveau d'automatisation et de contrôle humain d'un système, en le classant entre « *1. L'ordinateur n'offre aucune assistance ; l'homme doit prendre toutes les décisions et actions* » et "*10. L'ordinateur décide de tout, agit de manière autonome, sans tenir compte de l'homme*". Ce point de vue classique a été réévalué ces dernières années, en raison de l'ampleur des utilisations et des applications de l'intelligence artificielle au cours de la dernière décennie. Dans [Shn20], l'auteur propose un nouveau cadre théorique bi-dimensionnel pour raisonner sur ce sujet. Dans cette perspective, le contrôle humain et l'automatisation ne sont plus antagonistes, mais constituent des variables indépendantes : un système peut combiner à la fois un niveau élevé d'automatisation et un niveau élevé de contrôle humain, ce qui permet d'obtenir des systèmes fiables, sûrs et dignes de confiance (*reliable, safe and trustworthy*, RST). L'exemple d'un distributeur d'analgésiques illustre bien cette idée. Un système entièrement contrôlé par la·e patient·e, sans surveillance, permettrait un contrôle précis de l'utilisateur·ice, mais présenterait un risque de surdosage. Un système entièrement automatisé, sans aucun contrôle humain, garantirait une utilisation sûre, mais la·e patient·e pourrait subir une douleur évitable ou une sédation excessive. Dans ce cas, un système RST assurant à la fois un bon niveau de contrôle humain et d'automatisation consisterait en un distributeur contrôlé par la·e patient·e et doté d'un module de sécurité contrôlant la dose d'analgésique injectée, afin de réduire le risque de surdose ou d'état de manque.

Le domaine du traitement automatique du langage naturel (TALN) est aujourd'hui un exemple de la tension entre le contrôle humain et la performance. Par nature, comme cela est montré en philosophie du langage [Wit53], le langage naturel est ambigu, dépendant du contexte et implicite, et les langues construites logiques et non ambiguës telles que le *Lojban* [NC03] ou *Loglan* [Coo60] existent, mais ne comptent qu'un nombre restreint de locuteur·ices. Par conséquent, son traitement (génération, analyse, classification, etc.) est

une tâche difficile. Cependant, en raison de l'omniprésence du langage dans la vie humaine, la plupart des gens peuvent facilement résoudre de nombreuses tâches de TALN. Par conséquent, la question de l'implication humaine dans ces tâches se pose naturellement. On peut souligner que, si les premiers systèmes de TALN utilisaient des connaissances linguistiques de sens commun ou d'expert·es, avec le développement de l'apprentissage profond, l'utilisation de connaissances externes a en grande partie disparu, et l'intervention humaine ne subsiste que pour l'annotation des données. Cette évolution culmine aujourd'hui avec les grands modèles de langue basés sur des transformateurs, tels que BERT [Dev+19] pour l'analyse ou GPT [Bro+20] pour la génération. Cette évolution s'accompagne d'un gain impressionnant en termes de performances, mais malgré ces progrès, de nombreuses tâches restent trop difficiles pour être entièrement automatisées sans intervention humaine.

La construction de graphes de connaissances à partir de textes est une tâche importante qui reste difficile à automatiser. Cette tâche est apparue du fait de plusieurs dynamiques parallèles. Normalisé en 2001 par Tim Berners-Lee [BHL01], et malgré une lente émergence, le web sémantique est devenu au cours de la dernière décennie un domaine dynamique, à la fois dans l'industrie et la recherche : des graphes de connaissances massifs tels que YAGO [PWS20] ou Wikidata [VK14] sont apparus, le *linked open data cloud*[3] contient plus de 1 250 graphes de connaissances, et il existe de nombreuses techniques et outils pour explorer, construire, compléter, raisonner sur ou interroger les graphes de connaissances. Parallèlement, la popularisation des ordinateurs personnels et d'internet depuis 1995 a entraîné une numérisation massive des textes, et il est aujourd'hui raisonnable de dire que la plupart des connaissances humaines existent sous la forme de données numériques textuelles. Ces deux dynamiques créent naturellement un besoin d'intégrer ces connaissances au web sémantique. Couplé aux progrès décrits ci-dessus dans le domaine du TALN, cela a naturellement provoqué l'émergence de méthodes pour construire des graphes de connaissances à partir de textes.

Si l'apprentissage profond – et plus généralement l'apprentissage statistique – est devenu le paradigme prédominant en IA ces dernières décennies, ce domaine a une longue histoire avec le raisonnement automatisé et l'intelligence artificielle symbolique. Ces deux domaines ont pour objectif d'étudier et de reproduire des processus de raisonnement pour comprendre ou représenter des données et extrapoler des connaissances, sur la base de représentations symboliques de ces données. Ceux-ci possèdent des ramifications dans de nombreux domaines tels que l'analyse de données, le data mining, l'analyse

3. `https://lod-cloud.net/`

de programmes, les méthodes formelles ou le raisonnement sémantique, et recoupent d'autres domaines tels que les mathématiques, l'informatique théorique, l'algorithmique ou la philosophie. Parmi ceux-ci, l'analyse formelle des concepts (*formal concept analysis*, FCA) [GW99] est un cadre mathématique permettant de raisonner sur des données symboliques en les décrivant en termes d'*objets* ayant des *attributs* et en les regroupant en *concepts*. Formalisée pour la première fois en 1982 [Wil82], la FCA reste à ce jour dotée d'une communauté de recherche active, avec des applications dans de nombreux domaines, de la robotique [Zha+23] à l'exploration de règles [BKO21]. En particulier, des travaux ont été développés pour adapter ce cadre aux tâches d'apprentissage automatique [Kuz04; Kuz13b]. L'apprentissage automatique basé sur la FCA a la propriété intéressante d'être interprétable, car il repose sur l'exploitation de concepts formels, dont la description est symbolique.

Dans cette thèse, nous présentons une théorisation d'une IA centrée sur l'utilisateur·ice pour la construction de connaissances à partir de textes. Se présentant d'abord comme une interface complète pour la construction de graphes de connaissances à partir de textes, cette approche est basée sur une automatisation progressive des actions de l'utilisateur·ice. Pour ce faire, elle met en œuvre un système explicable de suggestions basé sur la FCA, utilisé pour produire des suggestions de qualité croissante à partir des actions passées de l'utilisateur·ice, et permettant à cellui-ci – ou à d'autres – de valider ou de rejeter les explications pour les suggestions valides. Les explications validées sont ensuite transformées en règles d'inférence, afin d'automatiser entièrement les cas similaires. La structure du workflow de ce système, en particulier le cycle "*production manuelle* → *suggestion* → *automatisation*", peut être adaptable à d'autres tâches de transformation de données. Comme les principales normes utilisées pour représenter des graphes de connaissances sont limitées aux connaissances factuelles, nous limitons l'application aux textes factuels. Ces textes tendent à être spécifiques à un domaine et ont donc un vocabulaire restreint, ce qui permet une automatisation efficace du processus. Par conséquent, les cas d'utilisation pourraient être la numérisation du contenu de corpus de textes factuels spécifiques à un domaine – par exemple, des rapports de procès ou d'expertise médicale – pour une exploitation plus facile de leur contenu. Afin de développer un tel système, cette thèse présente également des contributions en FCA, à travers l'extension des travaux théoriques et algorithmiques derrière la notion de *concepts de voisins* [Fer17a]. Ces travaux sont ensuite utilisés pour développer une nouvelle approche pour l'extraction de relations explicables, qui est une tâche centrale dans la construction de graphes de connaissances à partir de textes.

Contributions

Le travail présenté dans cette thèse est composé de plusieurs contributions, à la fois théoriques et appliquées, dans plusieurs domaines de recherche – IA centrée sur l'utilisateur·ice, traitement automatique du langage naturel, analyse formelle de concepts, extraction d'informations et web sémantique.

Workflow centré sur l'utilisateur·ice pour la construction de graphes de connaissances Nous présentons un nouveau modèle de workflow centré sur l'utilisateur·ice pour la construction de graphes de connaissances à partir de textes. Celui-ci est basé sur une modélisation sémantique et syntaxique intermédiaire des textes, générée par une unité de prétraitement. Il comporte également une unité interactive permettant au système d'apprendre à partir des actions de l'utilisateur·ice, et une unité automatisée permettant d'inférer des connaissances fiables directement à partir des actions passées de cellui-ci. Dans son utilisation, ce workflow suit le cycle "*production manuelle* → *suggestions* → *automatisation*" : il est conçu pour agir d'abord comme une interface manuelle pour la création de graphe de connaissances, sur laquelle un module de suggestion vient assister l'utilisateur·ice. Iel peut alors permettre l'automatisation complète de l'extraction des faits en validant certaines de ces suggestions. Ce travail a été publié dans [Aya22].

Modélisation du texte en tant que graphe Dans le prolongement de la contribution précédente, nous proposons une modélisation syntaxique et sémantique du texte sous forme de graphe, utilisable comme représentation intermédiaire dans le workflow précédent. Cette modélisation, basée sur l'utilisation d'outils classiques de TALN pour l'analyse syntaxique et sur une base de données lexicale, utilise le standard RDF – dont *RDF Schemas* – pour sa représentation. Cette contribution est publiée dans [ACF21; ACF22a].

Concepts de voisins Dans le domaine de l'analyse formelle de concepts, plusieurs contributions ont été apportées au cadre mathématique et algorithmique qui sous-tend la notion de concepts de voisins. Tout d'abord, nous formalisons cette notion dans le paradigme général de l'analyse formelle de concepts, au lieu de Graph-FCA, une de ses extensions aux données graphe. Ensuite, nous complétons le travail existant dans Graph-FCA pour permettre un traitement efficace du cas des concepts n-aires. Nous formalisons également la transformation des graphes RDF en graphe contextes – le format des données

dans Graph-FCA – en tenant compte des schémas RDF. Enfin, nous présentons CONNOR, une bibliothèque Java pour le calcul des concepts de voisins sur les graphes RDF. Ces travaux sont publiés dans [ACF24; ACF22b].

Extraction de relations dans les textes Nous proposons une nouvelle approche pour l'extraction de relations dans les textes, basée sur les concepts de voisins et conçue pour être compatible avec le workflow introduit précédemment. Tout d'abord, nous présentons une architecture en deux étapes pour l'extraction de relations dans les textes, basée sur la séparation entre la détection des relations (différencier les exemples positifs des négatifs) et la classification des relations (identifier le type de relation pour les exemples positifs). Deuxièmement, nous introduisons un module de détection des relations, en affinant un large modèle de langue existant. Ce module est également conçu pour faire partie de l'unité de prétraitement du workflow. Troisièmement, nous proposons une nouvelle approche symbolique et explicable pour la classification des relations, basée sur la modélisation des textes en tant que graphes et sur les concepts de voisins. Cette approche est également conçue pour faire partie de l'unité interactive du workflow présenté précédemment. Enfin, nous évaluons, séparément et conjointement, ces modules sur les tâches de détection, de classification et d'extraction de relations sur un benchmark d'extraction de relations bien connu. Ce travail a été publié dans [ACF21; ACF22a].

Table of Contents

INTRODUCTION

Artificial Intelligence is usually defined as the attempt to artificially reproduce an equivalent to human cognition. This is formulated in the famous question of Allan Turing in his seminal paper [Tur50]: *"Can machines think ?"*. This definition, vague and theoretical, masks in reality an umbrella term. In practice, it groups a set of techniques, more or less directly copying human cognition, that practically aims to automate tasks that are usually considered as intellectual, *i.e.*, that have an apparent need for high-level cognitive processes over mechanical processes, and therefore were considered as hardly automatable before the birth of computer science. It can be pointed out that the line between intellectual tasks and manual tasks is blurry, as most traditionally human-made tasks make use of high-level cognitive processes; by consequence, the difference between mechanical automation and artificial intelligence is mostly arbitrary. However, this points out the fact that AI, more than a new paradigm in human production, is overall a natural extension, caused by the invention of computing technologies, of the global automation process of production – whose origins cannot be properly traced, but that accelerated and became predominant since the industrial revolution. This aspect of AI became particularly clear this last decade, with the explosion of deep learning which, despite the initial inspiration of the neural networks as a copy of biological neurons, has not for main goal to reproduce or imitate human cognition, but to solve a task through the learning from massive input-output datasets [1]. However, this distance with natural cognition causes a major issue, as shown in [Rud19] and through diverse controversies [2]: how can a result be trusted if the intermediate steps for obtaining this result cannot be understood and if the user has no control on the produced output? The question of explainability became a predominant field in AI those last years [BH21], while, despite the development of works on interactive systems, the question of human control and of human-centered AI is still understudied.

The most classical point of view on the question of computer automation and human control, as formalized by Parasuraman *et al.* [PSW00] for example, is the trade-off model

1. This is why the term *machine learning* is sometime preferred to the term *artificial intelligence* to describe this kind of technology.

2. The most famous one is probably the case of *COMPAS*, a black-box system used in some US courts to evaluate the recidivism risk of a defendant, and which has shown a racist bias.

between those two aspects. In this conception, automation and human control are antagonized: having high human control implies having low automation, and having high automation implies having low human control. For example, Parasuraman proposes a scale for evaluating the level of automation and human control of a system, rating it between "*1. The computer offers no assistance; human must take all decisions and actions*" and "*10. The computer decides everything, acts autonomously, ignoring the human*". This classical point of view has been reevaluated those last years, due to the extent of the usages and applications of Artificial Intelligence this last decade. In [Shn20] the author proposes a new bi-dimensional framework for reasoning on this subject. In this framework, human control and automation are no longer antagonized, but are independent variables: a system can combine both high automation and high human control, which allows for Reliable, Safe and Trustworthy (RST) systems. This can be illustrated with the example of a painkiller dispenser. A fully patient-controlled system without surveillance would allow a fine control of the user, but would present a risk of overdose. A fully automated system with no human control would ensure a safe usage, but the patient may suffer excessive pain or may be over-sedated. In this case, an RST system ensuring both a good level of human control and automation would consist in a patient-controlled dispenser having a safety module monitoring the painkiller dose injected, in order to reduce risk of overdose or craving.

The field of Natural Language Processing (NLP) is today an example of the tension between human control and performance. By nature, as studied in philosophy of language [Wit53], natural language is ambiguous, context-dependent, and implicit, while logical, non-ambiguous constructed languages such as *Lojban* [NC03] or *Loglan* [Coo60] exist but keep a restrained number of speakers. Therefore, its automated processing (generation as well as analysis, classification and so on) is a hard task. However, due to the omnipresence of language in human life, most people can easily resolve many NLP tasks. Therefore, the question of human implication on these tasks naturally appears. It can be pointed out that, if the first NLP systems were using common sense or expert linguistic knowledge, with the growth of deep learning, expert intervention disappeared, and human intervention only remains on data annotation. This culminates today with the transformer-based large language models, such as BERT [Dev+19] for analysis or GPT [Bro+20] for generation. This evolution comes with an impressive gain in terms of performance, but despite those progresses, many tasks stay too hard to be fully automated without human intervention.

An important task that remains hard to fully automate is the knowledge graph construction from texts. This task emerged because of several parallel dynamics. Standardized in 2001 by Tim Berners-Lee [BHL01], and despite a slow emergence, the semantic web became during the last decade a dynamic field, both amongst industrials and researchers: massive knowledge graphs such as YAGO [PWS20] or Wikidata [VK14] appeared, the linked open data cloud [3] contains over 1,250 knowledge graphs, and techniques and tools to explore, build, complete, reason on or query knowledge graphs have been developed. In parallel, the popularization of personal computers and of the internet since 1995 caused a massive digitalization of texts, and today it is reasonable to say that most of human knowledge exists in the form of textual data. These two dynamics naturally creates a need for integrating this knowledge to the semantic web. Coupled to the progresses described above in NLP, this naturally caused an emergence of methods for constructing knowledge graphs from texts.

If deep learning – and more generally statistical learning – became the predominant paradigm in AI those last decades, this domain has a long history with automated reasoning and symbolic artificial intelligence. Those two fields have for objective to study and reproduce reasoning processes for understanding or representing data and extrapolating knowledge, based on symbolic representations of those data. These domains have ramifications in numerous fields such as data analysis, data mining, program analysis, formal methods or semantic reasoning, and intersect with other domains such as mathematics, theoretical computing, algorithmic or philosophy. Amongst those, Formal Concept Analysis (FCA) [GW99] is a mathematical framework for reasoning on symbolic data by describing it in terms of *objects* having *attributes* and by grouping them into *concepts*. First formalized in 1982 [Wil82], FCA still has an active research community, with applications in numerous fields, from robotics [Zha+23] to rule mining [BKO21]. In particular, works have been developed to adapt this framework to machine learning tasks [Kuz04; Kuz13b]. FCA-based machine learning has the interesting property of being interpretable, as it relies on the exploitation of formal concepts, whose description is symbolic.

In this thesis, we present a theorization of a user-centric AI for knowledge construction from text. First presenting itself as a comprehensive interface for the construction of knowledge graphs from text, this approach is based on a progressive automation of the user's actions. To do so, it implements an FCA-based explainable suggestion system, used to produce suggestions of increasing quality based on the user's past actions, and to

3. `https://lod-cloud.net/`

allow the user to validate or reject explanations for the valid suggestions. The validated explanations are then transformed in inference rules, in order to fully automate the similar cases. The structure of this system's workflow, in particular the cycle "*manual production* → *suggestion* → *automation*", is thought to be adaptable to other data transformation task. As the main knowledge graph standards are restrained to factual knowledge, we restrain the application to factual texts. Datasets of such texts tends to be domain specific, and therefore having a restrained vocabulary, which allows for an efficient automation of the process. Therefore, use-cases could be the digitalization of the content of factual domain-specific corpora of texts – e.g., trial reports or medical expertise reports – for an easier exploitation of their content. In order to develop such a system, this thesis also presents contributions in FCA, through extending the theoretical and algorithmic work behind the notion of *Concepts of Neighbors* [Fer17a]. This work is then used to develop a novel approach for explainable relation extraction, which is a central task in knowledge graph construction from text.

Contributions

The work presented in this thesis is composed of several contributions, both theoretical and applied, in several research fields – User-Centric AI, Natural Language Processing, Formal Concept Analysis, Information Extraction and Semantic Web.

User-centric Workflow for Knowledge Graph Construction We introduce a novel workflow model for user-centric knowledge graph construction from text. This workflow is based on an intermediate semantic and syntactic modeling of texts produced by a pre-processing unit. It also features an interactive unit for the system to learn from the user's action, and an automated unit, for inferring reliable knowledge directly from the user's past actions. In its use, this workflow is following on the "*manual production* → *suggestion* → *automation*" cycle: it is conceived to act at first as a manual interface for the knowledge graph completion, on which a suggestion module comes to assist the user. The user then can allow for the full automation of the extraction of facts by validating some of those suggestions. This work has been published in [Aya22].

Text Modeling as Graphs In the following of the previous contribution, we propose a syntactic and semantic modeling of text as graph to use as intermediate representation in

the previous workflow. This modeling, based on the use of classical NLP tools for syntactic analysis and on a lexical database, makes use of the RDF standard – in particular of *RDF Schemas* – for its representations. This contribution is published in [ACF21; ACF22a].

Concepts of Neighbors In the domain of Formal Concept Analysis, several contributions were made to the mathematical and algorithmic framework underlying the notion of concepts of neighbors. First, we formalize this notion in the general paradigm of *formal concept analysis* (FCA), instead of Graph-FCA as before. Then, we complete the existing work in Graph-FCA to make the efficient handle of the case of n-ary concepts easier. We also formalize the transformation of RDF graphs into graph contexts, taking into account RDF schemas. Finally, we introduce CONNOR, a Java library for computing concepts of neighbors on RDF graphs. These works are published in [ACF24; ACF22b].

Relation Extraction in Texts We propose a novel approach for relation extraction in text, based on Concepts of Neighbors and conceived to be compatible with the introduced workflow. First, we present a two-step architecture for relation extraction in texts, based on the separation between relation detection (differencing positive examples from negative ones) and relation classification (identifying the relation type for positive examples). Second, we introduce a relation detection module, by fine-tuning an existing Large Language Model. This module is also tailored to be part of the pre-processing unit of the previously introduced workflow. Third, we propose a novel, symbolic and explainable approach for relation classification, based on the modeling of texts as graphs and on the Concepts of Neighbors. This approach is also tailored to be part of the interactive unit of the previously introduced workflow. Finally, we evaluate, separately and conjointly, those modules on the relation detection, classification, and extraction tasks on a well-known relation extraction benchmark. This work has been published in [ACF21; ACF22a].

Thesis Outline

This thesis is divided in two parts. **Part I** presents the state of the art and elements of theoretical background. **Chapter 1** summarizes the existing works in the domains adjacent to this thesis, namely knowledge graph construction, user-centric AI, relation extraction and Formal Concept Analysis. **Chapter 2** presents the pre-existing theoretical background in Formal Concept Analysis used in the rest of the thesis. **Part II** presents

the different contributions of this thesis, divided in three chapters. **Chapter 3** introduces the novel workflow model for user-centric knowledge graph construction from text, as well as the modeling of texts as RDF graphs. **Chapter 4** presents the different contributions made in the domain of Formal Concept Analysis and introduces CONNOR, a Java library for the computation of Concepts of Neighbors. **Chapter 5** presents the new, explainable, two-step relation extraction method, by detailing its different modules and presenting a full experimental evaluation of it. Finally, **Chapter 6** recapitulates the content of the thesis, discusses the contributions and presents different perspectives for future works.

Part I

State-Of-The-Art

Chapter 1

Related Work

The work presented in this thesis overlaps several active research fields: natural language processing, semantic web, symbolic and statistical machine learning, graph mining. This chapter presents the state of the art in the fields and tasks analogous to our work. This chapter is structured as follows. Section 1.1 summarizes the history behind the notion of knowledge graphs, and presents the different existing approaches for constructing knowledge graphs. Section 1.2 presents the different methods for knowledge discovery in graph data, through graph mining, rule mining and instance-based learning on graphs. Finally, Section 1.3 presents the existing works on user-centric AI and interactive data mining.

1.1 Knowledge Graphs and their Construction

As shown in the chronological review by Gutierrez and Sequeda [GS21], the desire to represent knowledge in data can be traced back to the origins of computer science : semantic networks were introduced as early as 1956 in the context of machine translation [Ric56], and many formalisms from different fields were proposed for representing knowledge during the following decades, before converging with the normalization by the *World Wide Web Consortium* (W3C) in 1999 [LS99] and the 2001 seminal paper on the Semantic Web [BHL01]. In this section, we present the RDF formalism and its overlay for ontology representation, before summarizing the existing approaches for knowledge graph construction.

1.1.1 Knowledge Graph Representation

Standardized in 1999 by the W3C [LS99], the *resource description framework* (RDF) became the dominant format for knowledge graph notations.

Definition 1 *Let $\mathcal{U}$ be the set of unified resources identifiers (URI) as defined by the*

Subject	**Relation**	**Object**
`urn:royal:person#william`	`urn:royal:rel#spouse`	`urn:royal:person#kate`
`urn:royal:person#kate`	`urn:royal:rel#spouse`	`urn:royal:person#william`
`urn:royal:person#george`	`urn:royal:rel#parent`	`urn:royal:person#kate`
`urn:royal:person#george`	`urn:royal:rel#parent`	`urn:royal:person#william`
`urn:royal:person#charlotte`	`urn:royal:rel#parent`	`urn:royal:person#kate`
`urn:royal:person#charlotte`	`urn:royal:rel#parent`	`urn:royal:person#william`
`urn:royal:person#kate`	`rdf:type`	`urn:royal:gender#female`
`urn:royal:person#william`	`rdf:type`	`urn:royal:gender#male`
`urn:royal:person#charlotte`	`rdf:type`	`urn:royal:gender#female`
`urn:royal:person#george`	`rdf:type`	`urn:royal:gender#male`
`urn:royal:person#kate`	`urn:royal:rel#birthYear`	`"1982"`
`urn:royal:person#william`	`urn:royal:rel#birthYear`	`"1982"`
`urn:royal:person#charlotte`	`urn:royal:rel#birthYear`	`"2015"`
`urn:royal:person#george`	`urn:royal:rel#birthYear`	`"2013"`

Table 1.1 – Textual representation of an RDF graph presenting an extract of the British royal family

W3C in [BFM05], $\mathcal{B}$ *be an infinite set of variables called* ***blank nodes****, and* $\mathcal{L}$ *the set of finite alphanumerical strings called* ***literals****. A* ***RDF graph*** *is a set* $\mathcal{T}$ *of triples, each triple being of the form* (s, r, o) *with the* ***subject*** $S \in \mathcal{U} \cup \mathcal{B}$*, the* ***relation*** $R \in \mathcal{U}$ *and the* ***object*** $O \in \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$*.*

Table 1.1 presents an example of RDF graph in the form of triples, and Figure 1.1 shows its graphical representation. Each triple is the representation of an atomic fact – *e.g.*, the triple (`kate`, `birthyear`, "`1992`") expresses the fact that Kate was born in 1982. The nodes – called *entities* – are URIs representing members of the British royal family and literals representing years. The edges – called *relations* – are URIs representing properties of those entities and the URI `rdf:type` used for typing the entities.

In a RDF graph, blank nodes form anonymous entities (*e.g.*, an entity with an unknown value), while literals give fixed alphanumeric values (*e.g.*, dates). In practice, the RDF standard expresses labelled directed multigraphs, as multiple triples can have the same subject and object.

Several notations for RDF graph serialization exist. The more used are the XML-RDF notation [Swa04] – representing the RDF triples in a XML tree – and the Turtle notation [Bec+23] – using a more readable and less verbose structure.

In addition to the triples representing facts (also called *A-box*), an RDF graph can also contain a series of triples representing an ontology (also called *schema* or *T-box*) on the

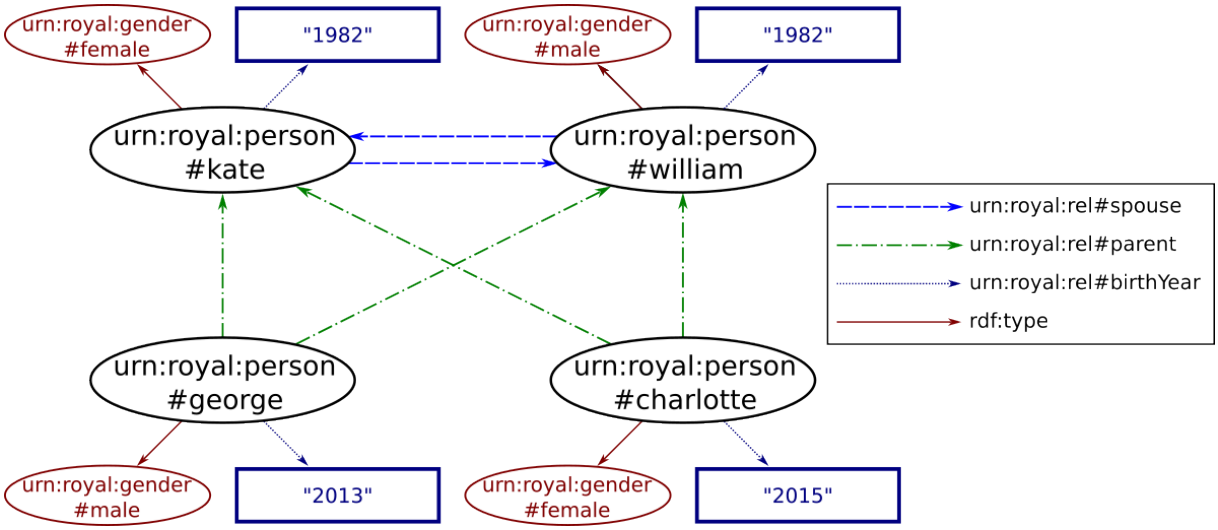

Figure 1.1 – Graphical representation of the RDF graph presented in Table 1.1

Subject	**Relation**	**Object**
`urn:royal:gender#male`	`rdfs:subClassOf`	`urn:royal:type#person`
`urn:royal:gender#female`	`rdfs:subClassOf`	`urn:royal:type#person`
`urn:royal:rel#parent`	`rdfs:subPropertyOf`	`urn:royal:rel#ancestor`
`urn:royal:rel#parent`	`rdfs:domain`	`urn:royal:type#adult`
`urn:royal:rel#spouse`	`rdfs:domain`	`urn:royal:type#adult`
`urn:royal:rel#spouse`	`rdfs:range`	`urn:royal:type#adult`

Table 1.2 – Example of RDF schema

graph (see [HKR09]), in the RDFS language. This schema gives extra information on the facts by adding a hierarchy over the types (with the triples of relation `rdfs:subClassOf`) and over the relations (with the triples of relation `rdfs:subPropertyOf`), and type constraints on the subjects and objects of a relation (with the triples of relation `rdfs:range` or `rdfs:domain`).

