Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Génération automatique et non supervisée d'ontologies à partir de corpus spécialisés

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Ingénierie Multilingue

par

Liza FRETEL

Directeur de mémoire :

Damien Nouvel

Encadrant :

Kévin Gravouil

Année universitaire 2023/2024

Table des matières

Liste des figures

Liste des tableaux

Remerciements

Je tiens à remercier mon directeur de mémoire, monsieur Damien Nouvel, pour son soutien et ses conseils tout au long de la rédaction de ce mémoire.

Merci également au corps enseignant du Master PluriTAL pour tout ce qu'ils m'ont apporté sur le plan personnel et professionnel, ainsi qu'à mes camarades de classe pour leur vitalité et leur bonne humeur à toute épreuve qui m'ont accompagnés durant ces deux dernières années.

Je remercie chaleureusement mon encadrant au sein de l'entreprise Airudit, monsieur Kévin Gravouil, pour tous les échanges enrichissants qui m'ont permis d'aborder mon sujet avec une base solide et d'avancer au fur et à mesure que les idées émergeaient.

Un grand merci à Philippe Lebas de m'avoir offert cette belle opportunité et aux membres de l'équipe Recherche et Développement d'Airudit, particulièrement à Angélique Burault et Anastasia Larionova pour leur intérêt porté à mes recherches ainsi qu'à tous mes collègues pour leur dévotion collective à mon intégration en entreprise, mention spéciale pour tous ces bons moments passés ensemble lors de nos parties de tarot endiablées.

Et enfin, je voue une reconnaissance intemporelle envers ma famille pour avoir cru en moi et m'avoir encouragée jusqu'à ce jour et plus encore.

RÉSUMÉ

Les ontologies jouent un rôle clé dans la structuration et la formalisation des concepts. En effet, elles ont la capacité de représenter comment un concept intéragit avec les autres, offrant ainsi un moyen de raisonnement plus proche du nôtre à la machine. Par ailleurs, les ontologies octroient la capacité de synthétiser et de partager des connaissances provenant de multiples ressources grâce au référencement inter-ontologique. Elles viennent combler les lacunes des LLMs (*Large Language Models*), qui peinent parfois à distinguer le vrai du faux [1]. De plus, les LLMs, bien qu'entraînés sur des données très diverses, ne sont pas omniscients, les rendant inefficaces dans un contexte très spécifique [2].

L'inconvénient majeur des ontologies, cependant, est que leur construction demande une certaine expertise et une charge cognitive élevée. Afin d'automatiser leur création, nous pouvons utiliser des corpus de spécialité et des techniques de traitement automatique des langues. Ce processus, nommé « apprentissage d'ontologies » (*Ontology Learning*), se divise en plusieurs sous-tâches, telles que l'extraction d'informations, la modélisation de ces informations au sein d'un graphe de connaissances, la sélection et l'organisation de concepts au sein d'une taxonomie, etc.

Quatre étapes sont abordées dans ce mémoire. La première étape est l'extraction et l'organisation de triplets [3] en graphes de connaissances à partir d'arbres syntaxiques. L'extraction obtient une F-mesure de 0,812 sur le corpus Solaris en exploitant la sortie du modèle de langue SpaCy [Honnibal and Montani, 2017]. Ensuite, pour réaliser la sélection de concepts, nous nous sommes appuyés sur des mesures statistiques comme le score de spécificité. L'organisation des concepts sous une taxonomie exploite les définitions présentes dans le corpus suivant la formulation « A est un B ». Pour le référencement d'ontologies, nous utilisons un modèle XNLI [4]. Le référencement de notre ontologie Catastrophes à Wikidata atteint un hit@1 à 0,53, prouvant que la vectorisation de concepts par XNLI est une méthode efficace pour référencer une ontologie de domaine à une ontologie généraliste.

Mots-clés : Ontology Learning, corpus de domaine, manipulation d'arbres syntaxiques, construction de graphes de connaissances, hiérarchisation de concepts.

1. Ces problèmes sont souvent liés aux hallucinations des LLMs, voir [Zhang et al., 2023], et au manque de factualité des informations données.
2. Voir également [Kandpal et al., 2023].
3. Un triplet est composé de deux entités et d'une relation entre elles.
4. Modèle NLI (*Natural Language Inference*) multilingue, plus précisément, le modèle mDeBERTa-v3-base-xnli-multilingual-nli-2mil7.

INTRODUCTION

Présentation générale

Les ontologies apportent une modélisation des connaissances et une capacité de conceptualisation et de raisonnement que les bases de données relationnelles ne possèdent pas. Voici quatre définitions d'ontologie tirées de CNRTL [5] :

1. « Partie de la philosophie qui a pour objet l'étude des propriétés les plus générales de l'être, telles que l'existence, la possibilité, la durée, le devenir. »

2. « Étude des êtres en eux-mêmes et non tels qu'ils nous apparaissent » (d'apr. Foulq.-St-Jean 1962).

3. « Partie de la philosophie qui a pour objet l'élucidation du sens de l'être considéré simultanément en tant qu'être général, abstrait, essentiel et en tant qu'être singulier, concret, existentiel. »

4. « Théorie sur l'être ; ensemble de vérités fondamentales de l'être. »

La première définition nous apprend qu'une ontologie regroupe des propriétés diverses qui constituent l'univers, en ancrant toute chose dans notre espace-temps. Cette définition fait référence à une ontologie de niveau supérieur (*top-level ontology* [6] telle que BFO [7] ou DOLCE [8], qui ont pour objectif de formaliser les propriétés