Table 1.2 presents an example of RDF schema for the RDF graph presented in Table 1.1. This schema expresses that an entity typed as male or female is a person, that the parent of someone is also their ancestor, that the object of a `parent` relation is an adult and that entities lined by a `spouse` relation are adults.

The role of RDF schema is to model knowledge on the vocabulary of an RDF graph, so that more triples can be inferred. This inference step is called *saturation*. Several strategies exist for saturating an RDF graph, depending on the application: either saturation is

Subject	**Relation**	**Object**
urn:royal:person#kate	rdf:type	urn:royal:type#person
urn:royal:person#william	rdf:type	urn:royal:type#person
urn:royal:person#george	rdf:type	urn:royal:type#person
urn:royal:person#charlotte	rdf:type	urn:royal:type#person
urn:royal:person#george	urn:royal:rel#ancestor	urn:royal:person#kate
urn:royal:person#george	urn:royal:rel#ancestor	urn:royal:person#william
urn:royal:person#charlotte	urn:royal:rel#ancestor	urn:royal:person#kate
urn:royal:person#charlotte	urn:royal:rel#ancestor	urn:royal:person#william
urn:royal:person#kate	rdf:type	urn:royal:type#adult
urn:royal:person#william	rdf:type	urn:royal:type#adult

Table 1.3 – Triples inferred from the A-box presented in Table 1.1 and the T-box presented in Table 1.2

made after each edition of the graph – which is a relatively cheap operation, but has to be applied often –, or saturation is done only when needed, and then can be more costly but rarely executed.

In the case of the A-box presented in Table 1.1 and the T-box presented in Table 1.2, the saturation produces the triples presented in Table 1.3.

Let us point out that RDF schema ontology can be used to infer new facts, but it cannot invalidate facts. It also cannot express more complex rules such as "an ancestor of an ancestor of an entity is an ancestor of this entity", or any other properties using cardinality, symmetry, equality and so on. However, RDF-based tools exist for those features. Constraints on the dataset can be expressed through the *shapes constraint language* (SHACL) [W3C17], while more complex ontologies can be expressed with the *web ontology language* (OWL) [Cal+12] – which is an overlay of RDF.

1.1.2 Knowledge Graph Construction

Several types of approaches co-exist for the construction and edition of knowledge graphs: manual – with tools for free or guided edition – automated from structured data and automated from natural language texts. This last type of approaches are part of the *information extraction* field, and can be divided between approaches having no previous vocabulary (*i.e.*, classes and properties) – called *open information extraction* (open IE) approaches – and those using a pre-existing vocabulary – called *closed information extraction* (closed IE) approaches. See [MHL20] for a survey on the intersection of the semantic web and information extraction.

Manual approaches The most obvious approach for constructing knowledge graphs is the manual one, with systems used for the free edition of knowledge graphs, and several tools have been developed for facilitating it. The most famous in this category is Protégé [Mus15], an ontology editor developed since the end of the 1990s by Stanford University. Until today, some massive knowledge graphs are still built using manual approaches, such as Wikidata [VK14], which relies on open contribution through a form-like web interface.

UTILIS [HFD12], on its side, is a tool for assisted edition of knowledge graph. Its role is to assist a user when adding a new entity to a knowledge graph by looking up similar entities in the graph and suggesting on this basis triples to add to the graph.

Automated approaches from structured data Other existing massive knowledge graphs, such as YAGO [PWS20] or DBpedia [Aue+07] are built through automated extraction of facts from structured web pages. Both those approaches rely on the exploitation of websites known for having structured data. DBpedia is focused on the parsing of Wikipedia infoboxes and article structures, in order to extract general knowledge facts. On its side, YAGO relies on the exploitation of Wikidata to reprocess its contents in order to obtain a more easily processable knowledge graph.

Open IE approaches A few methods of knowledge graph construction from texts rely on *open information extraction* techniques, *i.e.*, information extraction without previous knowledge on the information to extract. In the case of knowledge graph construction, it traduces by an absence of preexisting ontology or vocabulary over the resulting knowledge graph. [MLR18] presents a method for the construction of reified knowledge graphs directly from natural language texts in English, based on grammatical patterns in texts for identifying entities and their linkages. This kind of approach raises several problems. First, querying and exploiting a knowledge graph without information on its structure or vocabulary is a hard task. Second, as there is no control on the extracted facts, there is no guarantee on their relevancy, and depending on the text it can produce a noisy result.

Closed IE approaches Today, most of the methods for knowledge graph construction from texts are *closed information extraction* methods, based on a pre-existing vocabulary – in our case entity types and relation types. In this context, knowledge graph construction can be divided in two different tasks: *(named) entity recognition* (extracting entities from text) and *relation extraction* (identifying the relation – if any – between two entities).

The goal of *named entity recognition* (NER) is to identify where are located the entities in the text, and associate a type to each of them. For this task, as for many NLP tasks, systems based on *Long Short Term Memory* recurrent neural networks (LSTM) were predominant, sometimes combined with other types of layers. We can cite LSTM-CNN [CN16] that combines LSTM with *Convolutional Neural Network* layers, or LSTM-CRF and Stack-LSTM that combine LSTM with *Conditional Random Field* layers [Lam+16]. Today, as for many other tasks, those models have been outdated by methods using pre-trained transformer-based *Large Language Models* (LLMs) such as ERNIE [Sun+20] or K-BERT [Liu+20].

The task of NER can be completed with two other tasks. First, *coreference resolution* aims to identify the textual spans that refer to the same entity in a text. See [Suk+20] for a recent review on the subject. Second, *entity linking* (also called entity disambiguation) has for objective to identify which entity of a knowledge graph a textual entity span is an instance of. This task is useful in the case of completion of a pre-existing knowledge graph. Amongst contemporary methods, we can cite EDKate [Fan+16], using a simple disambiguation model over a joint embedding of graph entities and text spans. Le and Titov [LT18] present another approach, exploiting latent representation of relations between coreferences of a same entity.

Most approaches addressing the *relation extraction* (RE) task use deep learning methods. Historically, convolutional neural networks [NG15] and LSTM [Xu+15] were used first, then were replaced by graph convolution networks methods [ZQM18; WZ19], which allow taking into account the syntactic structure of sentences. Currently, the approaches that give the best results for the RE task use LLMs such as BERT [Dev+19] and its variants [Jos+20; Yam+20]. In this category, RECENT [AHH19] fine-tunes a LLM for each pair $(subjecttype, objecttype)$, and DeepStruct [Wan+22] specialize a LLM for structure prediction before fine-tuning. However, the performance of those approaches – with an F-score between 70 and 77% on the TACRED benchmark [Zha+17] – are still too low to allow a full automation. In addition, those fully statistical approaches lack of explanations for their predictions, which limits the possibilities of introducing human control in the process to improve reliability.

Symbolic approaches have also been proposed for the RE task. Their performance are often lower than deep learning methods, but by definition they provide interpretable results that can be used in a process with human control. The first symbolic approaches use rules such as regular expressions [GLR06] or syntactic patterns [FKZ07]. However,

these rules are handcrafted, and thus those approaches are time-consuming and often devoted to a specific corpus – such as biomedical literature for example. Some symbolic approaches automatically learn the linguistic rules. For instance, [Cel+15] uses pattern mining techniques to automatically extract those rules. The method presented in [BZ11] combines symbolic and machine learning techniques and proposes to learn patterns from a list of seed terms, *i.e.*, pairs of entities known to be in some target relation. In [Lee+15], a symbolic approaches based on FCA is proposed.

Today, modular NLP toolboxes such as SpaCy [HM17] or Stanford CoreNLP [Man+14] include modules for information extraction tasks.

1.2 Knowledge Discovery in Graphs

The Concepts of Neighbors approach, to which this thesis contributes, can be seen as a method for knowledge discovery in graph data: the objective is to find concepts related to a specific object [1]. Each concept is a description of objects – forming a rooted graph pattern – paired with the set of the objects matching this description, both the description and the set of objects being maximal. Those concepts can then be transformed in rules or used to perform instance-based machine learning. In that sense, Concepts of Neighbors is a method to extract structured knowledge from graphs, and therefore is part of the knowledge discovery field. In this section, we present the existing approaches for knowledge discovery in graphs related to the Concepts of Neighbors. First, we summarize the existing approaches in pattern mining on graphs. Then, we present the state of the art for rule mining. Finally, we present the existing works on instance-based learning on graphs.

1.2.1 Graph Mining

The domain of graph mining has been largely explored these last decades, and several types of approaches can be distinguished. We first present the graph mining methods using a measure of support to determine the pertinent set of patterns. Then we introduce more recent approaches, that focus on selecting an interesting set of patterns based on the optimization of measures based in information theory notions. Finally, we introduce

1. As stated later, Concepts of Neighbors, as part of Graph-FCA, can handle the case of tuples of objects, but we omit this here for the sake of simplification.

FCA, a method for knowledge discovery on symbolic data, and its extensions to graph data. For more complete surveys on the subject of graph mining, refer to [Fou+20; JCZ13]

Frequence-Based Graph Mining The most classic ones, such as AGM [IWM00], FFSM [HWP03] or gSpan [YH02], are complete approaches, mining all patterns over a given frequency – as does the *Apriori* algorithm on itemset mining [Agr+94]. Specifically on knowledge graph mining, SWApriori [RNS20] transforms knowledge graphs into itemsets in order to apply Apriori on it. However, these methods encounter the pattern explosion problem: either the frequency threshold is too high and the approaches return almost no pattern, or the frequency threshold is too low, and an intractable number of patterns (millions or more) is returned. This problem led to more parsimonious graph mining approaches. Some of them such as Gprune [Zhu+07] rely on the user for specifying constraints on the patterns, others choose to select specific subsets of the frequent patterns, such as SPIN [Hua+04] for the maximal patterns – *i.e.*, the set of the frequent patterns such as for each pattern, none of the larger patterns are frequent – and CloseGraph [YH03] for the closed patterns – *i.e.*, the set of the largest frequent patterns such as for each pattern, no larger pattern has the same support.

Information Theory-Based Graph Mining More recently, frameworks have been developed to drastically reduce the amount of mined patterns: e.g., the *Minimum Description Length* (MDL) principle or the *Maximum Entropy* (MaxEnt) principle. Those principles, both based on information theory notions, aim to optimize a measure for selecting a significant set of patterns.

On one hand, the MDL principle states that *"the best model is the one that compresses the data the best"*, *i.e.*, the set of patterns – called *model* – that best represents a dataset is the one that minimizes the description of the dataset by the model [Gal22]. In MDL for data mining, the model and dataset description lengths are defined as the minimal number of bits needed to encode it. GraphMDL+ [BCF21] is an example of MDL-based approach for graph mining.

On the other hand, the MaxEnt principle states that, for a given dataset and given constraints, the most representative modeling of this dataset is the one of higher entropy respecting the constraints. This principle is used in [Lee+15] for graph mining, encoding user subjectiveness as constraints.

Formal Concept Analysis on graphs First introduced in [Wil82], Formal Concept Analysis (FCA) is a mathematical framework used in data analysis to extract knowledge from categorical data. As presented in the reference book [GW99], this framework is used on finite sets of *objects*, each object matching a set of *properties* (also called *attributes*), and describes them in terms of *formal concepts*. A concept is defined as a set of objects O_c (its *extension*) and a set of attributes A_c (its *intension*), with the property that all the objects of O_c have A_c as set of common attributes and that O_c contains all the objects matching the attributes in A_c. For example, if we take for context the human beings, the concept of *mother* has for intension A_c the attributes *being a woman* and *having a child*, and have for extension O_c all the people matching those two attributes. If we add an attribute (*e.g.*, "being adult") to – or remove one from – A_c, the pair (O_c, A_c) no longer forms a concept, as the intension no longer describes the extension. From a pattern mining perspective, we can state that FCA mine concepts, each concept being a pattern – its intension – and a set of objects matching this pattern – its extension.

Several extensions of FCA were conceived to handle more complex data than two-valued attributes and singular objects, such as Fuzzy FCA [Yan+08] that uses fuzzy logics or Logical Concept Analysis [FR00] that extends FCA to logical formulas. Amongst them, several are developed to handle relational data, especially with the growth of the semantic web field and its knowledge graphs. First, Ganter and Kuznetsov theorized the notion of *pattern structures* [GK01], which enables to use graph patterns as object descriptions. However, in this framework graphs are used to describe individual objects, not as a set of relationships between objects. *Relational Concept Analysis* (RCA) [Rou+13] has been theorized as an extension of FCA for handling relational data. However, RCA does not implement the handling of cycles in the intensions, as well as n-ary relations with $n > 2$. To solve this problem, the *Graph-FCA* framework [FC20] has been theorized in order to extend the FCA framework to directed labelled multi-hypergraphs, with no constraint on the form of the intensions. In Graph-FCA, the intensions are rooted – or *projected* – graph patterns, while the extensions are tuples of vertices of the graph matching this pattern. The main issue with FCA for relational data is its computational cost: the computation of the whole set of concepts in Graph-FCA become rapidly intractable for massive graphs.

1.2.2 Instance-Based Machine Learning

Instance-based machine learning – also called *lazy learning* – describes a paradigm in machine learning in which a local model is computed from the training data at each

query for performing a generation (or prediction, classification, ...). This opposes to *eager learning*, the dominant paradigm, in which a general model is computed from the training data, and is able to perform generation (resp. prediction, classification, ...) whatever is the query. The most famous instance-based classification algorithm is the *k nearest neighbors* (kNN) algorithm: for a given object to classify, it lists the k most similar individuals in the training data, and predict the most represented class amongst the annotations of those individuals.

As far as we know, instance-based learning on graph data has rarely been studied. The most known approach on this subject is Relational Instance-Based Learning (RIBL) [HWB01]: it consists in defining a numerical distance between items in a relational dataset and then applying an algorithm similar to the *kNN* algorithm to classify query instances. A similar approach has been used by UTILIS [HFD12], this time in the context of the instance-based guided edition of RDF graphs. However, as far as we know, all existing approaches use a numerical distance between objects.

Because of its relatively low computational cost, the idea of using instance-based learning to perform classification has been considered in FCA-based machine learning [Kuz04]: instead of computing the whole concept lattice of a context, it consists into computing only the concepts related to the object to be classified. Therefore, for a single classification, the number of concepts to compute is reduced from exponential to linear. This idea has been applied to relation classification in biomedical texts [Lee+15]. This principle has been extended with other techniques such as approximation, random sampling and parallelization to be applied to big data [Kuz13a; Kuz13b]. On a different task, information retrieval, the user query is considered as the query instance, and a notion of *cousin concepts* enables to find approximate answers to the user query and to rank them by increasing distance [CLN14]. However, the cousin concepts are found by navigating the concept lattice, which therefore has to be computed beforehand.

The notion of Concepts of Neighbors, initially presented in [Fer17a], has been developed at the intersection of those two aspects – FCA on relational data and FCA for machine learning – by proposing a theoretical and algorithmic frame for instance-centered computation of concepts in Graph-FCA. The main idea behind of Concepts of Neighbors is, for a given object, to compute only the concepts related to this object. In practice, this produces a set of concepts forming, by their extensions, a hierarchical partitioning of the objects of the graph. The produced concepts are then used as a symbolic distance – that can be degraded into a numerical one – allowing for the use of kNN-like algo-

rithms. Chapter 2 presents in detail the theoretical background of FCA, Graph-FCA and Concepts of Neighbors. This method has been used on several tasks, including knowledge graph completion [Fer20] and query relaxation [Fer18].

1.2.3 Rule Mining

Knowledge graphs aim to represent real world knowledge. But the real world knowledge – and the reality itself – is varying all the time, and a knowledge graph can only represent a fraction of it. Therefore, the knowledge graphs are *by nature* incomplete. This has been formulated as the *open world assumption*:

Definition 2 *The **open world assumption (OWA)** is the assumption that a given database is incomplete, and therefore that we cannot suppose the truth value of a fact that is not in the database. This is opposed to the **closed world assumption (CWA)**, in which we assume that an absent fact is false.*

Today, most applications on knowledge graphs work under OWA [2]. Thus, from this assumption, approaches have been developed to infer new triples based on the existing triples of a knowledge graph. This task is called *link prediction* – or *knowledge graph completion* – and aims, for a subject s and a relation r, predicting o such as the triple (s, r, o) is true [3]. Amongst different approaches for link prediction, a popular family of symbolic efficient approaches is called *rule mining*. Those approaches aim to discover reliable association rules based on the knowledge graph, and use those rules to predict new triples. As negative facts are not represented in knowledge graphs, rule mining approaches aim to produce *Horn rules*:

Definition 3 *For $\mathcal{T}$ a RDF graph, $\mathcal{E}$ the set of entities of $\mathcal{T}$, $\mathcal{R}$ the set of relations of $\mathcal{T}$, and $\mathcal{V}$ an infinite set of variables. A **Horn rule** $R : H \leftarrow B$ is a rule of the form:*

$$(s_0, r_0, o_0) \longleftarrow \bigwedge_{i=1}^{n} (s_i, r_i, o_i)$$

*such that for $i \in [0, n]$, the atoms of the body of the rule (s_i, r_i, o_i) are such that $s_i \in \mathcal{E} \cup \mathcal{V}$, $r_i \in \mathcal{U}$ and $o_i \in \mathcal{E} \cup \mathcal{V}$. H is called the **head** of the rule, and B the **body**.*

2. However, some applications use more restrictive assumptions, like the *partial completeness assumption*, that states that, for an entity s and a relation r, if o exists such as the knowledge graph contains the triple (s, r, o), then for any entity o', the triple (s, r, o') is either in the knowledge graph or false.

3. This link prediction instance can be denoted $(s, r, ?)$

An *instantiation* of a rule is a copy of the rule, where all variables have been substituted by entities, such that all triples of body are in the RDF graph. A *prediction* of a rule is the head atom of an instantiated rule. We denote $inst(R)$ the set of instantiations of a rule and $pred(R')$ the prediction of an instantiated rule R'.

Definition 4 *The **confidence** of a rule R is defined as:*

$$conf(R) = \frac{\#\{R' \in inst(R) \mid pred(R') \in \mathcal{T}\}}{\#inst(R)}$$

The confidence of a rule represents the proportion of instantiated rules having their head verified.

The goal of rule mining methods is to discover rules with a confidence over a given threshold. However, the search space of rules is too big to be fully explored. Then, each rule mining method introduces a *language bias – i.e.,* the subset of generated and tested rules – through its generation method. We here present several state-of-the-art rule mining methods.

AnyBURL AnyBURL [Mei+19] (for *Anytime Bottom-Up Rule Learning*) is a method for rule mining with the good property of being an *anytime* algorithm: the longer the execution is, the richer the result is. As its name indicates, this is a *bottom-up* method: rules are abstracted from patterns directly sampled from the knowledge graph. The patterns sampled by AnyBURL are *straight ground paths*, *i.e.*, paths without internal cycles (but the whole rule can form a loop). Each path is abstracted as a *straight ground path rule*, by using the first atom of the path as head of the rule and the rest of the atoms as body, and by replacing the entities in the queue of the rule by variables. The AnyBURL algorithm mines rules by increasing length: rules of length 1 are mined until most of the mined rules are redundant, the rules of length 2 are mined, and so on. The main drawback of this approach is the language bias: the rules cannot form internal cycles or trees, which causes a loss in expressiveness.

AMIE First presented in [Gal+13], then refined for faster execution in AMIE+ [Gal+15] and AMIE 3 [LGS20], AMIE (for *Association rule-Mining under Incomplete Evidence*) is one of the major rule-based approaches. This is a *top-down* method: the algorithm generates rules, keep those with high confidence and then refine them to get longer rules,

and so on, through a classical breadth-first search. The generation and the refinement processes are designed such as the mined rules respect those three given properties :

— **Connectivity**: two triples are *connected* if they share a variable or an entity. A rule is connected if each triple is connected transitively to every other triple of the rule. In other terms, the triples of the rule form a connected graph. For example, the following rule is not connected:

$$(v_s, \texttt{spouse}, v_o) \longleftarrow (v_1, \texttt{parent}, v_s) \wedge (v_2, \texttt{parent}, v_3)$$

— **Closure**: a variable in a rule is *closed* if it appears at least twice in the rule. A rule is closed if each variable of this rule is closed. For example, in the following rule, the variable v_2 is not closed, and thus the rule is not closed:

$$(v_s, \texttt{spouse}, v_o) \longleftarrow (v_1, \texttt{parent}, v_s) \wedge (v_1, \texttt{parent}, v_o) \wedge (v_2, \texttt{parent}, v_o)$$

— **Non-reflexivity**: a reflexive rule is a rule admitting triples of the form (x, r, x), with x a variable or an entity. For example, $(v_0, spouse, v_0)$ is a reflexive atom. Reflexive rules are not mined because reflexive atoms are absent of most knowledge graphs.

The language bias of this approach is much less strong than for AnyBURL: rules can have cycles and form trees. Contrarily to AnyBURL, AMIE is not anytime: for a given RDF graph $\mathcal{T}$, a minimum confidence C and maximum rule length l, it computes all closed, connected, non-reflexive rules with a confidence higher than C and a length smaller than l.

Concepts of Neighbors In [Fer20], Concepts of Neighbors are used for link prediction, by transforming the rooted graph patterns used as concept intensions into rule bodies. This approach is instance-based: instead of computing a set of general rules beforehand and using them for all instances of link prediction on this KG, we compute a set of specific rules for each link prediction instance.

Let $\mathcal{T}$ be a RDF graph, s an entity of $\mathcal{T}$ and r a relation of $\mathcal{T}$. In this approach, to resolve the link prediction problem $(s, r, ?)$, rules are generated by using intensions of the concepts of neighbors of s as rule bodies, and by generating triples for rule heads. For a given intension I having v_s as projected variable, two types of rules are generated:

— **by-copy rules**: $(v_s, r, o) \longleftarrow I$, for each entity o of the knowledge graph.

— **by-analogy rules**: $(v_s, r, v_o) \longleftarrow I$, for each variable $v_o \neq v_s$ appearing in the triples of the intension I;

By-copy rules semantically state that, when an entity e matches I as v_s, then it has a given fixed property. An example of a by-copy rule would be that *"If someone is the parent of someone else, then they are an adult"*:

$$R : (v_s, \texttt{rdf} : \texttt{type}, \texttt{adult}) \longleftarrow (v_0, \texttt{parent}, v_s)$$

By-analogy rules have a different semantics: the head triple is a shorter way to express – or generalize – the semantics expressed by the intension – or at least by a part of it. An example of by-analogy rule would be that *"if X has Y for parent of one of her parent, then X has Y for grandparent"*:

$$R : (v_s, \texttt{grandparent}, v_o) \longleftarrow (v_s, \texttt{parent}, v_1) \wedge (v_1, \texttt{parent}, v_o)$$

GBS More recently, [SL22] has introduced the *grammar-based pattern system* (GBS), a method using a generative grammar for rule mining. This grammar is used for generating item patterns, that are assembled to form triple patterns, themselves assembled to form rule patterns, that can be instantiated into rules. As AMIE, this approach is top-down: it generates abstract rule patterns, that are instantiated. Then the non-matching rules are pruned, and the kept ones are refined. Once rules are generated, a pruning strategy is applied in order to keep a set of rules that minimizes the overlap of the instantiation of head atoms.

The specificity of GBS is that, contrarily to the other approaches presented here, its main goal is not link prediction but RDF graph compression: the idea is to mine a set of rules, then to remove from the RDF graph the set of triples that are generated by these rules in order to minimize its size. Therefore, it can hardly be compared to the other approaches in terms of score on the link prediction task. However, it can be pointed out that effectively, the produced set of rules has a better compression ratio than AMIE, and the execution time is inferior to AMIE+ [4].

Comparison AnyBURL, AMIE and Concepts of Neighbors have similar performances on link prediction benchmarks. The difference resides in other properties. The main difference is that Concepts of Neighbors is instance-based: it mines rules specifically matching

4. No comparison with AMIE 3 – the fastest version of AMIE – has been found in the literature.

a specific link prediction instance $(e, r, ?)$, while AMIE and AnyBURL mine sets of general rules, with only some of them being usable for a given instance. Therefore, Concepts of Neighbors is more appropriate for knowledge graphs that are regularly updated, while AMIE and AnyBURL have to be preferred if the knowledge graph is fixed once and for all. Concerning computing performance, AMIE – especially with its enhancements AMIE+ and AMIE 3 – is faster for mining a great number of rules. However, AnyBURL and the Concepts of Neighbors have the property to be anytime, and therefore execution time can be fixed *a priori*.

1.3 User-Centric AI

As evoked in the introduction, the domain of user-centric AI is quite understudied. The framework proposed by Shneiderman [Shn20] – that replaces the unidimensional tradeoff between human control and automation by a bi-dimensional space allowing for simultaneous high human control and high computer automation – is still young, and the search for performance over user control in machine learning stays the dominant paradigm. However, several pre-existing systems for diverse applications are, by their relation with the user, analogous to the work presented in this thesis.

In the domain of data mining, [Lee14] presents methods for interactive data exploration, while [Lee+16] introduce a measure of subjective interestingness in graph mining.

As evoked earlier, in [HFD12] is presented UTILIS, a system for assisted edition of knowledge graphs based on a suggestion module using logical rules extracted from the description of existing objects. This approach is similar to the suggestion system developed in our work. In addition, its purpose is also the edition of knowledge graphs. However, contrarily to our work, it is neither text-centered nor featuring automation.

In [FR02], the authors present an FCA-based interactive system for progressive automation of e-mail classification. This approach is based on the incremental build of a *logical concept analysis* (LCA) context, based on the interaction with the user. By its use of FCA theory and by its incremental aspect based on the user's action, this approach is formally close to the work presented in this manuscript. However, it significantly differs by the difficulty of the task, and therefore by the complexity of the model: by the simple construction of a logical context, this approach solves the proposed task with accuracy superior to 85% after the processing of 100 individuals, while the best systems of the state-of-the-art scores a f-score of 0.75 on relation extraction, which is a subtask of knowledge

graph construction.

1.4 Conclusion

This chapter summarizes the existing scientific works related to the contributions presented in this thesis. First, it recapitulates the notions of semantic web and knowledge graph, its formalisms and the existing approach for the construction or edition of knowledge graphs. Then, it presents the state of the art in the domain of knowledge discovery on graphs, focusing on graph mining, instance-based machine learning and rule mining. Finally, it exposes the existing works in the domain of user-centric AI and interactive knowledge discovery.

As shown in this chapter, today the existing methods for constructing or editing knowledge graphs are very diversified: manual, with an automatic assistance, as well as fully automatized, from structured data and from texts, based on an existing vocabulary or not. The work presented in this thesis differs of those existing works by several aspects. Being a method for constructing knowledge graphs from texts, it is part of the information extraction field, but it differs from the other approaches by its user-centered aspect. As UTILIS [HFD12], it can be seen initially as an assisted manual method with a suggestion system, except that this method is based on texts, and that it features an automation process that, in the end, makes it similar to fully automated methods. In addition, our proposed system features interpretability properties, by the use of symbolic machine learning, that greatly differs from the state-of-the-art systems for information extraction.

Concepts of Neighbors, contrarily to other graph mining methods, produce a local set of patterns that are relevant to a query instance. Moreover, the generated patterns are *rooted patterns*, *i.e.*, they have a distinguished tuple of nodes that corresponds to the query instance. This property makes them fit for downstream inference tasks, such as instance-based machine learning, as evoked before.

Concerning the user-centered aspect, our work falls within the recent theoretical framework for human-centered AI developed in [Shn20]. It presents similarity with existing approaches, by its finality as well as by its functioning, but distinguish itself by the complexity of the task of construction of knowledge graphs from texts, and therefore by the complexity of the technologies used.

Chapter 2

Theoretical Background

An important part of the work presented in this thesis relies on notions developed in Formal Context Analysis (FCA), as introduced in Chapter 1. This chapter introduces the mathematical and algorithmic background developed in Formal Context Analysis and used in the presented work. Section 2.1 introduces the main definitions from the FCA framework. Section 2.2 presents Graph-FCA, an extension of FCA to graph data. Section 2.3 presents the notions related to the Concepts of Neighbors in the context of Graph-FCA, and the algorithmic framework developed for its computation.

2.1 Formal Concept Analysis

This section presents the basic notions of FCA, as introduced in the seminal paper [Wil82] and presented in the reference book [GW99].

Definition 5 *A **formal context** is a triple $K = (O, A, I)$ where O is a set of* objects*, A a set of* attributes *and $I \subseteq O \times A$ is an* incidence *relation between objects and attributes. For each object $o \in O$, we define $I(o) = \{a \in A \mid (o, a) \in I\}$ as the* description *of o.*

Table 2.1 gives an example of a formal context. The set of objects is an excerpt of the British royal family and the attributes are some human characteristics, here to be a man, a woman, an adult, a kid, and married. The incidence relation associates each person to their characteristics. For instance, in this context, Charles is a man, an adult and married.

Definition 6 *We define the **instances** of a set of attributes as the function*

$$\begin{aligned} inst\colon \mathcal{P}(A) &\to \mathcal{P}(O) \\ Y &\mapsto \{o \in O \mid Y \subseteq I(o)\} \end{aligned}$$

	man	woman	adult	kid	married
Charles	×		×		×
Charlotte		×		×	
Diana		×	×		×
George	×			×	
Harry	×		×		×
Kate		×	×		×
William	×		×		×

Table 2.1 – Example of a formal context where the set of objects is an excerpt of the British royal family.

that maps a set of attributes to the set of the objects that have all the attributes in their description[1].

Definition 7 *We define the* ***properties*** *of a set of objects as the function*

$$\begin{aligned} prop\colon \mathcal{P}(O) &\to \mathcal{P}(A) \\ X &\mapsto \bigcap_{o \in X} I(o) \end{aligned}$$

that maps a set of objects to the set of their common attributes[2].

For instance, in the context presented in Table 2.1, $inst(\{woman, adult\}) = \{Diana, Kate\}$ and $prop(\{Charles, William\}) = \{man, married, adult\}$.

Property 1 *The pair of functions* $(inst, prop)$ *forms a* ***Galois connection*** *between* $\mathcal{P}(O)$ *and* $\mathcal{P}(A)$*, i.e., for* $O_c \subseteq O$ *and* $A_c \subseteq A$*:*

$$prop(O_c) \subseteq A_c \iff inst(A_c) \subseteq O_c$$

For instance, with $A_c = \{man,\ adult,\ married\}$ and $O_c = \{Charles,\ Diana,\ Harry,\ Kate,\ William\}$ we have $prop(O_c) = \{adult,\ married\} \subseteq A_c$ and $inst(A_c) = \{Charles,\ Harry,\ William\} \subseteq O_c$.

Definition 8 *Let* K *be a formal context. A* ***concept*** *is a pair* $C = (O_c, A_C)$ *such that:*

— $O_c = inst(A_c)$

1. In the literature, $inst(Y)$ is usually denoted Y' or $ext(Y)$.
2. In the literature, $prop(X)$ is usually denoted X' or $int(X)$.

— $A_c = prop(O_c)$

*$O_c \subseteq O$ is called the **extension** of C and $A_c \subseteq A$ is called the **intension** of C.*

For example, in Table 2.1, $(\{Charles, William\}, \{man, married\})$ is not a concept whereas $(\{Charles, William, Harry\}, \{man, adult, married\})$ is a concept. Indeed:

$$ext(\{man, adult, married\}) = \{Charles, William, Harry\}$$
$$int(\{Charles, William, Harry\}) = \{man, adult, married\}$$

Definition 9 *Let $K = (O, A, I)$ be a formal context and $C_1 = (O_1, A_1)$ and $C_2 = (O_2, A_2)$ be two concepts of K. Concept C_1 is **more specific** than C_2 (or C_2 **more general** than C_1) – denoted by $C_1 \leq C_2$ – if and only if:*

— *$O_1 \subseteq O_2$ and*

— *$A_1 \supseteq A_2$.*

Property 2 *Let C_1 and C_2 be concepts. As (int, ext) is a Galois connection, C_1 is more specific than C_2 if and only if $\mathbf{A_2} \subseteq \mathbf{A_1}$*

For example, the concept of adult married man is more specific than the concept of man:

$$(\{Charles, Harry, William\}, \{man, adult, married\}) \leq (\{Charles, George, Harry, William\}, \{man\})$$

Theorem 1 *Let $K = (O, A, I)$ be a formal context. The set of all concepts of K ordered by generalization forms a **concept lattice**.*
It means that for all pairs of concepts of K C_1 and C_2 there exist:

— *an **infimum** $C_1 \wedge C_2$, i.e., the most general concept that is more specific than the 2 concepts;*

— *a **supremum** $C_1 \vee C_2$, i.e., the most specific concept that is more general than the 2 concepts.*

Figure 2.1 shows the concept lattice of the context given in Table 2.1. The concepts are represented by circles. Concepts – except the top and bottom ones – are labelled by their intension (in gray boxes on top of concepts) or by their extension (in white boxes on

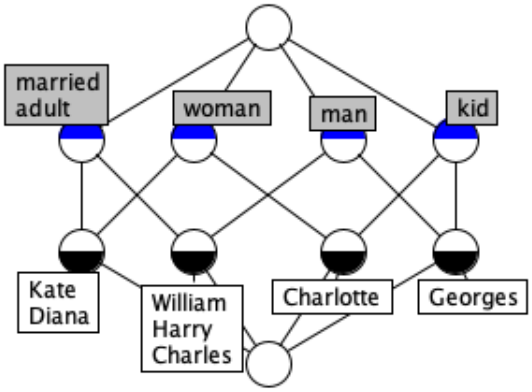

Figure 2.1 – Concept lattice derived from the context in Table 2.1.

the bottom of concepts). The concept labelled by "George" represents all persons in this context that are a man and a kid, for this example only George has those two attributes in its description. This concept is the infimum of the concepts labelled by "man" and the concept labelled by "kid".

2.2 Graph-FCA

Graph-FCA [Fer15; FC20] is an extension of FCA for knowledge graphs, and more generally for relational data. Indeed, in FCA, objects can be described individually but not the relationships between objects. The equivalent of a formal context for Graph-FCA is called a *graph context*.

Definition 10 *A* ***graph context*** *is a triple* $K = (O, A, I)$ *where* O *is a set of objects,* A *a set of attributes and* $I \subseteq O^* \times A$ *is an incidence relation between* tuples of *objects and attributes.*

Theoretically, this definition allows defining labelled directed multi-hypergraphs as graph contexts. However, for conciseness we only consider labelled directed multigraphs (and therefore $I \subseteq (O \cup O^2) \times A$), but definitions and concepts used below also apply to multi-hypergraphs. In such a context, $o \in O$ represents a vertex of the graph, $a \in A$ an

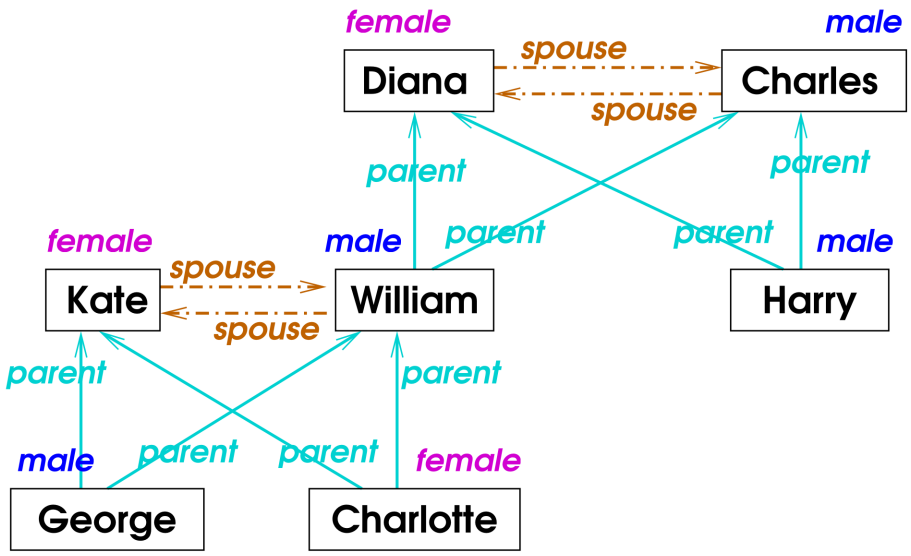

Figure 2.2 – Graph context representing the British royal family

incidence label (also called *attribute*) and $i \in I$ a unary (vertex label) or binary (labelled directed edge) hyperedge (also called *incidence*).

Figure 2.2 is a graphical representation of a graph context. It represents an excerpt of the British royal family. The boxes are the objects (e.g., *Charlotte* or *Harry*), the labels next to boxes are unary attributes (e.g., *man*, *woman*), and the arrow labels are binary attributes (e.g., *parent* or *spouse*). The unary incidence $((Kate), woman) \in I$ says that Kate is a woman; we call it a node label and also write it $woman(Kate)$ for readability and by analogy to predicate logic. The binary incidence $((George, Kate), parent) \in I$ says that Kate is a parent of George; we call it a labeled edge and also write it $parent(George, Kate)$.

The purpose of FCA is to discover *patterns* shared by objects. Whereas in classical FCA a pattern is a subset of attributes, in Graph-FCA a pattern is a *projected graph pattern*, which expresses a common graph structure rooted in one or several nodes.

Definition 11 *Let $\mathcal{V}$ be an infinite set of variables. A* ***graph pattern*** *$P \subseteq \mathcal{V}^* \times A$ is a set of pairs $(\overline{y}, a)$, with $\overline{y}$ a tuple of variables and a an attribute. Each of those pairs can be seen as an n-ary directed incidence (with $n = |\overline{y}|$) labelled by attribute a.*

A ***projected graph pattern (PGP)*** *is a pair $Q = (\overline{o}, P)$ where $\overline{o} \in \mathcal{V}^*$ a tuple of variables – called* ***projected variables*** *– and P is a graph pattern such as each incidence of P is transitively connected to at least one element of $\overline{o}$. The* ***arity*** *of a PGP is the length of $\overline{o}$. A PGP of arity k is also called a k-PGP. The set of PGPs of arity k over a set of attributes A is denoted by $PGP_k(A)$.*

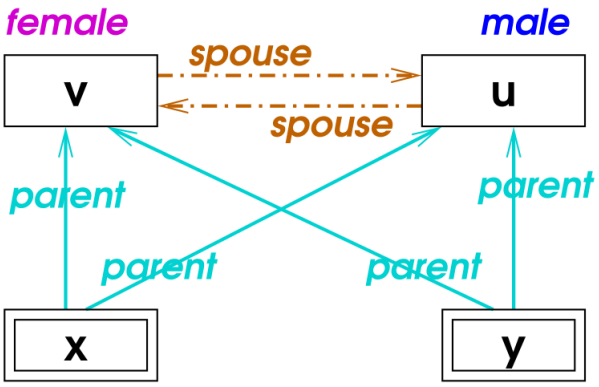

Figure 2.3 – PGP representing the relation between siblings of a married heterosexual couple

Figure 2.3 represents a 2-PGP describing the binary relation between siblings of a married heterosexual couple. Projected variables are in double boxes. In practice, PGPs can be seen as queries on the graph context, and we reuse the notation of such queries for the textual representation of PGPs:

$$P_{sibling}(x,y) = [x,y \leftarrow \quad man(u), woman(v), parent(x,u), parent(x,v), \\ parent(y,u), parent(y,v), spouse(u,v), spouse(v,u)]$$

A PGP **inclusion relation** – noted $Q_1 \subseteq_{PGP} Q_2$ – expresses that a k-PGP Q_1 is **more general** than another k-PGP Q_2, or that Q_2 is **more specific** than Q_1, *i.e.*, that by renaming the variables of Q_1 we can obtain a PGP having the same projection tuple than Q_2 and having its graph pattern included in the graph pattern of Q_2). If $Q_1 \subseteq_{PGP} Q_2$ and $Q_2 \subseteq_{PGP} Q_1$, the two PGPs are said equivalent ($Q_1 \equiv_{PGP} Q_2$). In addition, for two k-PGP Q_1 and Q_2, we define $Q_1 \cap Q_2$ as the most specific generalization of Q_1 and Q_2.

Besides, we define the *description* of the tuple of objects $\overline{o}$ as the PGP $Q(\overline{o}) = (\overline{o}, P(\overline{o}))$, where $P(\overline{o}) \subseteq I$ is the subset of incidences that are transitively connected to any element of $\overline{o}$. It is the union of the connected components of the graph context containing the elements of $\overline{o}$, rooted in those elements.

Definition 12 *We define the set of* **answers** *of a PGP as the set of answers of this PGP*

seen as a query.

$$\begin{aligned} ans\colon PGP_k(A) &\to \mathcal{P}(O^k) \\ Q &\mapsto \{\overline{o} \mid Q \subseteq_{PGP} Q(\overline{o})\} \end{aligned}$$

In the example PGP of Figure 2.3, the set of answers over the example graph context is made of the pairs $(Harry, William)$ and $(George, Charlotte)$, and also their inverse because of the pattern symmetry, and also the identity pairs like $(Harry, Harry)$ and $(Charlotte, Charlotte)$ because there is no inequality constraints between variables x and y. The answers of $Q(\overline{o})$ is the set of all tuples of objects that match everything that is known about $\overline{o}$.

Definition 13 *We define the* ***most specific query*** *of a set of k-tuples of objects as the largest query according to $\subseteq_{PGP}$ whose set of answers contains R.*

$$\begin{aligned} msq\colon \mathcal{P}(O^k) &\to PGP_k(A) \\ R &\mapsto \bigcap_{\overline{o} \in R} Q(\overline{o}) \end{aligned}$$

In the example graph context, the most specific query of Charlotte and Kate is that they are women: $msq(\{Charlotte, Kate\}) = [x \leftarrow woman(x)]$. The most specific query of Charlotte and William is $[x \leftarrow P_{sibling}(x, y), man(y)]$, where $P_{sibling}$ is the pattern of the above example PGP. It says that Charlotte and William have in common married parents ($P_{sibling}$) and a brother ($man(y)$). As proven in [FC20], the pair of functions (ans, msq) forms a **Galois connection** between $\mathcal{P}(O^k)$ and $PGP_k(A)$, for any k. This leads to the definition of graph concepts.

Definition 14 *Let $K = (O, A, I)$ be a graph context. A* ***graph concept*** *of arity k (also called a k-concept) is a pair $C = (R, Q) \in \mathcal{P}(O^k) \times PGP_k(A)$ such that $R = ans(Q)$ and $Q = msq(R)$. R is called the* ***extension*** *of the concept, and Q is called the* ***intension****.*

Definition 15 *Let $K = (O, A, I)$ be a graph context and $C_1 = (R_1, Q_1)$ and $C_2 = (R_2, Q_2)$ be two concepts of K.*
Concept C_1 is said ***more specific*** *than C_2 (or C_2* ***more general*** *than C_1) denoted by $C_1 \leq C_2$ if and only if $\mathbf{R_1} \subseteq \mathbf{R_2}$.*

Property 3 *As C_1 and C_2 are concepts and, as (int, ext) is a Galois connection, C_1 is more specific than C_2 if and only if $\mathbf{Q_2} \subseteq_{\mathbf{PGP}} \mathbf{Q_1}$*

Theorem 2 *Let $K = (O, A, I)$ be a formal context. The set of all k-concepts of K ordered by generalization forms a* ***concept lattice****.*
It means that for all pairs of concepts of K there exist:

- *an* ***infimum****, i.e., the most general concept that is more specific than the 2 concepts;*
- *a* ***supremum****, i.e., the most specific concept that is more general than the 2 concepts.*

2.3 Concepts of Neighbors

Concepts of Neighbors is a method for exploring the k-concept lattice of a graph context, by restricting this exploration to the concepts related to a given k-tuple of objects. This notion has been introduced in [Fer17a], and a reference journal paper on this subject is under review. In this section, we present the main notions related to Concepts of Neighbors, and gives a detailed overview of the existing algorithmic framework for the computation of concepts of neighbors.

2.3.1 Formal Definitions

The Concepts of Neighbors method relies on two major notions : the notion of conceptual distance and the notion of concepts of neighbors itself.

Definition 16 *Let K = (O, A, I) be a graph context. The* ***conceptual distance*** *between two k-tuples of objects $\overline{o}_1$ and $\overline{o}_2$ is the k-concept $\delta(\overline{o}_1, \overline{o}_2) = (R, Q)$ where the intension Q is the most specific k-PGP such that the extension R contains $\overline{o}_1$ and $\overline{o}_2$.*

Property 4 *Let $\overline{o_1}$ and $\overline{o_2}$ be two k-tuples of objects, and $\delta(\overline{o}_1, \overline{o}_2) = (R, Q)$. Then:*

- $Q = msq(\overline{o_1}) \cap msq(\overline{o_2})$
- $R = ans(Q)$

It can be proven that this conceptual distance has properties similar to a numerical distance, such as positivity, symmetry and triangle inequality, by using the concept extension inclusion as a partial order and a notion of conceptual supremum as addition [Fer17a]. The **extensional distance** $d(\overline{o}_1, \overline{o}_2) = |\delta(\overline{o}_1, \overline{o}_2).ext|$ can be used as a (degraded) numerical distance to evaluate the dissimilarity between o_1 and o_2 as the number of k-tuples between them.

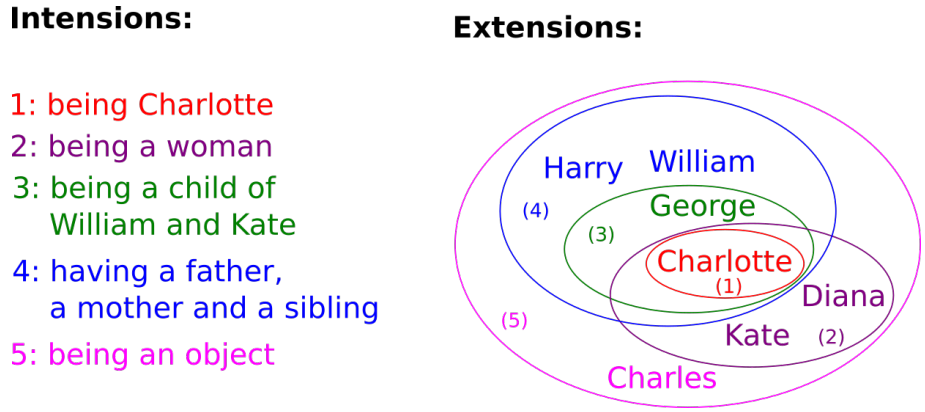

Figure 2.4 – Concepts of Neighbors of Charlotte

Definition 17 *The* ***concepts of neighbors*** *of a k-tuple $\overline{o}$ is the set of k-concepts $C\text{-}N(\overline{o}) = \{\delta(\overline{o}', \overline{o}) | \overline{o}' \in O^k\}$.*

The rank of a concept $\delta \in C\text{-}N(\overline{o})$ is recursively defined as:
— $rank(\delta(\overline{o}, \overline{o})) = 0$
— $\forall \delta \in C\text{-}N(\overline{o}), rank(\delta) = 1 + max\{rank(\delta') | \delta' \in C\text{-}N(\overline{o}), \delta'.ext \subseteq \delta.ext\}$

The **proper extension** of a concept $\delta.proper$ for $\delta \in C\text{-}N(\overline{o})$ is the set of k-tuples of its extension that are not in the extension of a more specific concept. The set of proper extensions forms a **partition** of O^k, and the set of extensions forms a hierarchical clustering of O^k.

Figure 2.4 presents the five concepts of neighbors of Charlotte in the graph context presented in Figure 2.2 (a). On the right, the extensions are presented as a Venn diagram, and on the left the intensions are expressed in plain English for simplicity. The first concept has for intension the whole graph centered on Charlotte and has only Charlotte in its extension. Then there are two larger concepts, one describing the children of Kate and William (Charlotte and George), and another one describing the women. Then there is an even larger concept describing the people having a father, a mother, and a sibling (Harry, William, Charlotte, and George). Finally, there is the top concept, having an empty intension and O as extension.

2.3.2 Algorithms

This section describes an efficient and anytime algorithm for the computation of concepts of neighbors, published in [Fer17a; Fer18]. The algorithm inputs are a graph con-

text $K = (O, A, I)$, and a k-tuple of objects $\overline{u} \in O^k$, called the *query instance*. The algorithm output is the collection of concepts of neighbors $C\text{-}N(\overline{u})$, with for each conceptual distance $\delta \in C\text{-}N(\overline{u})$ the concept intension $\delta.int$, the concept extension $\delta.ext$, and the proper extension $\delta.proper$.

This section describes the partitioning algorithm that explores the generalization space top-down to compute concepts of neighbors. Then an essential optimization in the computation of sets of answers is presented, by introducing the *lazy join* algorithm.

Iterative Partitioning of Instances into Concepts of Neighbors

This algorithm relies on the notion of *pre-concept of neighbors*, which is intuitively similar to a concept, except that the PGP is not necessarily the most specific query for those answers. The definition of pre-concept relies itself on the notion of match-set.

Definition 18 *Let $K = (O, A, I)$ be a graph context, and $\mathcal{V}$ an infinite set of variables. A **match-set** is a pair $M = (\overline{x}, R) \in \mathcal{V}^k \times \mathcal{P}(O^k)$, for some arity k. It defines a set of mappings from the k variables in $\overline{x}$ to objects of the context: $\overline{x} = dom(M)$ is called the **domain** of the match-set, and $R = rel(M)$ is called the **relation** of the match-set. Match-sets are equipped with two operations from relational algebra [Cod70]:*

- *the **projection** $\pi_{\overline{y}}\ M$ of a match-set on a sub-tuple $\overline{y}$ of the match-set variables;*
- *the **(natural) join** $M_1 \bowtie M_2$ of two match-sets.*

Definition 19 *Let K be a graph concept, $\overline{u}$ be the query instance with arity k, and O^k be the set of candidate instances. A* partition *of the set of candidate instances is a collection $\{C_l\}_l$ of* pre-concepts (of neighbors)*, where each pre-concept is a structure $C_l = (Q_l, R_l, V_l, M_l, H_l)$, s.t.:*

- *$Q_l = [\overline{u} \leftarrow P_l]$ is a k-PGP that is a generalization of $Q(\overline{u})$, with $\overline{x}_l$ being the tuple of all variables occurring in $\overline{u}$ or in P_l;*
- *$R_l = ans(Q_l) \cap O^k$ is the set of answers of Q_l in O^k;*
- *$V_l \subseteq R_l$ is a subset of answers such that the collection $\{V_l\}_l$ forms a partition of O^k;*
- *$M_l = (\overline{x}_l, ans((\overline{x}_l, P_l)))$ is the* match-set *containing all matchings of the pattern P_l on the graph context;*
- *$I_l \subseteq I$ is a set of incidences from the description of the query instance that remain available for specializing Q_l.*

Initially, there is a single pre-concept, the *initial pre-concept* that uses the empty PGP and that contains all candidate instances.

Definition 20 *The* initial pre-concept *is the pre-concept C_{init} s.t.:*

- *$Q_{init} = [\overline{u} \leftarrow \emptyset]$ is the empty PGP ($\overline{x}_{init} = \overline{u}$);*
- *$R_{init} = ans(Q_{init}) \cap O^k = O^k$ is the set of all candidate instances;*
- *$V_{init} = O^k$ is the set of all candidate instances;*
- *$M_{init} = (\overline{u}, O^k)$;*
- *$I_{init} = I$ is the set of all incidences from the graph context.*

Each pre-concept C_l s.t. $I_l \neq \emptyset$ may be split in two pre-concepts C_i and C_j by using an incidence $(\overline{w}, a) \in I_l$ to discriminate among instances V_l those that match the incidence from those that do not.

Definition 21 *The specialization of a pre-concept C_l by an incidence $(\overline{w}, a) \in I_l$ leads to two new pre-concepts C_i and C_j that replace C_l in the partition, and that are defined as follows (the definitions of R and M follow from the definition of Q):*

$$\begin{aligned} Q_i &= [\overline{u} \leftarrow P_l \cup \{(\overline{w}, a)\}] & Q_j &= Q_l \\ V_i &= V_l \cap R_i & V_j &= V_l \setminus R_i = V_l \setminus V_i \\ I_i &= I_l \setminus \{(\overline{w}, a)\} & I_j &= I_l \setminus \{(\overline{w}, a)\} \end{aligned}$$

According to the above definition, the cost of specializing a pre-concept looks very small. It amounts to add an incidence to a PGP, to perform basic set operations on sets of instances, and to remove an element from the set of incidences. However, the computation of V_i and V_j requires the set of answers $R_i = ans(Q_i)$ of the specialized PGP. This computation can be made incremental by relying on the match-set of pre-concepts. The match-set M of a graph pattern P is equal to the join of the match-sets of all incidences $(\overline{w}, a) \in P$:

$$\begin{aligned} M = & \bowtie_{(\overline{w},a) \in P} M_{(\overline{w},a)} \\ \text{where } M_{(\overline{w},a)} = & (\overline{w}, ans([\overline{w} \leftarrow a(\overline{w})])) \end{aligned}$$

As the join operator is associative and commutative, the match-set of the specialized PGP can be computed incrementally from the parent PGP.

$$M_i = M_l \bowtie M_{(\overline{w},a)}$$

Algorithm 1 $Partition(K, \overline{u})$

Require: A graph context $K = (O, A, I)$
Require: An arity $k > 0$ and a query instance $\overline{u} \in O^k$
Require: An optional timeout
Ensure: A collection of pre-concepts $\mathcal{C}$ partitioning V w.r.t conceptual distance to $\overline{u}$

$\mathcal{C} \leftarrow \{C_{init}\}$
while no timeout and there is a pre-concept $C_l \in \mathcal{C}$ such that $I_l \neq \emptyset$ **do**
  pick some $(\overline{w}, a) \in I_l$ that is connected to $\overline{u}$ in Q_l
  $Q_i \leftarrow [\overline{u} \leftarrow P_l \cup \{(\overline{w}, a)\}]$
  $M_i \leftarrow M_l \bowtie M_{(\overline{w},a)}$
  $R_i \leftarrow rel(\pi_{\overline{u}}\ M_i)$
  $V_i \leftarrow V_l \cap R_i$
  $V_j \leftarrow V_l \setminus V_i$
  $I_{ij} = I_l \setminus \{(\overline{w}, a)\}$
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_l\}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_i\}$, **if** $V_i \neq \emptyset$, **where** $C_i = (Q_i, R_i, V_i, M_i, I_{ij})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_j\}$, **if** $V_j \neq \emptyset$, **where** $C_j = (Q_l, R_l, V_j, M_l, I_{ij})$
end while

Finally, the set of answers is simply the projection of the match-set on the projected variables.

$$R_i = rel(\pi_{\overline{u}}\ M_i)$$

Algorithm 1 details the partitioning algorithm. Given a graph context, a query instance, and a set of candidate instances, it starts with a single pre-concept, the initial pre-concept (Definition 26). It then iteratively applies specialization steps (Definition 21) to pre-concepts in order to refine the partition. The process runs until no specialization is possible, or until a timeout has been attained. This timeout is optional, but it has the advantage to make the algorithm anytime (more on this below). Figure 2.5 shows an execution trace as a binary tree of pre-concepts. It represents the computation of the concepts of neighbors of Charlotte (see Figure 2.4): $\overline{u} = Charlotte$. The initial pre-concept contains the seven persons in the context, from Charles (C) to Charlotte (A). The first specialization uses the incidence $woman(Charlotte)$, separating the women (Diana, Kate, and Charlotte) on the left from the men on the right. Those women remain in the extension of the concept on the right but no more in the proper extent. The concept on the left is further split in two pre-concepts: woman with a parent on the left (A), and woman without a parent on the right (DK). Pre-concept (A) cannot be split further, although incidences can still be added to it. From pre-concept (DK), all remaining incidences lead

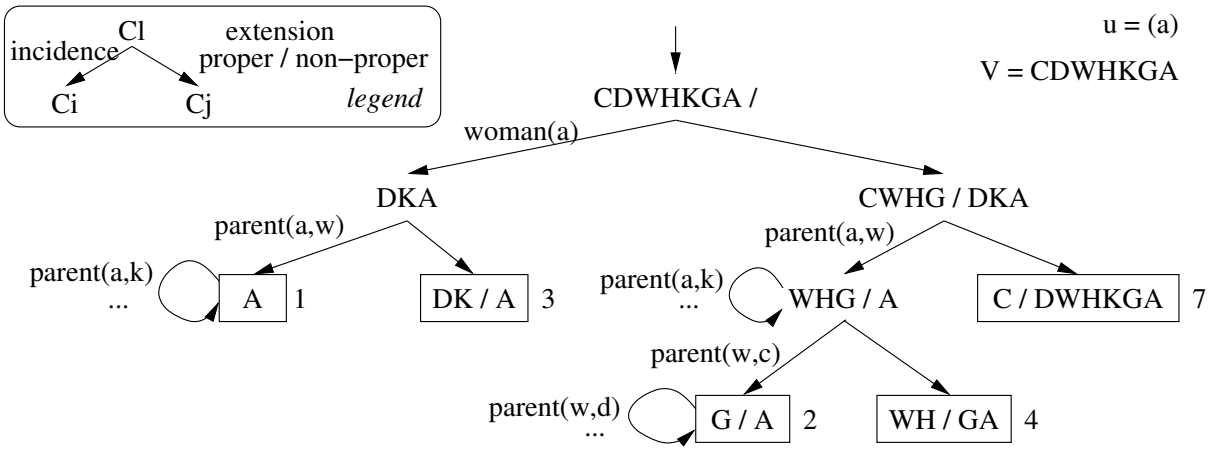

Figure 2.5 – Trace of the partition algorithm applied on the graph context in Figure 2.2 with $\overline{u} = Charlotte$. Pre-concepts are shown via their extension, with the proper extension on the left and the remainder on the right, if any. Objects are abbreviated by their initial, except for Charlotte (A). The same abbreviations in lowercase are used as variables in the incidences used for pre-concept specialization. Boxed pre-concepts at the leaves are the concepts of neighbors, and the number at their right is the extensional numerical distance.

to an empty extension because Diana and Kate have nothing else in common with Charlotte. The boxed pre-concepts at the leaves are the results of the algorithm. Their intent is the set of incidences that label the path from the root to this pre-concept. They coincide with the concepts of neighbors shown in Figure 2.4.

Lazy Join of Match-Sets

The partitioning algorithm of the previous section has good properties w.r.t. the exploration of the search space, but it hides a bottleneck in the computation of match-sets, that grow exponentially in size. It is actually possible to do better because the expected end result is the set of answers $R = \pi_{\overline{u}}\ M$, whose size is bounded by the number of candidate instances $|O|^k$.

We here describe a compact representation of a match-set M, called a *match-tree*, that is made of several local joins instead of the global join. It supports the incremental computation of match-sets assumed by the partitioning algorithm, and it performs joins in a local and lazy way to keep the representation as compact as possible.

A match-set M_l results from the set of incidences P_l. A match-tree is based on a tree structure over those incidences.

Algorithm 2 *LazyJoin*$(T, i, i^*, D^*, M^*, \Delta^*)$

Require: a match-tree T, a current incidence i in T labeled with (D, M, Δ),
and a new incidence i^* labeled with (D^*, M^*, Δ^*)
Ensure: two sets of variables Δ^+, Δ^-
1: $\Delta^+ \leftarrow \emptyset$; $\Delta^- \leftarrow \emptyset$
2: **for all** $i_c \in children(i)$, labeled with (D_c, M_c, Δ_c) **do**
3: $\Delta_c^+, \Delta_c^- \leftarrow LazyJoin(T, i_c, i^*, D^*, M^*, \Delta^*)$ {recursive call on each child node}
4: $\Delta^+ \leftarrow \Delta^+ \cup \Delta_c^+$; $\Delta^- \leftarrow \Delta^- \cup \Delta_c^-$
5: $M \leftarrow M \bowtie \pi_{\Delta_c} M_c$, if Δ_c or M_c was modified {update of local join}
6: **end for**
7: **if** $D \cap \Delta^* \neq \emptyset$ **then** {if this node defines a variable of the new element}
8: **if** i^* not yet inserted in T **then** {insert new node, if not yet inserted}
9: $\Delta^- \leftarrow \Delta^- \cup (\Delta^* \setminus D)$; $M \leftarrow M \bowtie \pi_{\Delta^*} M^*$; $parent(i^*) \leftarrow i$
10: **else**
11: $\Delta^+ \leftarrow \Delta^+ \cup (\Delta^* \cap D)$
12: **end if**
13: **end if**
14: $\Delta^+ \leftarrow \Delta^+ \setminus \Delta^-$
15: $\Delta^- \leftarrow \Delta^- \setminus \Delta^+$
16: $\Delta \leftarrow \Delta \cup \Delta^+ \cup \Delta^-$ {update Δ}
17: **return** Δ^+, Δ^-

Definition 22 *A* match-tree *is a rooted n-ary tree T where each node is an incidence $i = (\overline{w}, a)$ and is labeled by a tuple (D, M, Δ) where*[3]*:*

— *$D \subseteq \overline{w}$ is a subset of the variables used by incidence i;*

— *M is the* local match-set *s.t. $\overline{w} \subseteq dom(M)$;*

— *$\Delta \subseteq dom(M)$ is the subdomain that is useful to the node's ancestors.*

The *initial match-tree* T_{init} is used in the initial pre-concept in place of the initial match-set. It has a single root node that is a pseudo-incidence $i = \top(\overline{u})$ and that is labeled with $(\overline{u}, M_{init}, \overline{u})$.

The line $M_i \leftarrow M_l \bowtie M_{(\overline{w},a)}$ doing the incremental join in Algorithm 1 is replaced by

$$T_i \leftarrow LazyJoin(T_l, \top(\overline{u}), (\overline{w}, a), D^*, M^*, \Delta^*)$$

where *LazyJoin* is defined by Algorithm 2. It is based on a recursive traversal of the match-tree (lines 2-3) starting at the root $\top(\overline{u})$, inserting the new incidence $i^* = (\overline{w}, a)$ at an appropriate place (line 9), and updating local match-sets accordingly (line 5). The

3. By abuse of notation, we allow tuples of variables to be used as sets of variables.

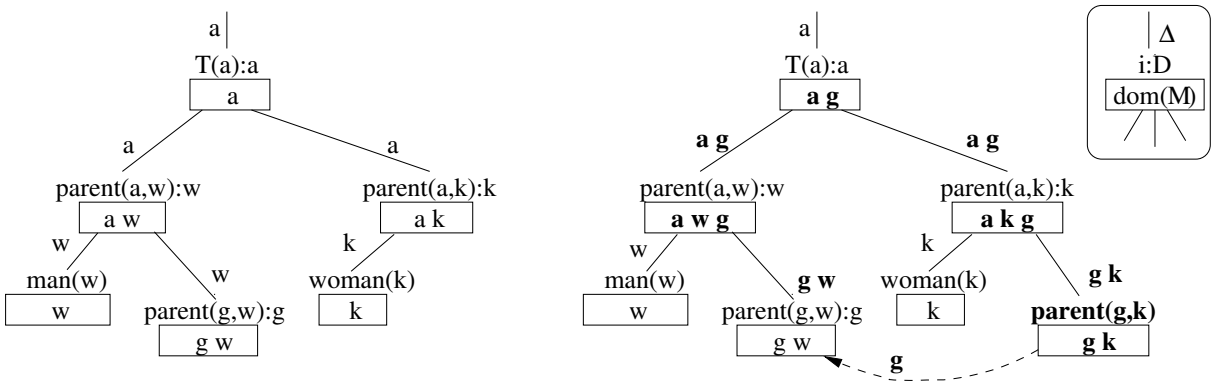

Figure 2.6 – Match-tree before and after lazy join with incidence $i^* = parent(g, k)$.

new incidence is labeled as follows, where for recall $\overline{x}_l$ is the tuple of all variables in the current pattern P_l:

$$D^* = \overline{w} \setminus \overline{x}_l, \quad \Delta^* = \overline{w} \cap \overline{x}_l, \quad M^* M_{(\overline{w},a)}.$$

Considering the variables Δ^* of the new incidence that are already in the match-set, the new incidence i^* is inserted as a child of the first visited node whose defined variables D contains at least one variable in Δ^* (lines 7-9). The common variables between D and Δ^* provide a connection between the current pattern and the new incidence. If some variables in Δ^* are not defined at the insertion node, they are returned and propagated through Δ^- as missing variables for join (line 4, 9, 17), and they are propagated from the incidences that define them through Δ^+ as available variables for join (line 4, 11, 17). Those missing and available variables are added to the Δ^+/Δ^- of incidences upward (line 16), except when they meet each other (lines 14-15).

Finally, the set of answers R corresponding to a match-tree T is $rel(\pi_{\overline{u}}\ M_\top)$, where $M_\top$ is the local match-set of the root.

Figure 2.6 shows the effect of the lazy join of the new incidence $i^* = parent(g, k)$ at the pre-concept with proper extension WHG in Figure 2.5. The input match-tree is on the left, and the output match-tree is on the right. The input match-tree corresponds to the PGP

$$[a \leftarrow man(w), woman(k), parent(a, w), parent(a, k), parent(g, w)],$$

and it is hence the result of 5 successive lazy joins, each introducing an incidence. Changes

(shown in bold on the right side) are propagated from the two incidences that define g and k (see D), and the new leaf is inserted under one of the two nodes as a child (here, under the node defining k). The insertion of other incidences, which led to the match-tree on the left, change only one path in the match-tree because they do not introduce a cycle. In Algorithm 2, the computation of Δ^+, Δ^- is used to correctly handle cycles. They are sets of variables of the new incidence i^* that are not in the match-set of its parent (g in the example), and hence need to be joined with distant nodes in the match-tree. Δ^- propagates up the tree branch of the parent (incidence $parent(a, k)$ in Figure 2.6), and Δ^+ propagates up the tree branch of distant incidences (incidence $parent(g, w)$). When they meet at their common ancestor (incidence $\top(a)$ here), the distant join can be done, and their propagation stops.

2.4 Conclusion

This chapter resumes the theoretical foundations of FCA, as well as the main definitions of Graph-FCA. In addition, it sums up the notion of Concepts of Neighbors and the linked notions, in the framework of Graph-FCA. Finally, it presents the algorithmic framework developed for the computation of concepts of neighbors. The elements presented in this chapter are used as a theoretical base for the work presented in Chapter 4, and an application of this is developed in Chapter 5.

Part II

Contributions

Chapter 3

A Workflow Proposal for User-Centric Explainable Artificial Intelligence

As presented before, with the rapid expansion of artificial intelligence appeared a need for human centered systems that guarantee a finer control of the user on the outcome, in order to obtain systems having both a high level of automation and a high level of human control. Those are designated in [Shn20] as *Reliable, Safe and Trustworthy* (RST) systems.

The work presented in this chapter introduces a workflow model for a system ensuring good level of both automation and human control, on the specific task of knowledge graph construction from texts. This workflow, based on progressive automation by imitation, aims to automatize most of the user's action whilst ensuring a maximal control on the produced graph, being therefore an RST system. Such a design for this task is particularly adapted because, even if knowledge graph construction is not necessarily *per se* a critical task, several elements make human control central for having a usable resulting knowledge graph. First, by the complex and sometime ambiguous aspects of the semantics of natural language, it is today illusory to have a fully automated system with high performance, and human curation is fastidious. In addition, for a given text, several knowledge graphs could be built, depending on the vocabulary and the knowledge the user is interested in, therefore user feedback is necessary to obtain the desired graph as output.

Theoretically, the proposed workflow could be used for any use case of knowledge graph construction from text, with any vocabulary and any scope, from the translation of an encyclopedia into a massive public knowledge graph to the archiving of knowledge contained in domain-specific text in a personal small RDF database. However, several design choices make this workflow fit better on domain-specific texts composed of short, factual sentences. This is caused by the iterative aspect of this workflow: the larger the

vocabulary is and the more diverse the formulations are, the longer the progressive automation will take and therefore the smaller is the benefit from this approach. Anyway, many use-cases can be imagined even with this constraint. For example, this system could be used by a medical professional for the processing and archiving of facts contained in medical evaluation reports, or by a lawyer for processing textual case reports. Once the desired knowledge graph obtained, it could be explored and queried thanks to user-friendly knowledge graph exploration tools such as Sparklis [Fer17b].

This chapter details a full presentation of this workflow. First, Section 3.1 presents a detailed overview of the workflow, its global structure, its use cases and its overall functioning. Then, the different units of this workflow are detailed: Section 3.2 details the design of the pre-processing unit; Section 3.3 presents the interactive unit; Section 3.4 presents the automated unit. Finally, Section 3.5 concludes this chapter.

3.1 Global Overview

The main concept behind this workflow of user-centric A.I. is progressive automation. It comes from the idea that the best way to have a result close to what would the user do by hand is to imitate its actions. From this idea, a progressive automation process emerges. Initially, the system behaves as a manual edition system with a naive suggestion module, letting the user in full control of the output. Then, based on the user's previous actions, the suggestion module begins to suggest more accurate actions to the user, fastening the process, as well as explanations for each of those actions. As the interaction history grows, the suggestions, as well as the provided explanations, become more and more accurate. Finally, the explanations validated by the user are used as trustworthy rules, used to fully automate a growing proportion of actions. In an advanced state, this system should be almost fully automated, only the ambiguous or novel cases should need actions from the user.

As it is shown on Figure 3.1, this workflow can be divided in three units. The first one is the *pre-processing unit*. This unit takes text as input and produces an intermediate modeling of it. This modeling is built from two pipelines. The first pipeline, consisting of a set of NLP tools and a modeling module, produces a syntactic and semantic graph modeling of text. This modeling is completed by the second pipeline, consisting of a relation detection module, that detects which pairs of entities in the text are likely to be linked by a semantic relation – and therefore could be transformed into triples. This

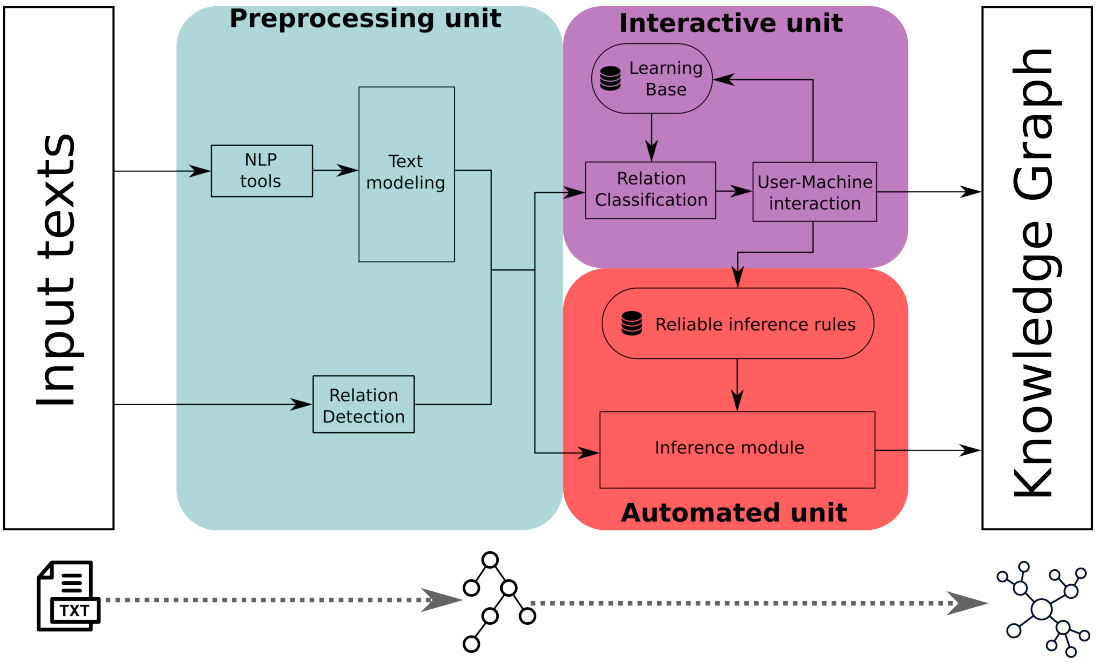

Figure 3.1 – Workflow overview

intermediate modeling is then used by the two other parallel units. One of them, the *interactive unit*, is composed of an explainable AI for relation classification. This module aims to produce triples that are candidates to add to the knowledge base, as well as an explanation for each of those predictions. Both the candidates and the explanations are then submitted to the user through a user-machine interaction module. Validated triples are used to populate the knowledge graph, and validated explanations are transformed into inference rules. Those rules are then used by the last unit, the *automated unit*, that uses them to directly generate reliable triples from the text modeling, using an inference module.

It can be pointed out that this workflow contains two different internal bases, in addition to the input set of texts and the output knowledge base. The first one, part of the interactive unit, is a learning base composed of annotated examples for relation classification, used by the relation classification module. The second one, part of the automated unit, is composed of inference rules that come from explanations produced by the relation classification module and that are used by the inference to produce triples. Based on the initial state of these bases, several cases can appear. The basic case is the case where the two bases are initially empty. In this case, the system is fully independent of any preexisting vocabulary, and works incrementally: initially, as the learning base

is empty, the relation extraction module is not able to produce predictions. Therefore, the system acts as an interface for guided edition of RDF graphs, as the user creates manually the first triples and populates the learning base. As this base gets bigger, the relation classification module becomes able to produce more and more reliable predictions to help the user. Finally, as the user validates some predicted triples and explanations, inferences rules are produced and come to populate the base used by the automated unit, and therefore automating progressively the whole process. From this basic case, various cases can be derived: if a pre-existing vocabulary can be used, the learning base can be initialized with generic examples containing this vocabulary; if it exists an annotated base of examples matching the desired vocabulary, it can be used as an initial learning base to bootstrap the suggestion module; if another instance of the system has already been used on a compatible corpus, its learning base and/or rule base can be both used to bootstrap the system.

3.2 Preprocessing Unit

As presented in Figure 3.1, the main objective of this unit is to produce, from the input texts, a comprehensive graph modeling representing the necessary knowledge to transform the texts into a knowledge graph, as well as listing the pairs of entities that could be linked by a relation. In this section, a global overview of this unit is first presented. Then, an example of semantic and syntactic modeling, based on a syntactic-level graph representation of sentences, is proposed and discussed.

3.2.1 Unit Overview

As evoked before, the main role of this subsystem is to transform the input natural language text into a modeling that can be used by the two other units. This modeling has to have two main properties. First, it should represent enough semantic and/or syntactic knowledge to be able to identify the facts to transform into RDF triples. In addition, this unit has to detect and specify which pairs of named entities could be linked by a relation that could be translated into a triple. This is needed for two separate reasons. First, this allows the system to suggest to the user couples of entities to transform into triples before the learning base has enough data for the relation classification module to be effective. Second, the module used for relation classification is not able to perform

relation detection, and therefore the couples that can be linked by a relation should be annotated as so. This point is discussed later in Chapter 5.

As a consequence, this pre-processing unit can be decomposed in three modules. The first one is a NLP pipeline able to extract linguistic and semantic data from text (*e.g.*, named entities, part-of-speech tags, sets of synonyms, semantic representation, depending on the modeling choice). The data produced by this module is then used by a second module to produce a graph modeling of the text. In parallel, a relation detection module is used to detect which pairs of named entities of the text could be linked by a relation that can be translated in triple. It can be pointed out that in this pre-processing unit, explainability is not needed: we look for the most efficient tools, and we rely on the interactivity of the other parts of the system to correct possible omissions and errors.

The choice to use a graph modeling for this intermediate representation is motivated by the symbolic, reasonable and interpretable aspects of graphs. As we seek for an interactive and reliable system, the decisions must be interpretable by the user. Using numerical representations (such as sentence embeddings) would be a prejudice, as those representations are hardly understandable by a non-expert user. This is why we opted for a more readable, symbolic representation, and more precisely for a representation as graphs, because of the comprehensiveness and the reasonable aspect of graph data.

3.2.2 A Proposal for Modeling Texts as Graphs

This section presents a proposal for modeling texts as graph. This modeling aims to model a text sentence by sentence, grouping syntactic and semantic knowledge in a graph. For practical reasons, this modeling is made using the RDF model. However, this graph is not a proper knowledge graph, but an intermediate modeling. The proposed modeling does not include the relation detection module described before, this aspect is discussed in Chapter 5. This section first presents the performed extraction of linguistic knowledge, then presents the processing made on this knowledge, then finally presents the modeling itself.

Linguistic Knowledge Extraction

The first step for modeling the semantic and syntactic aspect of a sentence is to extract this semantic and syntactic knowledge. For the proposed modeling, the syntactic aspect includes extracting tokens, part-of-speech tags, lemmas, dependency relations and named

ID	token	lemma	POS	NER	head	deprel
1	The	the	DT	-	2	det
2	University	University	NNP	ORG.	6	nsubj
3	of	of	IN	ORG.	4	case
4	Rennes	Rennes	NNP	ORG.	2	nmod
5	is	be	VBZ	-	6	cop
6	French	french	JJ	NATIO.	-	ROOT
7	.	.	.	-	6	punct

Table 3.1 – Example of a processed sentence.

entities, and the semantic aspect includes the set of synonyms of each verb and noun of the text, and their hypernym hierarchies. Other semantic aspects could be used to enrich this modeling, such as using the Abstract Meaning Representation [Ban+13] of the sentences.

NLP Pipeline For the syntactic aspects, a standard NLP pipeline is built in order to extract the needed knowledge. This pipeline contains a tokenizer, a part-of-speech (POS) tagger, a lemmatizer, a dependency parser and a named entity recognizer. In our case, Stanford CoreNLP [Man+14] is used to create this pipeline.

Table 3.1 shows the result of the processing of the sentence *"The University of Rennes is French"* by such a pipeline. For example, it shows that the 4th token is *Rennes*, has for lemma *Rennes* and for POS tag *NNP* (a proper noun). It is part of a named entity of type *ORGANIZATION*. It has for parent in the dependency tree the 2nd token, and it is linked to this token via relation *nmod* (linking a nominal modifier to its parent noun).

Lexical Enrichment This data is completed by using a hierarchical lexical database such as WordNet [Mil98] to enrich it semantically. Once a set of text has been processed by the NLP pipeline, we extract from WordNet the sets of synonyms (called *synsets*) of each lemma of noun or verb. Then for each of these synsets, we retrieve recursively its hypernym hierarchy, *i.e.*, the synsets that generalize its meaning. For example, Table 3.2 presents the hypernym hierarchy of two noun lemmas (*school* and *university*). It can be seen that all those hierarchies overlap on the most general synset (*entity*) and that a few other synsets have an overlapping hypernym hierarchy. This shows the semantics similarity of both those lemmas.

Lemma	Synset	Hypernyms
school	school (institution)	educational institution, organization, social group, group, abstraction, entity
	school (process)	education, learning, basic cognitive process, cognitive process, cognition, psychological feature, abstraction, entity
	school (body)	body, social group, group, abstraction, entity
	...	...
university	university (body)	body, social group, group, abstraction, entity
	university (establishment)	establishment, structure, artifact, whole, object, physical entity, entity
	university (institution)	educational institution, organization, social group, group, abstraction, entity

Table 3.2 – Example of hypernym hierarchy

Linguistic Knowledge Processing

Once this knowledge has been retrieved, post-processing is made in order to remove useless data and redundancies.

Removing punctuation. It has been observed that in the dependency trees, the punctuation tokens (dots, question marks, commas, parenthesis...) are systematically leaves. Additionally, those tokens bear little, if any, semantic knowledge that is not present in the syntactic structure itself. Therefore, removing them does not imply transformations on the syntactic tree and allows reducing easily the size of the modeling without losing much information. For example, in Table 3.1, token 7, which represents the dot token, can be removed from the data without losing significant knowledge.

Compound named entities. A named entity can overlap several contiguous tokens, for instance *University of Rennes* overlaps tokens 2-4 in the example. However, a named entity is a semantic unit: it holds its own meaning, which can be very different from the meaning of its individual tokens. Therefore, manipulating a named entity as a succession of tokens can cause an important loss of semantics. For example, a sentence evoking the *"United States of America"* and another one about a *"united family"* cannot be considered as referring to the same meaning in its use of the token *"united"*. Except when there is a parse error, an entity forms itself a tree in the dependency tree. We call it a *tree factor* (*i.e.* a subtree from which have been removed subtrees), by analogy with the definition of

ID	token	lemma	POS	NER	head	deprel
1	The	the	DT	-	2	det
2	University of Rennes	University of Rennes	NNP	ORG.	4	nsubj
3	is	be	VBZ	-	6	cop
4	French	french	JJ	NATIO.	-	ROOT

Table 3.3 – Example of sentence after post-processing.

a string factor (*i.e.*, a string suffix from which has been removed a suffix). The proposed solution is to collapse the tree factor into its root. Then the sentence retains a valid syntactic and semantic structure (no dangling link, for instance). Moreover, the linguistic properties of the root of a tree factor are most of the time a fair summary of the linguistic property of the whole tree factor. For example, in the sentence presented in Table 3.1, the named entity *"University of Rennes"* is collapsed into token 2, and tokens 3 and 4 disappear. The expression *"University of Rennes"* can indeed be seen as a proper noun (POS tag *NNP*). In case of a parse error, the named entity is collapsed to the last token as a fallback.

Table 3.3 presents the result of this processing on the data presented in Table 3.1. This shows that, even with the punctuation removed and the named entity collapsed to one token, the data stays syntactically coherent, while keeping all the significant knowledge.

Graph Modeling

In order to model a set of processed sentences as one RDF graph, each token is represented by an RDF node (e.g. in Figure 3.2, `id:1_2` for the 2nd token of the 1st sentence), and each dependency link is represented by an RDF edge. The lemmas, POS tags, and named entity types of a token are represented by RDF types on the corresponding node (see discussion below). Figure 3.2 gives the RDF representation for the example in Table 3.3. The 2nd token is modeled by node *id:1_2*, which has as types lemma *University of Rennes*, POS tag *NNP*, and named entity type *organization*, and is linked to node *id:1_6* by relation *nsubj*.

Representation of lemmas and POS tags as RDF types. In this modeling, we use relation *rdf:type* instead of defining specific relations for linking a node to its lemma or POS tag. This choice has been made because most of RDF applications process *rdf:type* triples not as binary relations, but as unary labels, labelling the typed RDF entity. Therefore,

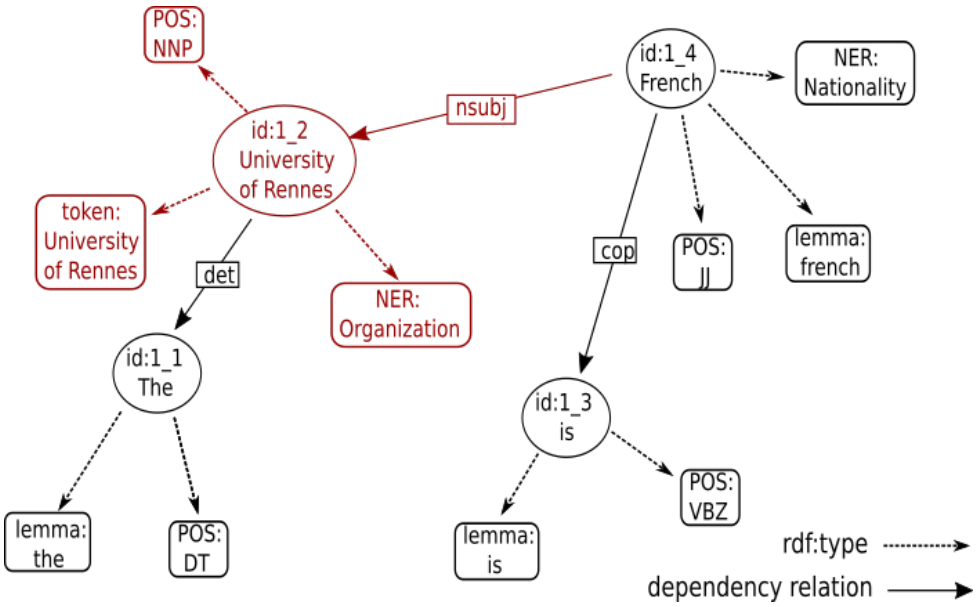

Figure 3.2 – Example of a sentence modeled as an RDF graph.

using this relation type has several advantages. First, it keeps the different sentences unconnected: without this mechanism, two sentences sharing a POS tag or a lemma would be connected, what could cause problems while exploring this modeling. Second, it allows for those labels to be treated atomically: without it, patterns presenting for example an entity having an unspecified POS tag could appear, which is useless information. Finally, as presented below, by the fact that the RDF specification allows for ontology presenting type hierarchy, this permits relaxation over the lemmas and POS tags.

Lemmatization of named entities. If, in the general case, the lemma of a token is a good representation of its semantics, it does not stand in the case of named entities. For instance, the concatenation of lemmas *unite state of America* is not a good representation of the named entity *United States of America*. Then, for the entity representing a named entity, instead of giving the lemma as an RDF type in the modeling, the list of tokens itself is used.

Type Hierarchies As evoked before, RDF graph can be enriched with inferred types and edges by declaring domain knowledge. The most common form of domain knowledge is hierarchies of types, based on the RDFS property `rdfs:subClassOf`. The inference rule is that: if node X has type A and A is a subclass of B, then X also has type B. We use

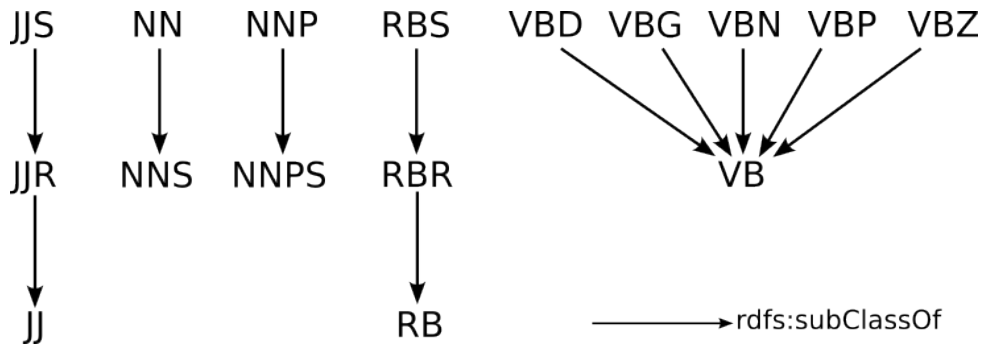

Figure 3.3 – POS tag hierarchy

several type hierarchies to enrich this modeling.

First, the set of POS tags [Tou+03] is fine-grained enough to create a hierarchy on a few POS tags: for example, the gerund of a verb (POS-tag *VBG*) is a subclass of verb (POS tag *VB*). In total, we have 11 `rdfs:subClassOf` triples added to the modeling's ontology, as presented in Figure 3.3.

Second, the previously retrieved lemma and synset hierarchy is added to the modeling. Each synset of a lemma is considered as a superclass of the lemma, and each hypernym of a given synset is considered as a superclass of this synset. Figure 3.4 shows a fragment of this lemma hierarchy for the lemmas *university*, *school*, *religion* and *faith*. It can be seen that those three lemmas can be relaxed into *institution*, but only *university* and *school* can be relaxed in *educational institution*. The lemma hierarchy thus increases the chance to find similarities between sentences using words that have different lemmas but close meanings.

Modeling Discussion

This modeling represent text with token-level entities linked by syntactic relation. This choice relies on the intuition that the semantics of a sentence reside on both the individual semantics of the tokens and the syntactic structure of the sentence. In the modeling, the semantics of individual tokens is represented by the lemmas and the hierarchy created over their synsets, and the syntactic structure is represented by the part-of-speech tags and dependency relations. Therefore, this modeling necessarily relies on the quality of the NLP tools used to extract this linguistic knowledge, each associated NLP task being a subject of research on its own.

The proposed modeling could be enriched by using higher level semantic represen-

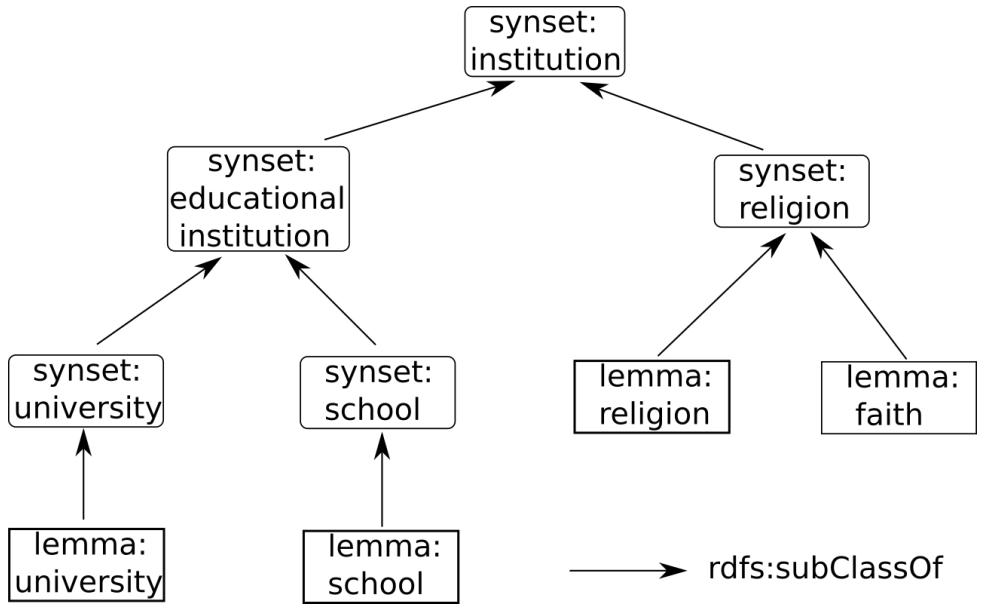

Figure 3.4 – Fragment of a type hierarchy obtained from WordNet

tations of NLP texts, such as *Abstract Meaning Representation* (AMR, see [Ban+13]). However, a trade-off between the comprehensiveness of the modeling and its size have to be found, in order to guarantee low enough execution times for the system to be interactive.

3.3 Interactive Unit

This unit has two main roles. First, it has to contain an explainable module for relation classification, able to reason on the modeling produced by the pre-processing unit. As the learning base of this module evolves, becoming richer through the interactions with the user, this module needs to rely on an *instance-based learning* approach (also called *lazy learning* approach), *i.e.*, a method that, instead of computing a general prediction model from a fixed learning base, computes a local prediction model for each instance to classify. In addition, this relation extraction module needs to produce explanations that simultaneously have to be easily understandable by the user and can be transformed into inference rules to be used by the last unit. Chapter 5 presents a proposition for this module, based on the *Formal Concept Analysis* framework.

The second main role of this unit is to provide a comprehensive interface for the

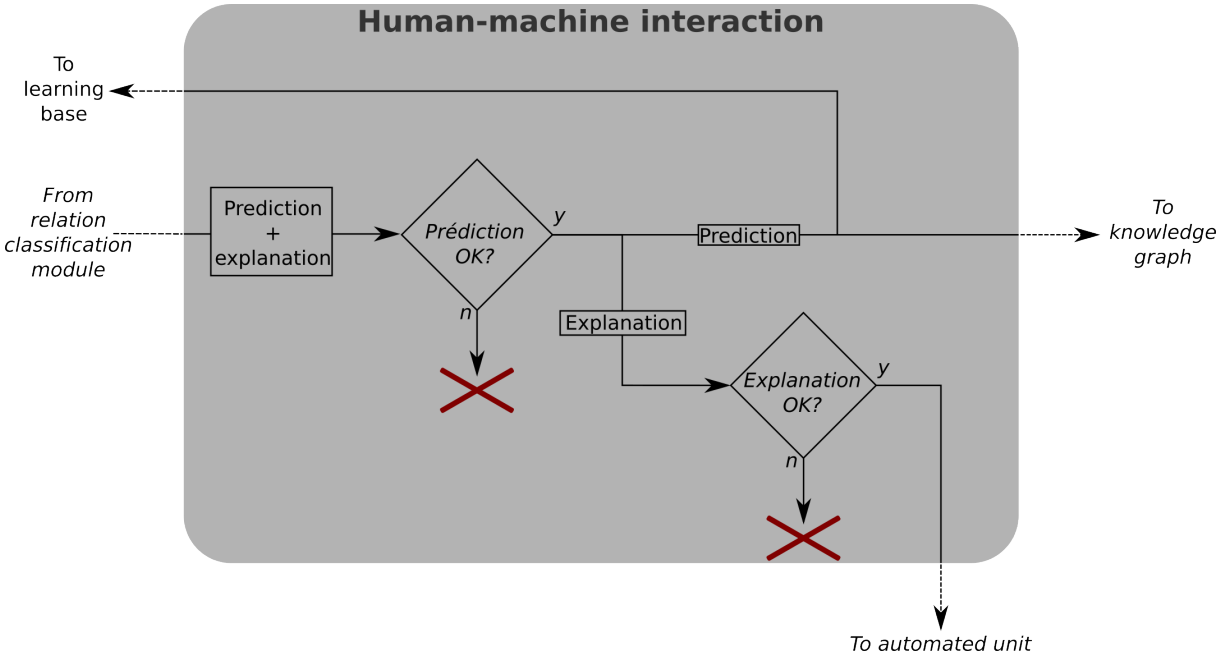

Figure 3.5 – Interaction model for prediction and explanation validation

edition of RDF graph, taking into account the modeling and entity couples created by the pre-processing unit, but also allowing for manual edition and typing of entities and relations, making the user able to correct the omissions and mistakes made by the pre-processing unit. This interface would be used to populate both the RDF graph and the learning base for the relation classification module. This interface should also manage the suggestions and related explanations formulated by the relation classification module, in order to submit them to the user for validation. Figure 3.5 details the interactive process for managing the suggestions: first, the interface suggests to the user a triple to add to the knowledge graph. The user can then accept or reject the suggestion. If the triple is accepted, it is added to the knowledge graph and to the learning base, and the explanation is displayed to the user. If they accept this explanation, it is translated into an inference rule and transmitted to the automated unit.

3.4 Automated Unit

This unit, simpler than the two others, is composed of only two elements: an inference module and its base of inference rules.

This has for consequence that the rules produced, and by extension the explanations returned by the relation classification module, have to be necessarily applicable on the text modeling, directly or after a transformation conserving the semantics. Therefore, those rules have to be of the form $xRy \leftarrow P(x, y)$, with x and y two variables, xRy an RDF triple and $P(x, y)$ a graph pattern that can be queried in the text modeling. In the case where the modeling is, as presented earlier, an RDF graph, $P(x, y)$ could be a SPARQL graph pattern, returning tuples matching a given pattern that has been tagged as trustworthy for predicting the relation R.

3.5 Conclusion

In this chapter is presented a novel workflow model for knowledge graph construction from natural language text. This workflow, designed to provide both a high level of human control and a high level of automation, is based on a progressive automation of the user's actions. The system rely on an intermediate modeling of the input natural language texts as graph. On this modeling is then used a suggestion module that, coupled to a user-machine interaction module, suggest triples to add to the produced knowledge graph, each suggestion being coupled with an explanation. The validated triples enrich the learning base of the suggestion module (that therefore continuously improves its suggestions) while the validated explanations are used by an automated inference module, that automatically produce triples from the intermediate modeling.

Several aspects of this contribution could be subject to future works. First, if this workflow model has been conceived for the construction of knowledge graphs from text, the concept and structure behind this workflow could be adapted for automating other tasks demanding a high level of human control on the result. For example, by replacing the text modeling by a graph representation of structured data and by replacing the relation extraction module by a module producing JSON data, a similar workflow could be conceived for user-centric translation, refactor and selection of XML documents into JSON data. Second, concerning the intermediate modeling, some enrichment using semantic representations of texts could be tested and evaluated, as well as the impact of the chosen tools used for the extraction of linguistic knowledge. Finally, concerning the automated unit, reliability could be enhanced by associating confidence values to rules, confidence that could vary during the interactions with the user. Then a threshold value could be defined: over this value, rules are applied automatically while under this value,

user confirmation is required. Further work would be needed to determine the nature of this confidence, the form that takes the interactions with the user, the value of this threshold and the pertinence of such a system.

Chapter 4

Concepts of Neighbors: Formalization and Application to RDF Graphs

With the domination of deep learning techniques – which is due to their impressive performances – in the machine learning fields, symbolic machine learning has been on the back burner, despite the interesting properties of some of these approach, such as explainability, interpretability, incementality and so on. However, issues around explainability and interpretability has come to the fore in these recent years, mainly with questioning around critical applications. In [Rud19], the author advocates for the development and use of interpretable models – *i.e.*, models for which the decision-making process can be understood by the user – over post-hoc explanations of black-box models. Most symbolic approaches, by being closer to human reasoning than statistical approaches, are suitable for having this property of interpretability. The different properties are also central in order to conceive user-centered applications, as evoked in the previous chapter.

Concepts of Neighbors, as presented in Chapter 1, is a pattern mining and instance-based machine learning approach for symbolic data. This approach relies on *Formal Concept Analysis* (FCA, [GW99]), a mathematical framework for reasoning on symbolic data, and was initially developed for graph data. This chapter, presenting works both published in [ACF22b] and in [ACF24], regroups our contributions to this approach made in the course of this thesis.

This chapter is structured as follows. Section 4.1 presents a new formalism of Concepts of Neighbors in the classical FCA framework, and its extension to other FCA-based frameworks. In Section 4.2 the tractability issues in the Graph-FCA case, caused by the existence of n-ary concepts, are addressed. Section 4.3 presents works made on using Concepts of Neighbors on RDF graphs, through the relation between Graph-FCA and RDF graphs, the handling of RDF schemas by the Concepts of Neighbors algorithmic

	man	woman	adult	kid	married
Charles	×		×		×
Charlotte		×		×	
Diana		×	×		×
George	×			×	
Harry	×		×		×
Kate		×	×		×
William	×		×		×

Table 4.1 – Example of a formal context where the set of objects is an excerpt of the British royal family.

framework, and the introduction of CONNOR, a Java implementation of the Concepts of Neighbors on RDF graphs.

4.1 Formalization of Concepts of Neighbors on FCA

Concepts of neighbors define a distance/similarity scheme in terms of FCA concepts, in a literal way because concepts are used directly as a measure of the distance/similarity between two objects. This is in contrast with the usual definitions of distance or similarity that use numerical values, even when applied to discrete structures. Concepts of neighbors also offer a local instance-based view of concepts – with FCA objects playing the role of instances – compared to the global view of concept lattices that is common with FCA.

In this section, we first define the notion of concepts of neighbors on standard FCA in order to explain the key ideas on a simple and well known formalism. We then show how this notion can be extended to FCA extensions, including the Graph-FCA extension. This formalization is a generalization of the initial formalization of concepts of neighbors, which is presented in Chapter 2.

4.1.1 The FCA Case

Table 4.1 presents an FCA formal context as defined in Chapter 2, and Figure 4.1 represents the concept lattice of this context. This example is used in the following of this section.

We first define the *conceptual distance* between two objects as the most specific concept that contains both of them.

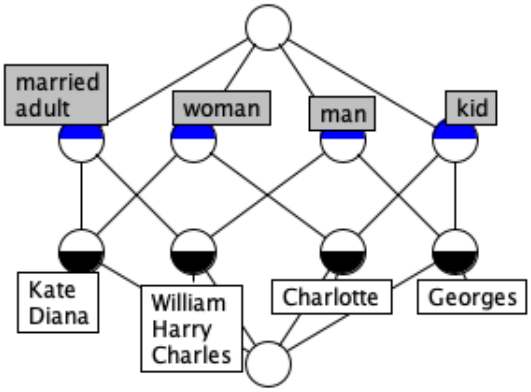

Figure 4.1 – Concept lattice derived from the context in Table 4.1.

Definition 23 *Let $K = (O, A, I)$ be a formal context, and $u, v \in O$ be two objects in this context. The* ***conceptual distance*** *between objects u and v is the concept $\delta(u, v) = (X, Y)$ s.t. $Y = prop(\{u, v\}) = I(u) \cap I(v)$, and $X = inst(Y)$.*

In other terms, the conceptual distance between two objects $\delta(u, v)$ is the most specific concept having both u and v in its extension. Or, from another point of view, this is the supremum of the most specific concept containing u in its extension and of the most specific concept containing v in its extension.

Intuitively, the more similar the two objects are, the more specific the conceptual distance is. A more specific concept has fewer objects and more attributes. Having more attributes means being more similar. The objects in the concept extent can be seen as in between the two objects, and hence fewer objects means a smaller distance. For example, in the formal context presented in Table 4.1, the conceptual distance between Charles and Charlotte has an empty intension and an extension containing all the objects, while the conceptual distance between Charles and Harry has for intension $\{man, adult, married\}$ and for extension $\{Charles, Harry, William\}$. It can be seen that the conceptual distance between Charles and Harry has three attributes in its intension while the conceptual distance between Charles and Charlotte has none. Therefore, Charles is more similar to Harry than to Charlotte. Reciprocally, the conceptual distance between Charles and Charlotte has seven objects in its extension, while the conceptual distance between Charles

and Harry has only three objects. Therefore, Charles is more distant to Charlotte than to Harry.

The duality between distance and similarity is here embodied in the duality between the extension and intension of a concept. Actually, the conceptual distance between two objects is at the same time their *conceptual similarity*. The above definition satisfies the properties of a distance if we take the partial order $\leq$ on concepts as distance order, and concept supremum $\vee$ as addition. It means that, for all objects $u, v, w \in O$:

1. (positivity) $\delta(u,u) \leq \delta(u,v)$, ($\delta(u,u)$ represents distance zero)
2. (symmetry) $\delta(u,v) = \delta(v,u)$,
3. (triangular inequality) $\delta(u,v) \leq \delta(u,w) \vee \delta(w,v)$.

Because of the ordering being partial, it is common to have objects, say v and w, that are at incomparable distances from u: i.e., $\delta(u,v) \not\leq \delta(u,w)$ and $\delta(u,w) \not\leq \delta(u,v)$.

Numerical versions of distance and similarity can be derived from the conceptual distance by using the cardinal of the extension or intension. Given the conceptual distance $\delta(u,v) = (X,Y)$:

— the **extensional distance** $d(u,v) = |X|$ is the cardinal of the extension,

— the **intensional similarity** $sim(u,v) = |Y|$ is the cardinal of the intension.

Note that those numerical versions are degraded versions, as they flatten the partial ordering over conceptual distances to a total ordering. This can lead to consider two objects as being at the same distance, whereas the conceptual distances are completely different.

Definition 24 *Let $K = (O, A, I)$ be a formal context, and $u \in O$ be an object. The* ***concepts of neighbors*** *of u is the set of all conceptual distances from u to any other object in the context: $C\text{-}N(u) = \{\delta(u,v) \mid v \in O\}$.*

For example, Figure 4.2 presents the set of concepts of neighbors of *Charlotte* by their extensions, based on the formal context defined in Table 4.1. $C\text{-}N(Charlotte)$ contains four concepts. The first one has for intension $\{woman, kid\}$ and for extension $\{Charlotte\}$. Then there are two more general concepts: $(\{Charlotte, Diana, Kate\}, \{woman\})$ and $(\{Charlotte, George\}, \{kid\})$. Finally, there is the most general concept, with an empty intension and the whole set of objects as extension.

For any context $K = (O, A, I)$ and any object $u \in O$, the concepts of neighbors $C\text{-}N(u)$ provide an instance-based view over the context, by partitioning the set of ob-

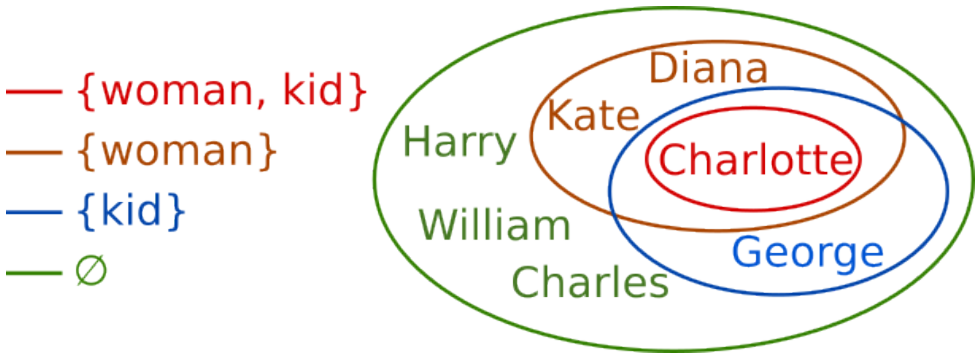

Figure 4.2 – Representation of the concepts of neighbors of *Charlotte* by their extensions

jects according to their conceptual distance with u. Therefore, the set of the concepts of neighbors of u is a subset of the formal concepts of C having u in their extension.

A direct consequence of the definition of the concepts of neighbors is that, whereas the number of concepts of K is majored by 2^n (with $n = card(O)$), the number of concepts of neighbors of u $card(C\text{-}N(u))$ is bounded by n.

Each concept of neighbors $\delta \in C\text{-}N(u)$ induces the subset of objects $\delta.proper := \{o \in O \mid \delta(u, o) = \delta\}$, i.e., the objects that are exactly at distance δ from u. This is a subset of the extension of δ, called the **proper extension** of δ.

A notion of **rank** can also be defined on the concepts of neighbors. As shown in Figure 4.2, for any object $u \in O$, the extensions of the concepts of neighbors of u form a hierarchical clustering of O. From this observation, we can define recursively the rank of a concept such as:

— $rank(\delta(u, u)) = 0$
— for $\delta \in C\text{-}N(u)$, $rank(\delta) = 1 + max\{rank(d) | d \in C\text{-}N(u), d \leq \delta\}$

In the example presented in Figure 4.2, there is one concept of rank 0 (the one in red containing only *Charlotte* in its extension), two of rank 1 (the concept in brown describing the women and the concept in blue describing the kids) and one of rank 2 (the top concept, in green). From this notion can be defined the notion of **concepts of nearest neighbors** of u, *i.e.*, the concepts of neighbors of u of rank 1. The concepts of nearest neighbors of u are the concepts describing the objects the more similar to u. Here, the concepts of nearest neighbors of *Charlotte* are the concept in blue (the concept describing the kids) and the one in brown (describing the women).

4.1.2 Concepts of Neighbors and FCA extensions

As presented in the previous section, the central notion for defining the concepts of neighbors is the notion of *conceptual distance* between two objects, *i.e.*, the most specific concept containing those two objects in their extension. Therefore, as long as this notion can be transposed, the notion of concepts of neighbors can be defined on any extension of FCA. Among those extensions, we can cite Logical Concept Analysis (LCA, [FR00]) for logical expressions, Fuzzy Formal Concept Analysis (FFCA, [Yan+08]) for fuzzy sets, or Relational Concept Analysis (RCA, [Rou+13]) and Graph-FCA [FC20] for relational data. Once the notion of conceptual distance is defined, the notion of concepts of neighbors of an object u is naturally defined as the set of the conceptual distances of u to the other objects.

In the case of Graph-FCA – that is, as presented in Chapter 1, a comprehensive extension of FCA to relational data –, when we extend the definitions of conceptual distance we naturally fall back on the initial definitions of concepts of neighbors, as presented in [FC20] (see Chapter 1). By consequence, the algorithms presented for the efficient computation of concepts of neighbors in Graph-FCA in Section 2.3.2 still stand.

4.2 Graph-FCA and the Case of n-ary Concepts

As presented in Chapter 2, an algorithmic framework for the efficient computation of concepts of neighbors in Graph-FCA case has been developed [Fer18].

A specificity of Graph-FCA is that, unlike in classical FCA, there is a notion of *arity*. A graph concept of arity k (also called a *k-concept*) is a concept having as extension a set of k-uples of objects and as intension a k-PGP. This has a direct combinatorial consequence: in FCA, the set of the concepts of neighbors of an object $u \in O$ contains at most $n = card(O)$ concepts, whereas in Graph-FCA the set of concepts of neighbors of a k-uple $u \in O^k$ contains at most n^k elements.

However, the algorithm for the efficient computation of concepts of neighbors works by partitioning the whole set of candidate instances (*i.e.*, the whole set of k-tuples of object) into proper extensions of concepts. Therefore, the explosion of the number of candidate instances makes this computation intractable. In previous works, even if Concepts of Neighbors have been theorized in the n-ary case, the existing applications were on the unary case (for knowledge graph completion [Fer20] or query relaxation [Fer18]) or on small datasets, and this tractability issue has not been addressed. But in the relation

classification application (see Chapter 5), we encounter this tractability issue. To address it, we propose to restrain the set of candidate instances to a subset of O^k.

The nature of the subset $V \subseteq O^k$ is not fixed and depending on the application. For example, as evoked in the introduction, the Concepts of Neighbors can be used to perform instance-based learning on graphs. In this case, V can be restrained to the set of annotated tuples of entities, as those are the entities used for classifying a new query instance. This is the solution used in Chapter 5 on the relation classification task.

Definition 25 *Let $K = (O, A, I)$ be a graph context, $u \in O^k$ a tuple of objects, and $V \subseteq O^k$ a set of candidate instances. The* ***projected concepts of neighbors*** *of u on V is defined as:*

$$C\text{-}N_V(u) = \{(R \cap V, Q) \mid (R, Q) \in C\text{-}N(u),\ R \cap V \neq \emptyset\}$$

This definition brings an adaptation of the definition of the initial pre-concept in the partitioning algorithm for the computation of the concepts of neighbors in order to compute $C\text{-}N_V(u)$ instead of $C\text{-}N(u)$.

Definition 26 *The* initial pre-concept *is the pre-concept C_{init} s.t.:*

- *$Q_{init} = [\overline{u} \leftarrow \emptyset]$ is the empty PGP ($\overline{x}_{init} = \overline{u}$);*
- *$R_{init} = ans(Q_{init}) \cap V = V$ is the set of all candidate instances;*
- *$V_{init} = V$ is the set of all candidate instances;*
- *$M_{init} = (\overline{u}, V)$;*
- *$I_{init} = I$ is the set of all incidences from the graph context.*

Once this initialization modified, Algorithm 1 presented in Section 2.3.2 can be applied as it is, in order to compute $C\text{-}N_V(u)$.

4.3 Concepts of Neighbors on RDF graphs

Graph-FCA, as evoked before and as presented in Chapter 1, is an FCA extension to relational data. More specifically, a graph context $K = (O, A, I)$ is a labelled oriented multi-hypergraph, O being its set of vertices, A being its set of hyperedge labels (also called relations), and I being its incidence relation. Meanwhile, the RDF format (*Resource Data Framework*, the standard format for describing knowledge graphs in the semantic web), describes a set of *(subject, relation, object)* triples, forming a labelled oriented multigraph,

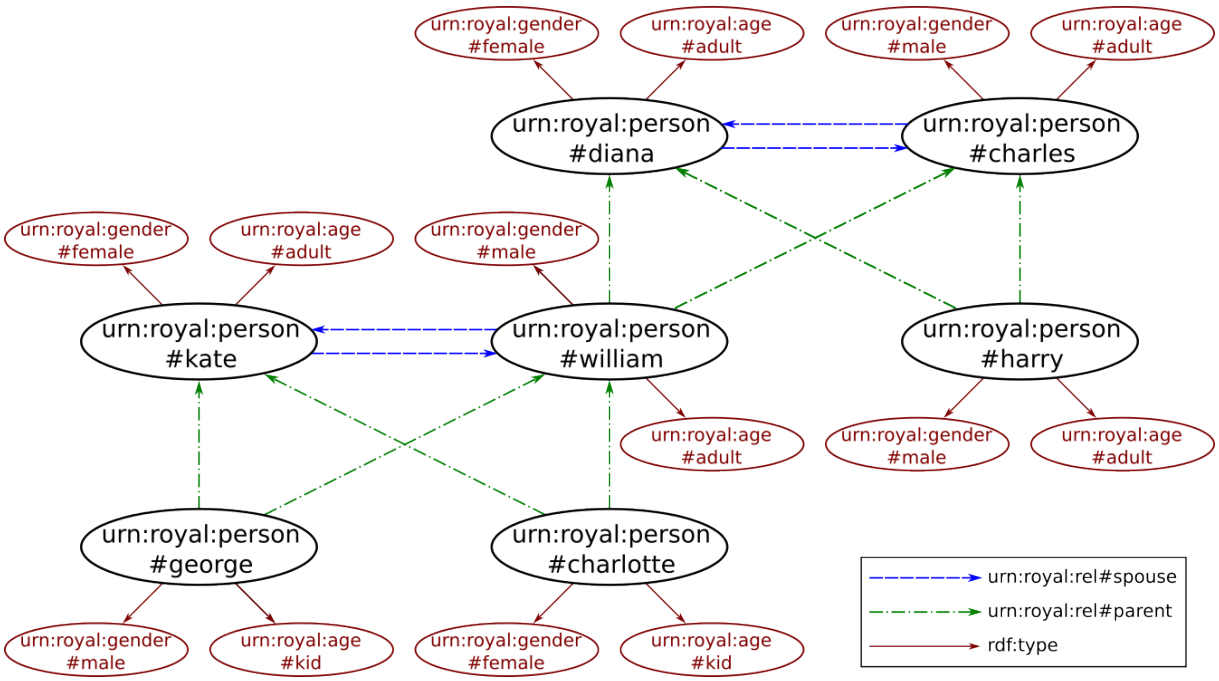

Figure 4.3 – RDF graph representing the British royal family

including an ontology – called *schema* – and an inference mechanism. In this section, we first present how to transform an RDF graph into a context graph. Then, we present how to integrate the inference mechanisms in the Concepts of Neighbors. Finally, we present CONNOR, a Java implementation of the Concepts of Neighbors on RDF graphs.

4.3.1 RDF Graph as Graph Context

An RDF graph is a set of triples (s, r, o), s (the subject) being a URI or a blank node, r (the predicate) being a URI and o (the object) being a URI, a blank node or a literal value. s and o are *entities*, while r is a *relation*. Figure 4.3 represents an RDF of an excerpt of the British royal family. Here, entities are represented by ovals, and relations by arrows.

Algorithm 3 presents a procedure to translate RDF graphs into graph context. In this procedure, the triples that define the type of an entity (*i.e.*, triples having *rdf:type* as relation) are treated specifically: the subject is associated to a variable, and a label (*i.e.*, a unary hyperedge) is added to the graph context, labelling this variable with the subject of the triple. In the other case, the subject and the object are associated to variables, and an edge (*i.e.*, a binary hyperedge) labelled by the relation is added between these two

Algorithm 3 Translation of RDF graph as graph context

Require: An RDF graph T
Require: An infinite set of variables $\mathcal{V}$
Ensure: A graph context $K = (O, A, I)$ equivalent to the RDF graph

$O \leftarrow \emptyset$, $A \leftarrow \emptyset$, $I \leftarrow \emptyset$
$D \leftarrow$ an empty dict
for each $(s, r, o) \in T$ **do**
  if $s \in keys(D)$ **then**
    $v_s \leftarrow D.get(s)$
  else
    $v_s \leftarrow$ a new variable from $\mathcal{V}$
    $D.put(s, v_s)$
  end if
  $O \leftarrow O \cup \{v_s\}$
  if s is an URI **then**
    $A \leftarrow A \cup \{s\}$
    $I \leftarrow I \cup \{((v_s), s)\}$
  end if
  if $r =$ *rdf:type* **then**
    $A \leftarrow A \cup \{o\}$
    $I \leftarrow I \cup \{(v, o)\}$
  else
    if o is a literal **then**
      $v_o \leftarrow$ a new variable from $\mathcal{V}$
    else if $o \in keys(D)$ **then**
      $v_o \leftarrow D.get(o)$
    else
      $v_o \leftarrow$ a new variable from $\mathcal{V}$
      $D.put(o, v_o)$
    end if
    $O \leftarrow O \cup \{v_o\}$
    if i is an URI or a litteral **then**
      $A \leftarrow A \cup \{o\}$
      $I \leftarrow I \cup \{(v_o, o)\}$
    end if
    $A \leftarrow A \cup \{r\}$
    $I \leftarrow I \cup \{((v_s, v_o), r)\}$
  end if
end for
return (O, A, I)

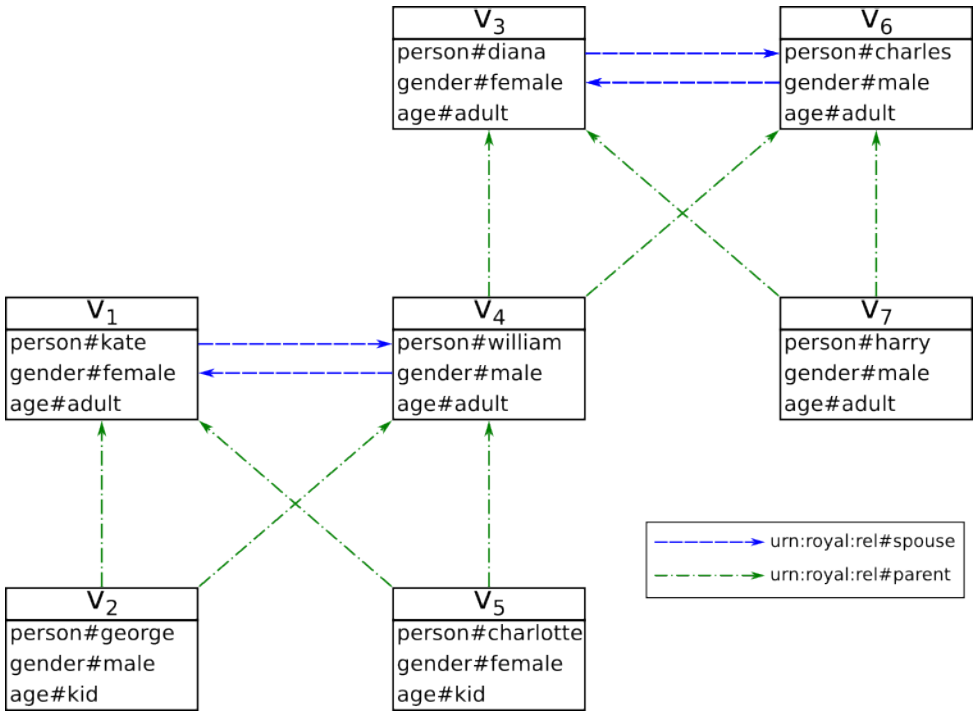

Figure 4.4 – Translation of Figure 4.3 as a graph context

variables. In addition, if an entity is a URI or a literal, the variable associated with this entity is labelled with its URI or literal. Note that each instance of a literal has its own variable.

In this representation, each RDF entity is represented by an object labeled by its URI or literal – or unlabeled in the case of a blank node. The advantage of this representation is that each object is an anonymous variable, and the name of the entity encoded by this variable is considered as a property of this variable. This allows for RDF entities to be processed anonymously. In the case of Concepts of Neighbors, it allows having anonymous entities in the intension of the concept.

Figure 4.4 represents the same data as Figure 4.3 translated into a graph context. In this representation, objects are represented as boxes with their labels listed under, and the edges are represented by arrows. It can be observed that the entity representing *George* in Figure 4.3 is translated as an object v_2, labelled by its URI (`person#george`) and its types (`gender#male` and `age#kid`), and is linked to the objects v_1 and v_4 by two edges of attribute `rel#parent`.

```
1  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2
3  urn:royal:age#adult
4      rdfs:subClassOf urn:royal:human .
5
6  urn:royal:age#kid
7      rdfs:subClassOf urn:royal:human .
8
9  urn:royal:rel#spouse
10     rdfs:domain urn:royal:age#adult ;
11     rdfs:range urn:royal:age#adult .
12
13 urn:royal:rel#parent
14     rdfs:range urn:royal:age#adult .
```

Table 4.2 – Example of T-box in *turtle* notation

4.3.2 RDF Ontology and Concepts of Neighbors

In addition to the triples representing facts (also called *A-box*), an RDF graph can also contain a series of triples representing an ontology (also called *schema* or *T-box*) on the graph (see [HKR09]). This schema gives extra information on the facts by adding a hierarchy over the types (with the triples of relation *rdfs:subClassOf*) and over the relations (with the triples of relation *rdfs:subPropertyOf*), and type constraints on the subjects and objects of a relation (with the triples of relation *rdfs:range* or *rdfs:domain*).

Table 4.2 shows a T-box for the RDF graph represented in Figure 4.3. This T-box indicates that entities that are adults or kids are humans, that if an entity has a spouse or is the spouse of another entity then it is an adult, and that only adults can be parents.

The existence of a T-box implies a saturation mechanism: the A-box can be saturated with new facts thanks to the T-box. However, the saturation process of a graph can be costly, and has to be run after any modification of the graph.

In order to palliate the cost implied by this saturation, mechanisms can be added to the partitioning algorithm in order to take into account the type and relation hierarchy while computing the concepts of neighbors. This way, this information is taken into account into the intensions of the concepts, even while working on an unsaturated RDF graph. To do so, we first need to define a function for listing the relaxation of an incidence according to the T-box.

Algorithm 4 $Relax((o_1, \ldots, o_k), a)$

Require: A graph context $K = (O, A, I)$ corresponding to the A-box of the RDF graph
Require: An incidence $((o_1, \ldots, o_k), a)$
Require: A set of triplets T forming the T-box of the RDF graph
Ensure: A set of incidences J induced by $((o_1, \ldots, o_k), a)$ and the T-box

1: $J \leftarrow \emptyset$
2: **if** $k = 1$ **then**
3: $S \leftarrow \{c \mid (a, \textit{rdfs:subClassOf}, c) \in T\}$
4: **for** $c \in S$ **do**
5: $J \leftarrow J \cup \{((o_1), c)\}$
6: **end for**
7: **else if** $k = 2$ **then**
8: $S \leftarrow \{p \mid (a, \textit{rdfs:subPropertyOf}, p) \in T\}$
9: **for** $p \in S$ **do**
10: $J \leftarrow J \cup ((o_1, o_2), p)$
11: **end for**
12: $S \leftarrow \{c \mid (a, \textit{rdfs:domain}, c) \in T\}$
13: **for** $c \in S$ **do**
14: $J \leftarrow J \cup ((o_1), c)$
15: **end for**
16: $S \leftarrow \{c \mid (a, \textit{rdfs:range}, c) \in T\}$
17: **for** $c \in S$ **do**
18: $J \leftarrow J \cup ((o_2), c)$
19: **end for**
20: **end if**
21: **return** J

Algorithm 4 presents this function: for a given incidence, if this incidence is a label, then it relaxes the incidence into the superclasses of this label. If the incidence is an edge, then it relaxes this incidence into the superproperties of the edge's relation type, and types the object and subject of the edge according to the domain and range of this relation type.

For example, we can apply this algorithm to the incidence $((v_2, v_4), \texttt{rel\#parent})$ of the graph context presented in Figure 4.4 with the T-box detailed in Table 4.2. As in the T-box, the relation `rel#parent` has the class `age#adult` as range, line 18 of the algorithm adds $((v_4), \texttt{age\#adult}$ to the induced incidences. Then, if we apply the same algorithm to this new incidence, as `age#adult` has for superclass `human`, we obtain the incidence $((v_4), \texttt{human}$.

Once this relaxation function is defined, the algorithm for the computation of concepts

```
1  ConnorModel model = new ConnorModel(rdfModel);
2  List<String> target = new ArrayList<>({"urn:royal:person#
       charlotte"});
3  Table table = Table.ofArity(1);
4  rdfModel.listSubjects().forEachRemaining(s ->
5          table.addInitRow(IntStream.of(1)
6                  .mapToObj(i -> s.asNode())
7                  .collect(Collectors.toList())));
8  );
9  ConnorPartition partition = new ConnorPartition(model, target,
       table, 0);
10 AtomicBoolean cut = new AtomicBoolean(false);
11 ConnorThread thread = new ConnorThread(partition, cut);
12 thread.start();
13 thread.join(2 * 1000); // 2000ms = 2s
14 cut.set(true);
15 System.out.print(partition.toJson());
```

Table 4.3 – Example of usage of CONNOR

of neighbors has to be modified in order to add the relaxed incidences to the set of incidences to explore. To do so, line 9 Algorithm 1 in Chapter 2 has to be modified as:

$$I_{ij} = Relax(w, a) \cup (I_l \setminus \{(w, a)\})$$

4.3.3 CONNOR: a Java Implementation

The main implementation of the Concepts of Neighbors is CONNOR, a Java library, presented in the article [ACF22b]. This library is a free and open-source software [1], based on Apache Jena [2], a well-known Java library for representing and reasoning on the RDF graphs of the Semantic Web. In this library, graph contexts are represented as RDF graphs and handled with the Jena library. This library gives a comprehensive interface for the handling of graph contexts and concepts of neighbors, and provides classes encapsulating the algorithms for the efficient computation of concepts of neighbors.

Table 4.3 presents an example code for the computation of the concepts of neighbors of `person#charlotte` in the RDF graph presented in Figure 4.3.

1. Accessible here: `https://gitlab.inria.fr/hayats/CONNOR`
2. `https://jena.apache.org/`

The CONNOR library is structured into four main classes, which are detailed below.

`ConnorModel` This class encapsulates an RDF model that represents a graph context. In addition to this model are added data structures made in order to access more rapidly to the triples of the graph, as well as structures to keep in memory results of queries made on the model. There are two ways to create a model using this class. The first way consists in creating an empty model and adding triples one by one. The second way consists in creating a model from a pre-existing RDF model. Line 1 of Table 4.3 shows an example of creation of `ConnorModel` from an RDF graph.

`ConceptOfNeighbors` As its name tells, this class represents a concept of neighbors, and hence a graph concept too. As expected, an object of this class is characterized by its intension (decomposed into two attributes: the list of the projection variables and the graph pattern), its extension and its proper extension. Those elements can be accessed through usual getter methods. An object of this class can easily be transformed into a JSON object serialization and further processing via the `toJson` method. The creation of objects of this class is handled by the `ConnorPartition` class, and should not be done by library users.

`ConnorPartition` This class is central to the computation of concepts of neighbors. It takes its name from the fact that, as presented above, the proper extensions of the concepts of neighbors form a partition of the set of objects. This class contains all the information needed for the computations of concepts of neighbors, such as of course the graph context (represented by a `ConnorModel`), the concepts of neighbors once the computation is done (represented by a collection of `ConceptOfNeighbors` objects), but also the tuple of objects (called *target*) for which we want to compute the concepts of neighbors. A `ConnorPartition` object can be translated into JSON for serialization and further processing. An example of serialization in JSON is given by Line 15 of Table 4.3.

This class implements the algorithms presented in Section 2.3.2, as well as the relaxation mechanism presented in Section 4.3.2.

To use this class, the base constructor of this object takes as argument a `ConnorModel` object, the target (represented by a list of URIs), the set of candidate instances and a `maxDepth` parameter. An example of usage is given by Section (4) of Table 4.3. The set of candidate instances is represented by the parameter `initializationTable`, which is

a `Table` object which represents a set of tuples of entities. An example of construction of such a table is given in Table 3-8 of Table 4.3. Concerning the `maxDepth` parameter, its role is to set, if desired, a limit to the depth of the intension from the elements of the target. If set to zero, no limit is applied.

The class method to call in order to launch the computation of concepts of neighbors is called `fullPartitioning(cut)`. Taking a mutable Boolean named `cut` as a parameter, it refines the partition until convergence or until `cut` is switched to `true`.

`ConnorThread` This class encapsulates the computation of concepts of neighbors using a `ConnorPartition` in a Java thread, so that the main thread just needs to launch this thread and switch `cut` to `true` when desired. An example of usage is given by Lines 10-14 of Table 4.3.

4.4 Conclusion

This chapter presents various contributions made on Concepts of Neighbors, an FCA-based approach for instance-based learning on symbolic data. First, we present a new formalization of Concepts of Neighbors in the general FCA framework, and a method for extending those notions to FCA extensions, including Graph-FCA. Then we introduce how to manage the case of n-ary concepts, which is a specificity of concepts of neighbors in the Graph-FCA framework. Finally, we introduce the application of Concepts of Neighbors on RDF graphs, by presenting a translation of RDF graphs into graph context, introducing the management of RDF schemas in the computation of concepts of neighbors, and by promoting CONNOR, a Java library specialized for Concepts of Neighbors on RDF graphs.

Several aspects of this work would need further researches. For example, the existing algorithms for the computation of concepts of neighbors in the Graph-FCA case could be adapted and extended to other FCA extensions, in order to expand the applications of this approach. Still from an algorithmic point of view, an enhancement of the *LazyJoin* algorithm (presented as Algorithm 2 in Section 2.3.2) have been theorized, through a post-insertion downward propagation reducing the size of the match-sets. Preliminary experiments have been run and have shown a positive impact of this modification on unary cases, but further work is needed to adapt it to the n-ary case. In complement, different strategies for choosing an incidence in the partitioning algorithm would need exploration and further enhancements. From a more global point of view, it can be pointed out that

Concepts of Neighbors are still a quite novel approach, still subject to new applications and diverse enhancements.

Chapter 5

Two-Step Explainable Relation Extraction with Concepts of Neighbors

The *semantic web* (*SW*) and its *knowledge graphs* (KGs), as evoked before, evolved during the last decade from a concept theorized early in the History of computer science [GS21] and formalized in 2001 [BHL01] to a field both invested by academics and industrial actors [Hit21]. Coupled with the growth of the usage of the World Wide Web, which caused a great amount of texts and knowledge to be freely accessible, this created a need for *information extraction* (IE) methods able to translate the semantic data contained in texts into knowledge graphs. Amongst the tasks of IE, we focus here on the *relation extraction* (RE) task – the task consisting into identifying the relation (if any) between two entities.

The work presented in this chapter proposes a novel, two-step explainable RE method. This approach divides the RE task into two sub-tasks – *relation detection* and *relation classification* – and while using a state-of-the-art deep learning model for the relation detection task, it grants interactivity and interpretability by using a novel, symbolic and explainable method for relation classification. This method is based on the modeling presented in Section 3.2.2 and on the Concepts of Neighbors, a FCA-based approach that can be used for instance-based classification of graph entities. This chapter presents the results published in [ACF21; ACF22a].

In this chapter, we detail this two-step method for relation extraction. Section 5.1 presents an overview of the proposed method, and how the different modules of this method are inserted in the workflow presented in Chapter 3; Section 5.2 presents the Relation Detection module; Section 5.3 describes the Concepts of Neighbors-based Relation Classification module; Section 5.4 shows the experimental evaluation of this approach and of its modules; Section 5.5 concludes the chapter.

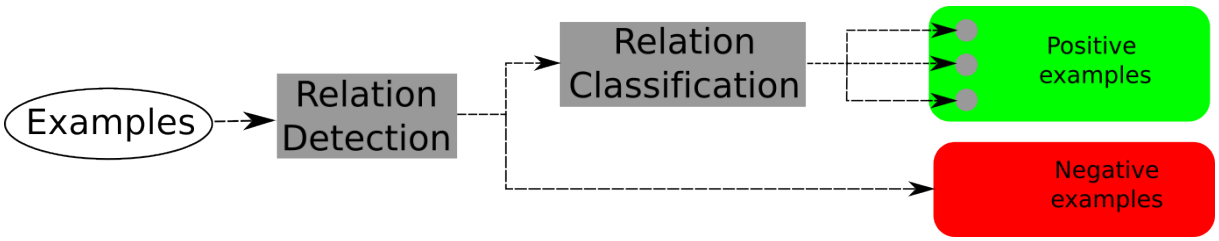

Figure 5.1 – Two-step relation classification process

5.1 Method Overview

Nowadays, as presented in Chapter 1, state-of-the-art systems for relation extraction use deep-learning large language models. However, these approaches do not fit the requirements presented in Chapter 3: they do not present the needed explainability properties, and they do not rely on an instance-based learning method. Therefore, we developed a new instance-based learning approach for relation extraction, using the modeling presented in Section 3.2.2. As this method works by similarity between the annotated examples, this conflicts with the fact that, to perform relation extraction, a method has to distinguish positive examples (*i.e.*, pairs of entities having a semantic relation between them) and negative ones (*i.e.*, pairs of entities that do not have such a semantic relation between them). Indeed, if it seems reasonable to consider that a positive example of a given relation class is similar to other examples of the same class, there is no reason for a negative example to be similar to any other negative example. Therefore, the developed method does not discriminate between positive and negative examples (a task called *relation detection*), but performs *relation classification*, *i.e.*, the task of determining the relation type presented in an example, knowing that this example is positive.

This is why we developed a two-step workflow for relation extraction, as presented in Figure 5.1. In this workflow, the method for relation detection discriminates the positive examples from the negative examples, and our instance-based method classifies the positive examples among the existing relation types. Such a two-step approach has already been exploited with promising results [Mal+21].

In addition, the relation detection and relation classification modules presented in this chapter are usable in the workflow presented in Chapter 3 and reproduced in Figure 5.2: the deep learning-based relation detection module fits in the preprocessing unit, while the FCA-based relation classification module, as an explainable instance-based learning,

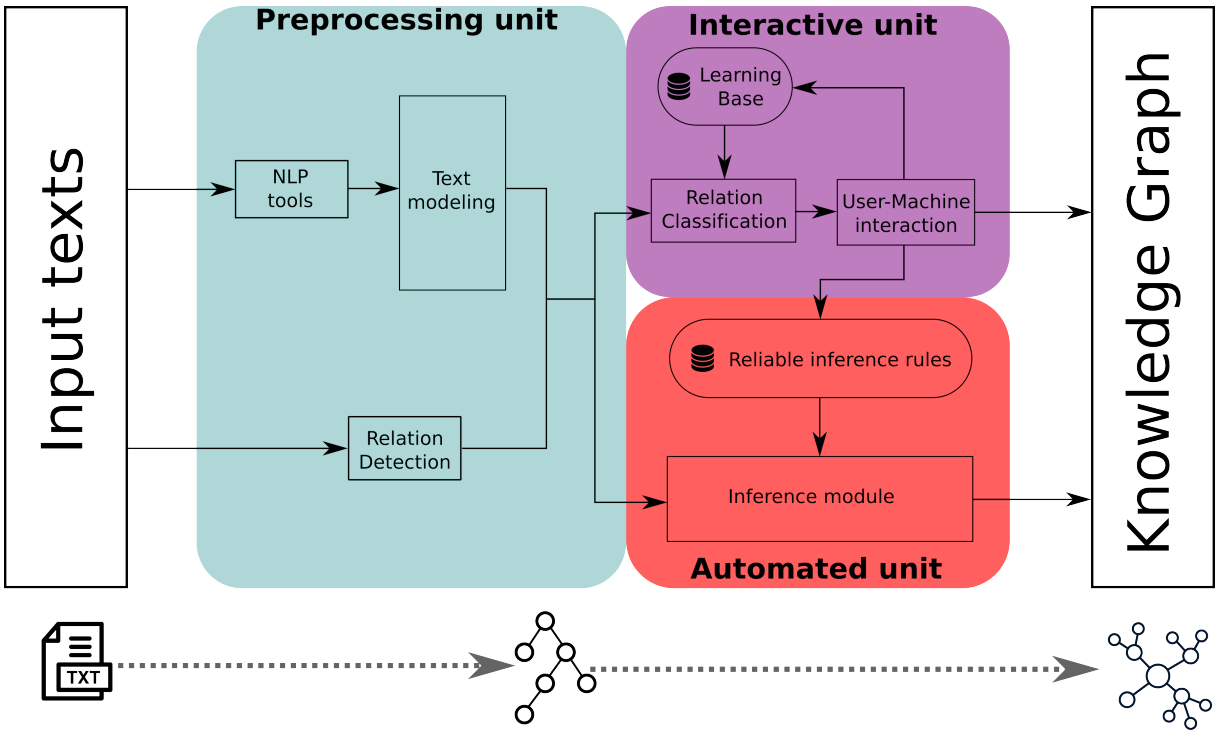

Figure 5.2 – Workflow overview

module fits in the interactive unit.

5.2 Relation Detection with Large Language Models

As there is, as far as we know, no efficient symbolic or fully explainable method for relation detection that we know of, we decided to favor performance and therefore to use a state-of-the-art deep learning approach. Moreover, there is not much need to explain the non-existence of a relation, and an explanation for a type of relation is also an explanation for the existence of a relation.

Today, the state-of-the-art in relation extraction is dominated by transformer-based *large language models* (LLM) such as BERT [Dev+19] and its variants. One of those variants, LUKE [Yam+20], has the particularity to handle both single words and multi-word entities, and has shown good results on diverse entity-based NLP tasks, including relation extraction. We decided to use this LLM as a relation detector.

We consider several configurations of LUKE for relation detection. The first one, called

luke-base, simply consists in reusing the fully trained model for relation extraction and post-process the output in order to merge all the positive predictions into one class. A second configuration, called *luke-detect* consists in specializing LUKE for relation detection. We remove LUKE's last classification layer, and replace it by two layers: a fully connected layer of size n and an output neuron with a *sigmoid* activation function. Then the model is fine-tuned in order for it to predict 1 on the positive examples and 0 on the negative examples.

As this second configuration is specialized for relation detection (where the first performs relation detection through relation extraction), we expect this second configuration to show better results. Several sizes for the hidden fully-connected layer could be tested.

5.3 Relation Classification with Concepts of Neighbors

As presented in Chapters 1 and 4, the Concepts of Neighbors is an FCA-based approach for instance-based learning that can be applied on entities in graph data. In this section, we present how to perform relation classification, using the Concepts of Neighbors approach on the modeling of texts as RDF graphs presented in Chapter 3. First, this section presents the setting for using the Concepts of Neighbors method on this graph modeling. Then, it presents scoring strategies used for performing relation classification from concepts of neighbors.

The task is the following: from a test example, which is a sentence annotated with the position and type of two entities (a subject and an object), and from a set of training examples, which are additionally annotated with a relation type, and assuming that a relation linking those two entities exists, how to predict the relation type linking the subject and the object of the test example? The proposed idea is, from the modeling presented in Section 3.2.2, to compute the concepts of neighbors of the $(subject, object)$ pair of entities, and deduce from the annotation of the training examples what is the relation type that is the more likely to link the subject to the object.

Table 5.1 presents an example of training dataset and test example for the relation classification task. Here, the objective is to predict the relation linking $Gödel_7$ to $Austria_7$ in the given sentence, based on the six annotated examples. Entities are numbered in order to distinguish two spans of the same entity (*e.g.*, $Russel_6$ designate the entity span of *Russel* in the sixth example).

#	Sentence	Subject	Subj. type	Object	Obj. type	Relation type
Training examples						
(1)	*Wittgenstein was born in Austria*	$Wittgenstein_1$	`person`	$Austria_1$	`place`	`place_of_birth`
(2)	*Wittgenstein died in England*	$Wittgenstein_2$	`person`	$England_2$	`place`	`place_of_death`
(3)	*Wittgenstein is a philosopher*	$Wittgenstein_3$	`person`	$philosopher_3$	`title`	`pers:title`
(4)	*Russel was born in England*	$Russel_4$	`person`	$England_4$	`place`	`place_of_birth`
(5)	*Russel died in England*	$Russel_5$	`person`	$England_5$	`place`	`place_of_death`
(6)	*Russel is a philosopher*	$Russel_6$	`person`	$philosopher_6$	`title`	`pers:title`
Test example						
(7)	*Gödel was born in Austria*	$Gödel_7$	`person`	$Austria_7$	`place`	

Table 5.1 – Training dataset and test example for the relation classification task

In this section, we first present how to compute concepts of neighbors on the modeling for performing relation classification while keeping tractable execution time. Then, we introduce scoring methods for transforming a set of concepts of neighbors into a relation type prediction.

5.3.1 Concepts of Neighbors for Relation Extraction in Texts

In order to use the Concepts of Neighbors approach on the modeling presented in Section 3.2.2, a few aspects need to be addressed. First, we need to clearly identify the RDF nodes representing the subject and the object in each sentence. Then, as presented in Chapter 4, the problem of the explosion of number of candidate instances has to be faced, due to the fact that we need to compute the concepts of neighbors for a $(subject, object)$ couple of entities. This problem is addressed in this specific case of relation classification through several optimizations, in order to keep a tractable computing time.

Identification of the Subjects and Objects Let us consider a set of sentences in which are identified $(subject, object)$ couples of entities by their textual spans [1]. Once a relation detection module has been used to identify the couples of entities that are linked by a relation, and once the sentences are modeled according to the modeling presented

1. Note that there can be several couples of entities identified by sentence.

in Section 3.2.2, there is a need to unambiguously identify the nodes of the RDF graph forming the $(subject, object)$ couples. The issue is that subjects or objects can overlap several tokens. The strategy to collapse the named entities presented in Section 3.2.2 solves this problem in the vast majority of cases, as subjects and objects are most of the time either named entities or one-token expressions. However, ambiguous cases still exist in a small proportion of the sentences. This problem can appear in three cases: first, when the subject or object is expressed as a nominal group (e.g., *"the man"*, *"the university"*); second, when the subject or object includes a named entity but is longer than it (e.g., *"the President of the United States of America"* is a subject, whereas only *"United States of America"* is tagged as a named entity); third, when the subject or object is a named entity that has not been tagged by the named entity recognition module.

The solution proposed to solve these cases is similar to the solution presented in Section 3.2.2 to collapse named entities: as a subject or an object is necessarily a group of contiguous tokens with a particular meaning, it must form a tree factor in the dependency tree. Therefore, it can be considered that the root of this factor carries the semantic and syntactic information, and then can be pointed out as the subject or the object. Like for named entities, if the tokens do not form a tree factor because of a wrong annotation or a parsing error, the last token is used instead of the root. It can be pointed out that, unlike for named entities for which the whole tree factor was merged into a unique token, the choice has been made not to merge the tokens constituting the subject or the object as their syntactic structure (if any) is generally informative, like in *"the President of X"*.

Restriction to Annotated Examples The other main issue is the large number of candidate instances: as detailed in [Fer20], to compute the Concepts of Neighbors for a tuple of k objects in a graph involving n objects, the algorithm has to generate and partition n^k tuples. Therefore, for a large graph, the computation of concepts of arity greater than 1 is rapidly intractable. In the present case, if we consider a dataset composed of several thousands sentences, there are billions of candidate instances. This issue is solved by computing the *projected concepts of neighbors* of the $(subject, object)$ couple from a test sentence on the $(subject, object)$ couples of training sentences, as presented in Section 4.2. It permits to simultaneously drastically reduce the computation cost and remove useless data from the computed concepts – *i.e.*, the couples of entities that are not annotated – while keeping all the knowledge of interest. From a numerical point of view, in the case of a training dataset of n sentences having in average k tokens, this optimization reduces the

number of candidate instances from k^2n^2 to n. For a training dataset of 10,000 sentences with average length of 20 tokens per sentence, we obtain a reduction of the numbers of candidate instances from 40 billions to 10,000.

In the case of the example presented in Table 5.1, the set of candidate instances V contains the following pairs:

$$\begin{aligned} V = \{&(Wittgenstein_1, Austria_1), (Wittgenstein_2, England_2), \\ &(Wittgenstein_3, philosopher_3), (Russel_4, England_4), \\ &(Russel_5, England_5), (Russel_6, philosopher_6)\} \end{aligned}$$

The other pairs, such as $(Wittgenstein_1, Wittgenstein_1)$, $(Wittgenstein_2, Russel_4)$ or $(England_5, Russel_5)$ are removed from this set, as we have no interest to include them in our concepts of neighbors.

Relation Type Restriction In addition, in a relation extraction dataset each subject and object has a type, and these types can be used to reduce the set of candidate instances further. For example, if an example has for subject a person and for object a location, it can be seen that the relation expressed by this example could be *place_of_birth* or *place_of_living*, but cannot be *age* or *parent*. Therefore, for a given couple *(subject type, object type)*, a set of compatible relations can be deduced from the training dataset. If there is only one compatible relation, this relation can be predicted without computing Concepts of Neighbors for this example, and if there are several possible relations, the set of *(subject, object)* couples from the training dataset that are annotated with compatible relation types can be used as the set of possible neighbors in the algorithm. From an external point of view, this optimization is equivalent to the fact of having a different classifier for every pair of subject and object type. This idea of using the types of the subject and of the object to restrain the set of possible relation types has been developed independently and simultaneously to our work as the RECENT paradigm in [LC21].

In our case example, our test individual (7) has for subject type `person` and for object type `place`. By searching in the learning dataset, we can observe that to this pair of types are is only associated two relation types: `place_of_birth` and `place_of_death`. Therefore, only the training examples annotated with one of those relation types are conserved in the set of candidate instances, *i.e.*, examples (1), (2), (4) and (5).

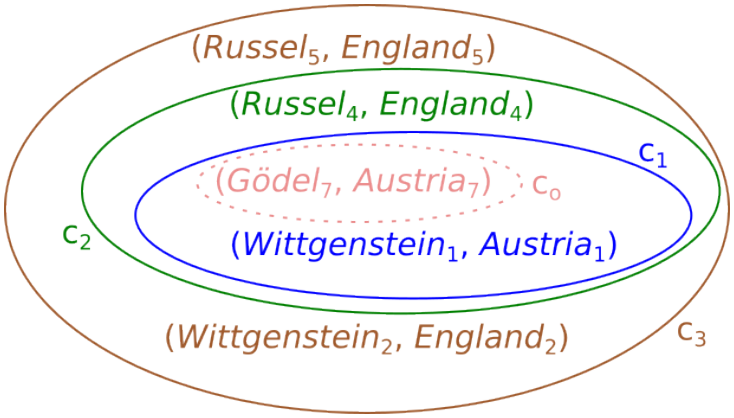

Figure 5.3 – Extensions of the concepts of neighbors of $(Gödel_7, Austria_7)$

Dependency Tree Pruning Another optimization has been proposed, this time on the modeling itself. The idea, first proposed in [ZQM18], states that not all dependencies are of the same interest for extracting relations: only those close to the path between the subject and the object carry useful information. However, it can be easily seen that reducing the dependency tree to a path would remove essential information for relation extraction, e.g., in the case of a negation attached to a verb that is on the path. Our solution is to prune the dependency tree to keep only the path between the subject and the object, plus the tokens up to maximal distance K from this path. Several values of K were tested, and the value $K = 1$ appears to be a good trade-off between size reduction and performance.

5.3.2 Scoring Methods

The computation of Concepts of Neighbors of a *(subject, object)* pair from the test dataset returns a set of concepts, each concept is associated to a set of neighbor couples, and to an extensional distance. In addition, the specialization presented in the previous section ensures that each neighbor couple is annotated with a compatible relation type. From this result, in order to be able to predict a relation, we need to associate a score to each relation type. In the following, we present two scoring methods: one based on a weighted vote, and another based on the confidence measure.

Figure 5.3 represents the extensions of the concepts of neighbors of the couple ($Gödel_7$, $Austria_7$) amongst the training examples of the dataset presented in Table 5.1, once the restriction on the set of candidate instances have been made. The objective here is, based

on these concepts, to produce a prediction for the relation type of individual (7). Note that in this figure, we represent the concept c_0 for readability reasons, but as it does not contain any annotated example, it is not part of the projected concepts of neighbors, and therefore it is not used for classification.

Exponential-weighted Vote (EV). For this scoring method, each neighbor (s, o) of each concept "votes" for its annotated relation type, denoted $r(s, o)$. However, if not weighted, this method will only reflect the proportion of each relation among all training examples annotated with a compatible relation. To avoid this problem, the extensional distance $dist(c)$ of concept c can be used to weight each vote. The extensional distance of a concept of neighbors measures the degree of similarity between the couple from which the concept has been computed and the neighbors that the concept contains: the lower the distance, the higher the similarity. Therefore, each vote is weighted by a decreasing function of the extensional distance.

$$score(R, C) := \sum_{c \in C} \sum_{(s,o) \in proper(c)} w(c)\ \mathbf{1}_{r(s,o)=R} \quad \text{where } w(c) = e^{-\frac{dist(c)}{A}}$$

Here, $\mathbf{1}_{r(s,o)=R}$ denotes the binary function that returns 1 if $r(s, o) = R$, 0 otherwise. We have chosen the inverse exponential function to define each weight $w(c)$ because of its rapid decrease, which privileges nearest neighbors. This way, the relation of one very similar example is preferred to the relation of many vaguely similar examples. As the exponential function is rapidly decreasing, the constant A is here to keep the value of $w(c)$ above the floating point numbers precision.

In the example presented in Table 5.1 and Figure 5.3, by taking $A = 1$, the relation types would have the following scores:

$score(\texttt{place\_of\_birth}, C) = e^{-1} + e^{-2} \approxeq 0.503$
$score(\texttt{place\_of\_death}, C) = 2 \times e^{-5} \approxeq 0.036$

From those scores, we can predict that individual (7) has for relation type `place_of_birth`, which is true.

Maximum Confidence (MC). The second method is similar to the method used by AnyBURL [Mei+19], and has been successfully used with Concepts of Neighbors for link

prediction [Fer20]. The idea is to consider the intension $int(c) = [s, o \leftarrow P_c]$ of each concept c, to use the graph pattern P_c as the body of a rule, and for each relation type r to compute the confidence of the rule $\mathbf{R}_{c,r} : P_c \rightarrow r(s, o)$, defined as usual as:

$$conf(\mathbf{R}_{c,r}) = \frac{|\{(s, o) \mid r(s, o)\} \cap ext(c)|}{|ext(c)|}$$

For each relation type r, the score is the list of the confidences of all rules $\mathbf{R}_{c,r}$ predicting that relation, in descending order.

$$score(r, C) := (conf(\mathbf{R}_{c,r}))_{c \in C} \text{ in descending order}$$

Such scores are ranked according to inverse lexicographic ordering. That is, the predicted relation type is the relation type with the higher maximal confidence. If several relation types have the same maximal confidence, the relation type with the higher second maximal confidence is predicted, and so on.

In our example Figure 5.3, we obtain the following confidence values:

$$conf(R_{c_1,\texttt{place\_of\_birth}}) = \frac{1}{1} = 1$$
$$conf(R_{c_2,\texttt{place\_of\_birth}}) = \frac{2}{2} = 1$$
$$conf(R_{c_3,\texttt{place\_of\_birth}}) = \frac{2}{4} = 0.5$$
$$conf(R_{c_1,\texttt{place\_of\_death}}) = \frac{0}{1} = 0$$
$$conf(R_{c_2,\texttt{place\_of\_death}}) = \frac{0}{2} = 0$$
$$conf(R_{c_3,\texttt{place\_of\_death}}) = \frac{2}{4} = 0.5$$

It gives us the following scores:

$$score(\texttt{place\_of\_birth}, C) = (1, 1, 0.5)$$
$$score(\texttt{place\_of\_death}, C) = (0.5, 0, 0)$$

As with the previous scoring method, the predicted relation is `place_of_birth`, which is correct.

5.4 Experiments

In this section, we present the different experiments made with our relation extraction system and the subsequent results. Those experiments can be divided in three parts: 1) the LUKE-based relation detection module, 2) the Concepts of Neighbors-based relation classification module, and 3) the whole RE method.

All experiments were made on the TACRED dataset [Zha+17], a widely-used standard dataset for relation extraction. This dataset is made of 106,264 examples, split into a training corpus (68,124 examples), a development corpus (22,631 examples) and a test corpus (15,509 examples). Each example of this dataset is a sentence with two entity mentions (a subject and an object), each mention being typed among 23 possible types (*e.g.*, *person*, *organization*, *place* or *date*), and annotated with a relation type among 41 effective classes (*e.g.*, *per:date_of_birth* or *org:city_of_headquarter*) plus a negative *no_relation* class representing the absence of relation between the subject and the object. For greater fidelity with real-world data, in which two entities randomly chosen are probably not linked by any relation, 79.5% of the examples are in the *no_relation* class. [2]

5.4.1 Relation Detection

We evaluate the different configurations of LUKE [Yam+20] presented in Section 5.2, in order to choose the best one for relation detection.

Experiment Design As presented in Section 5.2, several configurations of LUKE were tested. But first, on the *luke-detect* configuration, the hyperparameter n, fixing the size of the hidden fully-connected layer, has to be determined. Then, in addition to *luke-base* and *luke-detect*, a third configuration, called *luke-reprod* has been tested. Theoretically equivalent to *luke-base*, it consists into reproducing the fine-tuning on TACRED to see if this fine-tuning is reproducible, and to have another comparison point for *luke-detect*.

The code is freely accessible [3], and the experiments were run using a Tesla V100 GPU.

Results Table 5.2 presents the precision, recall and F1 scores obtained by *luke-detect* on the development dataset with several values for n, the size of the hidden layer. It appears

2. A complete description of the dataset can be found here: `https://nlp.stanford.edu/projects/tacred/`

3. See `https://gitlab.inria.fr/hayats/luke-redect`

n	P	R	F1
50	74.1	80.6	77.2
100	71.5	83.3	77.0
200	72.4	83.6	77.2
400	**74.5**	80.8	**77.5**
600	73.2	81.0	76.9
800	71.9	**83.8**	77.4
1000	73.3	81.6	77.2
1200	71.8	83.4	77.1
1400	73.0	81.9	77.2

Table 5.2 – Precision, recall and F-score for *luke-detect* with different sizes for the hidden layer on the development dataset

Approach	P	R	F1
luke-base	74.8	79.9	**77.3**
luke-reprod	**76.8**	75.2	76.0
luke-detect	73.1	**80.1**	76.4

Table 5.3 – Precision, recall and F-score for relation detection methods on the test dataset

that, even if this approach is not very sensitive to the size of the hidden layer, a maximum for the F1 is attained for $n = 400$. This is the value of n used in the following.

Table 5.3 shows the performance for the three detailed configurations. It can be read that, contrary to our expectations, *luke-reprod* does not reproduce the results from *luke-base*, by having an F-score inferior by 1.3 points. LUKE's implementation being in Python, this is probably due to a problem in dependency versioning. However, even if the reproduction was a failure, we can observe that *luke-detect*'s F-score is superior by 0.4 points to *luke-reprod*'s one. Therefore, it can be hoped that if we were able to reproduce perfectly *luke-base*, *luke-detect* would have a better F-score.

It is interesting to point out that if *luke-base* has an overall better F-score, *luke-reprod* outperforms its precision and *luke-detect* outperforms its recall. However, having a lower recall means having more false-negative examples, which means missing some examples expressing a relation, which we want to avoid, while having a lower precision means trying to classify a relation on examples that express none, which is also problematic. This is why we prefer F-score over precision or recall, and therefore we use *luke-base* as a relation detection module in the following experiments.

Timeout (s)	10	20	30	60	120	300	600	1200
Ours (EV)	79.5	79.7	79.6	80.1	80.4	80.3	80.4	80.4
Ours (MC)	82.0	82.1	82.7	82.9	83.4	**83.6**	**83.6**	**83.6**
Baseline	80.4							

Table 5.4 – Accuracy for relation classification, compared to the baseline.

5.4.2 Relation Classification

We now evaluate our Concepts of Neighbors-based relation classification module individually on the Relation Classification task.

Experiment Design These experiments are made on the positive examples of TACRED, *i.e.*, the examples that have an annotation other than *no_relation*. As our method does not have any use of a development corpus, we merge this corpus with the training one. We finally obtain a dataset composed of 18,446 training examples and 3,325 test examples. The quality measure usually used on TACRED is the micro-averaged F-score. However, as there is no negative class on this task, this measure does not make sense, and therefore we use accuracy.

In these experiments, as we work on a subset of TACRED we cannot compare this approach directly to other existing methods. Therefore, we compare our approach to a basic baseline in the RECENT ([LC21], see Section 5.3.1) paradigm. This baseline simply predicts, for given subject and object types, the relation type that appears the most often amongst the training examples with the same subject and object types. Therefore, scoring over this baseline implies that useful knowledge based on the semantics of the sentences has been used to make predictions.

As the algorithm for the computation of Concepts of Neighbors is *anytime*, we have to choose a timeout for our experiments. In order to see how the timeout influences the classification task, several timeouts were tested between 10 and 1200 seconds. In addition, the dependency tree pruning strategy (see Section 5.3.1) was used with $K = 1$. Finally, experiments were run using both the MC and the EV scoring method.

Our approach was implemented in Java [4] and uses the *CONNOR* library ([ACF22b], see Section 4.3.3) for the computation of Concepts of Neighbors.

4. Accessible here: `https://gitlab.inria.fr/hayats/conceptualknn-relex`

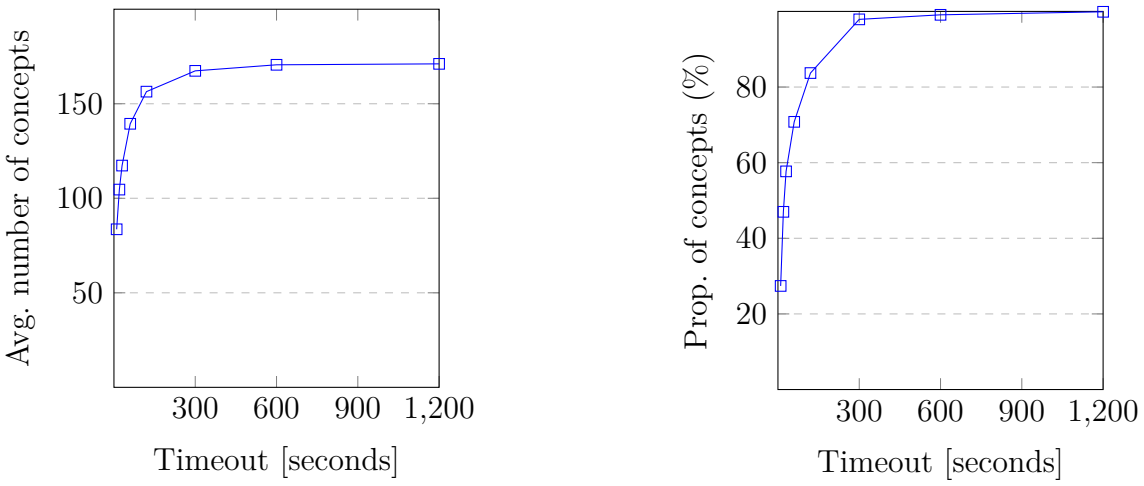

Figure 5.4 – Average number of computed (pre-)concepts per example and proportion of examples having a fully computed sets of Concepts of Neighbors

Results Table 5.4 presents the accuracy for the baseline and for our approach. First, it can be observed that the baseline has an accuracy of 80.4%, which is particularly high, which means that the dataset leaves little space for progress. Second, it shows that our approach with the EV scoring method shows negative result, hardly competing with the baseline for timeouts over 600 seconds. Then, it can be read that for any timeout, the proposed approach with the MC scoring method has a better accuracy than the baseline, surpassing it by 3.2 points for a timeout of over 300s.

In addition, this table clearly shows a saturation phenomenon: there is an important gain when timeout gets from 10s to 120s, a gain that is far smaller from 120s to 1200s. It can be intuited that this comes from the fact that most concepts are computed before 120 seconds, and only a few concepts are added after 120s. This is confirmed by Figure 5.4: the first graph shows that effectively most of the concepts are computed in less than 120s, and the second graph shows that, for a timeout of over 600s, for over 99% of the examples the full set of concepts of neighbors is computed, *i.e.*, the prediction is made from the exact set of concepts of neighbors, and not from an approximation of it.

In the light of the results, in the following, this relation classification method is used with the MC scoring method and a timeout of 300 seconds.

Method	F1 score
LUKE [Yam+20]	**72.7**
BERT-LSTM-Base [SL19]	67.8
Ours	*66.9*
C-GCN [ZQM18]	66.4
GCN [ZQM18]	64.0

Table 5.5 – F-score for several Relation Extraction methods on TACRED

5.4.3 Relation Extraction

Now that we have shown that our Concepts of Neighbors-based method is a valid approach for relation classification and that we have chosen a deep learning relation detection module, both can be assembled to form a full relation extraction method. In this subsection, we present the experimental process to evaluate this method, as well as both quantitative and qualitative results.

Experiment Design We evaluate our two-step approach on the full TACRED dataset in order to compare it to existing approaches. To do so, according to the structure presented in Figure 5.1, we process the test examples of TACRED with *luke-base*, and obtain examples classified as positive or negative. Then, each example classified as positive is processed by our Concept of Neighbors module for relation classification. Note that, as the relation detection step is made separately, the relation classification cannot predict the *no_relation* class.

Quantitative Results Table 5.5 compares our method with previous Relation Extraction methods. It shows that although our method is not competitive with pre-trained language models such as BERT or LUKE, it outperforms approaches based on graph convolution networks. Indeed, our method beats by 2.9 F-score points the basic graph convolution network (GCN) and by 0.5 points the contextualized graph convolution network (C-GCN). This is interesting because our method and those two methods are conceptually close: both are based on the representation of sentences as a graph, both use the pruned dependency tree of the sentences, and both add to this modeling a semantic layer (a word embedding for GCN and C-GCN, WordNet for our approach). The main benefit of our approach is its ability to provide explanations for the examples classified as positive.

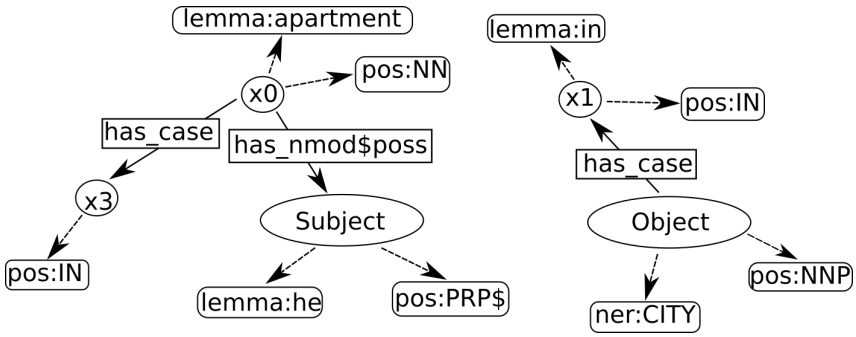

Figure 5.5 – Example of rule body

Qualitative Results To illustrate the explainability of our approach, let us take for example the sentence *"Sollecito has said he was at **his** own apartment in **Perugia**, working at his computer." luke-base* predicts that there is a relation between the subject (*his*) and the object (*Perugia*). As the subject is a person and the object a city, there are only three compatible relations: *per:cities_of_residence*, *per:city_of_death* or *per:city_of_birth*. After computation of the Concepts of Neighbors, we observe that the relation *per:city_of_residence* is predicted, as six rules of confidence 1 predict it, while only one rule each predicts the other two compatible relations. Figure 5.5 shows the body of one of those rules. It can be read as:

— The subject has lemma *he* and is the possessor of an apartment;

— The object is the name of a city in which there is something.

Even if this pattern is too specific to form a general rule, it can be inferred that, knowing there is a relation between the subject and the object, we can be pretty sure that any sentence following this pattern has the relation *per:cities_of_residence* between its subject and its object. To complete this explanation, we can look at the training examples matching this rule. In our case, there is one sentence matching it: *"Wilbert Gibson walked from **his** apartment to the grocery store earlier this week – that's what people do in **New York City** – and thought this must be what it's like to be a celebrity."* We can see that this sentence effectively expresses the relation *per:cities_of_residence*, but quite implicitly. Therefore, this is interesting to see that this kind of pattern can be captured and exploited by our approach.

In practice, we observe that the rules of maximum confidence have systematically a confidence equal to 1. This is due to the fact that Concepts of Neighbors compute rules specific enough to match a few cases, and therefore to have a low extensional distance. After reviewing the explanations for ten randomly chosen correct predictions, we can ob-

serve that 56% of the 172 graph patterns seem reliable. Most of those reliable explanations are considered as such because of a lemma or a synset appearing in the graph pattern (for example, the word *daughter* to characterize the relation *per:children*). In addition, we observe that the reliability of the explanations depends on the relation type. For example, it can be pointed out that for an example predicting the relation *per:top_member/employee*, most of the explanations are invalid. This is caused by the fact that there is a great variety of words or formulations expressing this relation, and therefore the same one is rarely used several times.

5.5 Conclusion

This chapter presents a two-step explainable approach for relation extraction. This approach is based on the division of the relation extraction task in two sub-tasks. The first one, relation detection, is addressed with a fine-tuned transformer-based LLM. The second one, relation classification, relies on the modeling presented in Chapter 3, and is addressed by an explainable, instance-based approach using the Concepts of Neighbors method. Both the relation detection and the relation classification modules are conceived to be integrated in the workflow presented in Chapter 3. This work has been evaluated on TACRED, a standard dataset for relation extraction. It has shown positive results on both the sub-tasks, and even if it cannot compete with state-of-the-art transformers-based approach on relation extraction, the performances are similar to graph convolution deep-learning approaches while presenting an explicit gain in terms of explainability.

Several aspects of the presented work need further research. First, concerning the relation detection module, as presented before the fine-tuning of LUKE has not been possible to reproduce, and an equivalent fine-tuning would probably have shown better performances. In addition, the domain of LLM is evolving rapidly, and new, more efficient models are published regularly. Concerning the relation classification module, it relies on both the graph modeling of texts presented in Chapter 3 and on the Concepts of Neighbors method presented in Chapter 4. Possible enhancement for those aspects are presented in those chapters. Concerning the experimental settings, the TACRED dataset has raised criticisms through the years, due to the low reliability of its annotations, especially for the harder cases, as presented in [AGH20]. In addition, TACRED is composed of examples extracted mainly from newspapers, written in literary English, and with a wide spectrum of themes. Therefore, this is quite far from the considered use-cases (described in Chap-

ter 3), with domain-specific texts composed of short, factual sentences. The creation of a new dataset (based on the WebNLG dataset [Gar+17], originally made for natural language generation) with domain-specific sub-datasets and more factual examples has been initiated during this thesis, but this has not been published yet.

Finally, the main aspect needing enhancement is the explainable aspect of this method. As for today, this method is explainable for an expert user, capable of interpreting confidence values and concept intensions. In order to make this method explainable for a non-expert user, and therefore allowing interaction as presented in Chapter 3, the current explanation (*i.e.*, the set of confidence values and the set of concepts of neighbors) has to be reduced and transformed into a concise explanation. Preliminary works are being made on this aspect, using a graph mining method to extract the significant patterns from the intensions of the concepts used for the prediction.

Chapter 6

Conclusion and Perspectives

This thesis is part of the current research in both the semantic web and the natural language processing fields, by attacking the problem of knowledge graph construction from text. This problem is crucial today, for several reasons. First, the mass of available text data is bigger than ever and virtually covers the whole human knowledge, but can be hard to access. Second, many semantic web technologies are now for querying, exploring, completing or editing knowledge graphs. Finally, diverse approaches exist for making a knowledge graph usable by an end-user. Therefore, transforming text in knowledge graph could allow for an easier process, completion, and access to diverse forms of knowledge, from generalist encyclopedic knowledge to domain-specific or context-specific knowledge. However, due to the ambiguity of language and of the task itself, knowledge graph construction from text is far from being trivial.

The approach proposed in this work has the specificity to be user-centered: instead of proposing an end-to-end system trained on pre-existing data, we present a system that learns by interacting with the user, and co-construct the knowledge graph with them. To do so, the system is designed according to a user-centered workflow, allowing for interaction during the whole process. The overall objective is to produce a resulting knowledge graph This workflow relies on different technologies and methods in terms of text representation, human-machine interaction, machine learning and inference systems, and some of them are studied in this thesis.

The first contribution of this thesis is the workflow model itself (Chapter 3). The workflow relies on an intermediate representation of text as a graph, representing both its semantic and syntactic aspects. This intermediate representation also embeds a pre-annotation to the elements that could appear in the resultant knowledge graph – this pre-annotation being made by a machine learning approach. Two units use this representation. The first is an interactive channel, based on a human-machine interface as well as an instance-based, explainable machine learning approach using Concepts of Neighbors, a technique issued from the Formal Concept Analysis (FCA) framework. This channel

aims to assist the user in the manual construction of the knowledge graph. The second is an automated channel that, based on the actions of the user through the interactive unit and through a rule-based inference system, extracts knowledge from the intermediate representation into the knowledge graph. The second contribution (Chapter 4), mainly theoretical, is a set of additions to the FCA framework, especially on the Concepts of Neighbors method, in order to adapt it for the desired task: relation extraction on graphs. These contributions consist into a new, more general formalization of the notion of concepts of neighbors, an adaptation of it for the n-ary case, and a method for handling RDF graphs in this framework. The last contribution (Chapter 5) is the conception and evaluation of a relation extraction system, composed of a deep learning relation detection module and a Concepts of Neighbors-based relation classification module. Both those modules are conceived to be compatible with the workflow – for the construction of the intermediate representation for the relation detection module, and for the interactive unit for the relation classification module. This method presents results comparable to pre-existing deep learning approach, while presenting advantages in terms of interpretability.

Perspectives

Several aspects of the contributions of this thesis require further research. Those different points are detailed below.

Explainability of the Concepts of Neighbors As of today, our relation classification method presented in Chapter 5 is explainable for an expert user: the classification results from a scoring made on the set of concepts of neighbors of a given individual, and an expert can interpret the score and the intensions and extensions of the concepts to comprehend and verify the explanation. However, a non-expert end user cannot understand easily such an explanation, nor validate it. In addition, as presented in Chapter 3, the explanation should take the form of a rule applicable by the inference system on the intermediate modeling. Preliminary works exploring the idea of using a graph mining method (namely GraphMDL+ [BCF21]) on the intensions of the concepts used in the prediction in order to produce such rules have been conducted, and have shown promising results.

New Relation Extraction Dataset TACRED [Zha+17], the standard dataset in Relation Extraction used in this thesis, which is a widely used benchmark in the domain, can

be criticized. First, from a general point of view, as shown in [AGH20], the annotations of TACRED, obtained by crowdsourcing, are of poor quality, especially for the hardest individuals. In addition, the data of TACRED itself is noisy: it should contain annotated plain sentences in English, mainly extracted from newspapers, but some of those sentences either are malformed (e.g., by merging the headline of an article and its first sentence) or contains artifacts (such as metadata of newspaper article or special characters replaced by strings – e.g., `"-RRB-"` replacing `"("`). Finally, this dataset contains longs, complex sentences written in a literary style, which is quite far from our use cases on factual texts. Therefore, we decided to create a new relation extraction dataset. This dataset is based on WebNLG [Gar+17], a dataset for text generation from RDF triples already adapted to the reverse task (RDF triples generation from text). The sentences are shorter, factual, and domain-centric, and therefore should be more adapted to our needs. The main task for transforming this dataset into a relation extraction dataset is entity alignment on text, which is not trivial and asks for more development work before evaluation and publication.

Human-Machine Interface The user interface, core of the interactive channel of the workflow, should have several properties. First, it should be an edition tool for the produced knowledge graph. Second, it should map the text (or its intermediate representation) with the knowledge graph. In addition, it should present and permit validation of the suggestions and explanations produced by the relation extraction module. In other terms, even if technically part of the interactive unit, this interface has to interact with modules and bases in the whole system. Initially, a basic command line interface could be developed for experimenting with interactions scenarios, before developing a graphical interface.

Automated Unit This unit is constituted of two elements: a rule base and an inference system. A basic implementation would be using a plain set of rules and a query system for the inference. For example, the rules could be formulated as SPARQL queries and the inference system could be a basic query engine applying those queries on the intermediate representation. Then, the system could be enriched by adding a confidence values on the rules, confidence that could evolve through user interactions.

Extension of the Pattern Language of Projected Graph Patterns In the presented Graph-FCA theory, the pattern language of PGP is quite restrictive, as it does not

allow unlabeled hyper-edges or paths of unspecified length. This pattern language could be extended in order to handle more complex graph patterns, on the model of regular expressions. This would impact the PGP inclusion definition, as well as the combinatorics of the graph concepts. In particular, for what concerns the concepts of neighbors, this could have an important impact on the algorithmic complexity, that should be studied.

Algorithmic Enhancements for the Computation of Concepts of Neighbors In a more theoretical and algorithmic side, several aspects of the algorithmic framework of the Concepts of Neighbors could be developed. The point of these enhancements is to reduce the computation time of Concepts of Neighbors, in order to grant interactivity even while working on important datasets.

First, as no strategy for choosing an incidence in the partitioning algorithm is specified, the current implementations use a basic *first in, first out* strategy. However, there is no reason to believe this strategy is optimal, neither in terms of computation time nor in terms of relevance of the explored concepts. Therefore, diverse strategies, based on the structure of the data and the already processed incidences, could be explored. In addition, it has been experimentally observed that the choice of some specific incidences provokes an explosion of the computation time due to a combinatorial problem. Those problematic cases could be characterized, and specific strategies to avoid them could be conceived.

The *lazy join* algorithm is also subject to enhancement. A post-insertion downward propagation algorithm has been developed in order to reduce the size of the match-sets – and by consequence of the match-trees – and therefore have better performance in terms of memory and time. However, if this enhancement has shown positive results in the case of unary concepts of neighbors, further work is needed to adapt it to the n-ary case, due to the potentially disjoint nature of the intensions. Some strategies have already been considered, but this needs further research before publication.

References

[AGH20] Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig, « TACRED Revisited: A Thorough Evaluation of the TACRED Relation Extraction Task », *in*: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 1558–1569.

[Agr+94] Rakesh Agrawal et al., « Fast Algorithms for Mining Association Rules », en, *in*: *Proc. of the 20th International Conference on Very Large Databases*, 1994, pp. 487–499.

[AHH19] Christoph Alt, Marc Hübner, and Leonhard Hennig, *Improving Relation Extraction by Pre-trained Language Representations*, June 2019, URL: http://arxiv.org/abs/1906.03088.

[Aue+07] Sören Auer et al., « DBpedia: A Nucleus for a Web of Open Data », *in*: *Asian Semantic Web Conference (ASWC)*, 2007, pp. 722–735, DOI: 10.1007/978-3-540-76298-0_52.

[Ban+13] Laura Banarescu et al., « Abstract Meaning Representation for Sembanking », en, *in*: *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, 2013, pp. 178–186.

[BCF21] Francesco Bariatti, Peggy Cellier, and Sébastien Ferré, « GraphMDL+: interleaving the generation and MDL-based selection of graph patterns », *in*: *Annual ACM Symposium on Applied Computing*, 2021, pp. 355–363.

[Bec+23] David Beckett et al., *RDF 1.2 Turtle*, Oct. 2023, URL: https://www.w3.org/TR/rdf12-turtle/ (visited on 10/12/2023).

[BFM05] Tim Berners-Lee, Roy T. Fielding, and Larry M. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, Request for Comments RFC 3986, Num Pages: 61, Internet Engineering Task Force, Jan. 2005, DOI: 10.17487/RFC3986, URL: https://datatracker.ietf.org/doc/rfc3986 (visited on 10/11/2023).

[BH21] Nadia Burkart and Marco F. Huber, « A Survey on the Explainability of Supervised Machine Learning », en, *in*: *Journal of Artificial Intelligence Research* 70 (2021), pp. 245–317.

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila, « The Semantic Web », *in*: *Scientific American* 284.*5* (2001), pp. 34–43.

[BKO21] Saurabh Bansal, Sriram Kailasam, and Sergei Obiedkov, « Approximate Computation of Exact Association Rules », *in*: *International Conference on Formal Concept Analysis*, ed. by Agnès Braud et al., 2021, pp. 107–122.

[Bro+20] Tom B. Brown et al., *Language Models are Few-Shot Learners*, arXiv:2005.14165 [cs], July 2020.

[BZ11] Asma Ben Abacha and Pierre Zweigenbaum, « Automatic extraction of semantic relations between medical entities: a rule based approach », *in*: *Journal of Biomedical Semantics* 2.*5* (2011), pp. 1–11.

[Cal+12] Diego Calvanese et al., *OWL 2 Web Ontology Language Profiles (Second Edition)*, Dec. 2012, URL: `https://www.w3.org/TR/owl2-profiles/`.

[Cel+15] Peggy Cellier et al., « Sequential pattern mining for discovering gene interactions and their contextual information from biomedical texts », *in*: *Journal of Biomedical Semantics* 6.*1* (2015), p. 27.

[CLN14] Victor Codocedo, Ioanna Lykourentzou, and Amedeo Napoli, « A semantic approach to concept lattice-based information retrieval », *in*: *Annals of Mathematics and Artificial Intelligence* 72.*1* (2014), pp. 169–195.

[CN16] Jason P.C. Chiu and Eric Nichols, « Named Entity Recognition with Bidirectional LSTM-CNNs », *in*: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 357–370.

[Cod70] E. F. Codd, « A relational model of data for large shared data banks », *in*: *Communications of the ACM* 13.*6* (1970), pp. 377–387.

[Coo60] James Cooke Brown, « Loglan », *in*: *Scientific American* 202.*06* (1960), pp. 53–63.

[Dev+19] Jacob Devlin et al., « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding », en, *in*: *Proceedings of NAACL-HLT 2019* (2019), pp. 4171–4186.

[Fan+16] Wei Fang et al., « Entity Disambiguation by Knowledge and Text Jointly Embedding », *in*: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2016, pp. 260–269.

[FC20] Sébastien Ferré and Peggy Cellier, « Graph-FCA: An extension of formal concept analysis to knowledge graphs », *in*: *Discrete Applied Mathematics* 273 (2020), pp. 81–102.

[Fer15] Sébastien Ferré, « A Proposal for Extending Formal Concept Analysis to Knowledge Graphs », *in*: *International Conference on Formal Context Analysis*, 2015, pp. 271–286.

[Fer17a] Sébastien Ferré, « Concepts de plus proches voisins dans des graphes de connaissances. », *in*: *Actes IC 2017 28es Journées francophones d'Ingénierie des Connaissances*, 2017, pp. 163–174.

[Fer17b] Sébastien Ferré, « SPARKLIS: An Expressive Query Builder for SPARQL Endpoints with Guidance in Natural Language », en, *in*: *Open Journal Of Semantic Web 0* (2017), p. 16.

[Fer18] Sébastien Ferré, « Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search », en, *in*: *The Semantic Web*, 2018, pp. 209–224.

[Fer20] Sébastien Ferré, « Application of Concepts of Neighbours to Knowledge Graph Completion », *in*: *Data Science* Pre-press.*Pre-press* (2020), pp. 1–28.

[FKZ07] Katrin Fundel, Robert Küffner, and Ralf Zimmer, « Relation extraction using dependency parse trees », *in*: *Bioinformatics* 23.*3* (Feb. 2007), pp. 365–371.

[Fou+20] Philippe Fournier-Viger et al., « A survey of pattern mining in dynamic graphs », en, *in*: *WIREs Data Mining and Knowledge Discovery* 10.*6* (2020), DOI: 10.1002/widm.1372.

[FR00] Sébastien Ferré and Olivier Ridoux, « A Logical Generalization of Formal Concept Analysis », en, *in*: *International Conference on Conceptual Structures*, 2000, pp. 371–384.

[FR02] Sébastien Ferré and Olivier Ridoux, « The Use of Associative Concepts in the Incremental Building of a Logical Context », en, *in*: *Conceptual Structures: Integration and Interfaces*, Lecture Notes in Computer Science, Springer, 2002, pp. 299–313.

[Gal+13] Luis Antonio Galárraga et al., « AMIE: association rule mining under incomplete evidence in ontological knowledge bases », *in*: *Proc. of the 22nd int. conf. on World Wide Web - WWW '13*, 2013, pp. 413–422, DOI: 10.1145/2488388.2488425.

[Gal+15] Luis Galárraga et al., « Fast rule mining in ontological knowledge bases with AMIE+ », en, *in*: *The VLDB Journal* 24.*6* (2015), pp. 707–730.

[Gal22] Esther Galbrun, « The minimum description length principle for pattern mining: a survey », *in*: *Data Mining and Knowledge Discovery* 36.*5* (2022), pp. 1679–1727.

[Gar+17] Claire Gardent et al., « Creating Training Corpora for NLG Micro-Planning », *in*: *Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 179–188.

[GK01] Bernhard Ganter and Sergei O. Kuznetsov, « Pattern Structures and Their Projections », *in*: *International Conference on Conceptual Structures*, 2001, pp. 129–142.

[GLR06] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano, « Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature », *in*: *European Chapter of the Association for Computational Linguistics*, 2006, pp. 401–408.

[GS21] Claudio Gutierrez and Juan F. Sequeda, « Knowledge graphs », *in*: *Communications of the ACM* 64.*3* (2021), pp. 96–104.

[GW99] Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis: Mathematical Foundations*, en, Springer, 1999.

[HFD12] Alice Hermann, Sébastien Ferré, and Mireille Ducassé, « An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs », en, *in*: *International Conference on Knowledge Engineering and Knowledge Management*, vol. 7603, 2012, pp. 185–199, (visited on 02/17/2021).

[Hit21] Pascal Hitzler, « A Review of the Semantic Web Field », en, *in*: *Communications of the ACM* 64.*2* (2021), pp. 76–83.

[HKR09] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph, *Foundations of Semantic Web Technologies*, en, Chapman and Hall/CRC, 2009.

[HM17] Matthew Honnibal and Ines Montani, « spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing », 2017.

[Hua+04] Jun Huan et al., « SPIN: mining maximal frequent subgraphs from graph databases », *in*: *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 581–586.

[HWB01] Tamás Horváth, Stefan Wrobel, and Uta Bohnebeck, « Relational Instance-Based Learning with Lists and Terms », *in*: *Machine Learning* 43.*1* (2001), pp. 53–80.

[HWP03] J. Huan, W. Wang, and J. Prins, « Efficient mining of frequent subgraphs in the presence of isomorphism », *in*: *IEEE International Conference on Data Mining*, 2003, pp. 549–552.

[IWM00] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda, « An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data », en, *in*: *Principles of Data Mining and Knowledge Discovery*, 2000, pp. 13–23.

[JCZ13] Chuntao Jiang, Frans Coenen, and Michele Zito, « A survey of frequent subgraph mining algorithms », en, *in*: *The Knowledge Engineering Review* 28.*1* (Mar. 2013), pp. 75–105, DOI: 10.1017/S0269888912000331.

[Jos+20] Mandar Joshi et al., « SpanBERT: Improving Pre-training by Representing and Predicting Spans », *in*: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 64–77.

[Kuz04] Sergei O. Kuznetsov, « Machine Learning and Formal Concept Analysis », *in*: *International Conference on Formal Concept Analysis*, 2004, pp. 287–312.

[Kuz13a] Sergei O. Kuznetsov, « Fitting Pattern Structures to Knowledge Discovery in Big Data », *in*: *International Conference on Formal Concept Analysis*, 2013, pp. 254–266.

[Kuz13b] Sergei O. Kuznetsov, « Scalable Knowledge Discovery in Complex Data with Pattern Structures », *in*: *Pattern Recognition and Machine Intelligence*, 2013, pp. 30–39.

[Lam+16] Guillaume Lample et al., « Neural Architectures for Named Entity Recognition », *in*: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 260–270.

[LC21] Shengfei Lyu and Huanhuan Chen, « Relation Classification with Entity Type Restriction », *in*: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 390–395.

[Lee+15] Artuur Leeuwenberg et al., « Exploring Pattern Structures of Syntactic Trees for Relation Extraction », en, *in*: *International Conference on Formal Concept Analysis*, vol. 9113, 2015, pp. 153–168.

[Lee+16] Matthijs van Leeuwen et al., « Subjective interestingness of subgraph patterns », *in*: *Machine Learning* 105.*1* (Oct. 2016), pp. 41–75.

[Lee14] Matthijs van Leeuwen, « Interactive Data Exploration Using Pattern Mining », *in*: *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics: State-of-the-Art and Future Challenges*, Lecture Notes in Computer Science, 2014, pp. 169–182.

[LGS20] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek, « Fast and Exact Rule Mining with AMIE 3 », en, *in*: *European Semantic Web Conference*, Lecture Notes in Computer Science, 2020, pp. 36–52.

[Liu+20] Weijie Liu et al., « K-BERT: Enabling Language Representation with Knowledge Graph », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, Number: 03, 2020, pp. 2901–2908.

[LS99] Ora Lassila and Ralph R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, 1999, URL: `https://www.w3.org/TR/PR-%20rdf-syntax/Overview.html` (visited on 09/08/2023).

[LT18] Phong Le and Ivan Titov, « Improving Entity Linking by Modeling Latent Relations between Mentions », *in*: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp. 1595–1604.

[Mal+21] Cyrielle Mallart et al., « Active Learning for Interactive Relation Extraction in a French Newspaper's Articles », *in*: *Recent Advances in Natural Language Processing*, 2021, pp. 886–894.

[Man+14] Christopher D. Manning et al., « The Stanford CoreNLP Natural Language Processing Toolkit », *in*: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.

[Mei+19] Christian Meilicke et al., « Anytime Bottom-Up Rule Learning for Knowledge Graph Completion », en, *in*: *International Joint Conference on Artificial Intelligence*, 2019, pp. 3137–3143.

[MHL20] Jose L. Martinez-Rodriguez, Aidan Hogan, and Ivan Lopez-Arevalo, « Information extraction meets the Semantic Web: A survey », en, *in*: *The Semantic Web* 11.*2* (2020), ed. by Andreas Hotho, pp. 255–335.

[Mil98] George A. Miller, *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press, 1998.

[MLR18] Jose L. Martinez-Rodriguez, Ivan Lopez-Arevalo, and Ana B. Rios-Alvarado, « OpenIE-based approach for Knowledge Graph construction from text », en, *in*: *Expert Systems with Applications* 113 (2018), pp. 339–355.

[Mus15] Mark A. Musen, « The Protégé Project: A Look Back and a Look Forward », *in*: *AI matters* 1.*4* (June 2015), pp. 4–12.

[NC03] Nick Nicholas and John Woldemar Cowan, eds., *What is Lojban? =: i la lojban. mo*, en, La Verne, Tenn: Logical Language Group, 2003, ISBN: 978-0-9660283-1-7.

[NG15] Thien Huu Nguyen and Ralph Grishman, « Relation Extraction: Perspective from Convolutional Neural Networks », en, *in*: *Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 39–48, (visited on 12/09/2019).

[PSW00] R. Parasuraman, T.B. Sheridan, and C.D. Wickens, « A model for types and levels of human interaction with automation », *in*: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 30.*3* (2000), pp. 286–297.

[PWS20] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek, « YAGO 4: A Reason-able Knowledge Base », *in*: *The Semantic Web*, 2020, pp. 583–596.

[Ric56] R H Richens, « Preprogramming for mechanical translation », en, *in*: *Mechanical Translation* 3.*1* (1956), pp. 20–25.

[RNS20] Reza Ramezani, Mohammad Ali Nematbakhsh, and Mohamad Saraee, « Mining Association Rules from Semantic Web Data without User Intervention », *in*: *Journal of Computing and Security* 7.*1* (2020), pp. 81–94.

[Rou+13] Mohamed Rouane-Hacene et al., « Relational concept analysis: mining concept lattices from multi-relational data », *in*: *Annals of Mathematics and Artificial Intelligence* 67.*1* (2013), pp. 81–108.

[Rud19] Cynthia Rudin, « Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead », *in*: *Nature Machine Intelligence* 1.*5* (2019), pp. 206–215.

[Shn20] Ben Shneiderman, « Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy », *in*: *International Journal of Human–Computer Interaction* 36.*6* (2020), pp. 495–504.

[SL19] Peng Shi and Jimmy Lin, *Simple BERT Models for Relation Extraction and Semantic Role Labeling*, 2019.

[SL22] Tangina Sultana and Young-Koo Lee, « Efficient rule mining and compression for RDF style KB based on Horn rules », en, *in*: *The Journal of Supercomputing* 78.*14* (2022), pp. 16553–16580, DOI: 10.1007/s11227-022-04519-y.

[Suk+20] Rhea Sukthanker et al., « Anaphora and coreference resolution: A review », *in*: *Information Fusion* 59 (2020), pp. 139–162.

[Sun+20] Yu Sun et al., « ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, Number: 05, 2020, pp. 8968–8975.

[Swa04] Aaaron Swartz, *application/rdf+xml Media Type Registration*, Request for Comments RFC 3870, Internet Engineering Task Force, Sept. 2004, URL: https://datatracker.ietf.org/doc/rfc3870 (visited on 10/12/2023).

[Tou+03] Kristina Toutanova et al., « Feature-rich part-of-speech tagging with a cyclic dependency network », *in*: *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, vol. 1, 2003, pp. 173–180.

[Tur50] A. M. Turing, « Computiong Machinery and Intelligence », *in*: *Mind* 59.*236* (1950), pp. 433–460.

[VK14] Denny Vrandečić and Markus Krötzsch, « Wikidata: a free collaborative knowledgebase », *in*: *Communications of the ACM* 57.*10* (2014), pp. 78–85.

[W3C17] W3C, *Shapes Constraint Language (SHACL)*, en, July 2017, URL: `https://www.w3.org/TR/shacl/` (visited on 10/13/2023).

[Wan+22] Chenguang Wang et al., « DeepStruct: Pretraining of Language Models for Structure Prediction », *in*: *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 803–823.

[Wil82] Rudolf Wille, « Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts », *in*: *Symposium on Ordered Set*, 1982, pp. 445–470.

[Wit53] Ludwig Wittgenstein, *Philosophical Investigations*, Fourth Edition, Wiley-Blackwell, 1953, ISBN: 978-1-4051-5928-9.

[WZ19] Felix Wu and Tianyi Zhang, « Simplifying Graph Convolutional Networks », en, *in*: *International Conference on Machine Learning* (2019), p. 11.

[Xu+15] Yan Xu et al., « Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths », en, *in*: *Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1785–1794.

[Yam+20] Ikuya Yamada et al., « LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention », *in*: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6442–6454.

[Yan+08] Kyoung-Mo Yang et al., « Fuzzy Concept Mining based on Formal Concept Analysis », *in*: *International Journal of Computers* 2.*3* (2008).

[YH02] Xifeng Yan and Jiawei Han, « gSpan: graph-based substructure pattern mining », *in*: *IEEE International Conference on Data Mining*, 2002, pp. 721–724.

[YH03] Xifeng Yan and Jiawei Han, « CloseGraph: mining closed frequent graph patterns », *in*: *ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug. 2003, pp. 286–295.

[Zha+17] Yuhao Zhang et al., « Position-aware Attention and Supervised Data Improve Slot Filling », *in*: *Conference on Empirical Methods in Natural Language Processing*, 2017.

[Zha+23] Zhuo Zhang et al., « Robot path planning based on concept lattice », *in*: *International Journal of Approximate Reasoning* 153 (2023), pp. 87–103.

[Zhu+07] Feida Zhu et al., « gPrune: A Constraint Pushing Framework for Graph Pattern Mining », en, *in*: *Advances in Knowledge Discovery and Data Mining*, 2007, pp. 388–400.

[ZQM18] Yuhao Zhang, Peng Qi, and Christopher D. Manning, « Graph Convolution over Pruned Dependency Trees Improves Relation Extraction », *in*: *Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2205–2215.

Publications

[ACF21] H. Ambre Ayats, Peggy Cellier, and Sébastien Ferré, « Extracting Relations in Texts with Concepts of Neighbours », *in*: *International Conference on Formal Concept Analysis*, 2021, pp. 155–171, DOI: https://doi.org/10.1007/978-3-030-77867-5_10.

[ACF22a] H. Ambre Ayats, Peggy Cellier, and Sébastien Ferré, « A Two-Step Approach for Explainable Relation Extraction », en, *in*: *Advances in Intelligent Data Analysis*, 2022, pp. 14–25, DOI: https://doi.org/10.1007/978-3-031-01333-1_2.

[ACF22b] H. Ambre Ayats, Peggy Cellier, and Sébastien Ferré, « CONNOR: Exploring Similarities in Graphs with Concepts of Neighbors », *in*: *Existing Tools and Applications for Formal Concept Analysis (ETAFCA)*, 2022, pp. 219–224.

[ACF24] H. Ambre Ayats, Peggy Cellier, and Sébastien Ferré, « Concepts of neighbors and their application to instance-based learning on relational data », *in*: *International Journal of Approximate Reasoning* 164 (2024), p. 109059, DOI: 10.1016/j.ijar.2023.109059.

[Aya22] H. Ambre Ayats, « Construction de Graphes de Connaissance à partir de textes avec une I.A. centrée-utilisateur », *in*: *Rencontres Etudiants Chercheurs en Informatique pour le TAL (RECITAL)*, 2022, pp. 33–46.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Construction de graphes de connaissances à partir de textes avec une intelligence artificielle explicable et centrée-utilisateur·ice

Mot clés : Traitement automatique du langage naturel, explicabilité, IA centrée-utilisateur·ice, analyse de concepts formels, web sémantique, graphes de connaissances

Résumé : Avec les progrès récents dans le domaine de l'intelligence artificielle, la question du contrôle humain est devenu centrale. Aujourd'hui, cela passe à la fois par des recherches en explicabilité et des systèmes centrés autour de l'interaction avec l'utilisateur·ice. De plus, avec l'expansion du web sémantique et des méthodes de traitement automatique du langage naturelle, la tâche de construction de graphes de connaissances à partir de textes est devenu un enjeu important. Cette thèse présente un système centré-utilisateur·ice pour la construction de graphes de connaissances à partir de textes. Cette thèse présente plusieurs contributions. Tout d'abord, nous introduisons un workflow centré-utilisateur·ice pour la tâche sus-citée, ayant la propriété d'automatiser progressivement les actions de l'utilisateur·ice tout en lui laissant un contrôle fin du résultat. Ensuite, nous présentons nos apports dans le domaine de l'analyse de concepts formels, utilisés afin de concevoir un module d'apprentissage fainéant et explicable pour la tâche de classification de relations. Enfin, nous présentons nos apports dans le domaine de l'extraction de relations, et comment ces apports s'inscrivent dans le workflow présenté précédemment.

Title: Knowledge Graph Construction from Text with an Explanable, Human-Centered Artificial Intelligence

Keywords: Natural Language Processing, Explainability, User-Centric AI, Formal Concept Analysis, Semantic Web, Knowledge Graphs

Abstract: With recent advances in artificial intelligence, the question of human control has become central. Today, this involves both research into explainability and designs centered around interaction with the user. What's more, with the expansion of the semantic web and automatic natural language processing methods, the task of constructing knowledge graphs from texts has become an important issue. This thesis presents a user-centered system for the construction of knowledge graphs from texts. This thesis presents several contributions. First, we introduce a user-centered workflow for the aforementioned task, having the property of progressively automating the user's actions while leaving them a fine-grained control over the outcome. Next, we present our contributions in the field of formal concept analysis, used to design an explainable instance-based learning module for relation classification. Finally, we present our contributions in the field of relation extraction, and how these fit into the presented workflow.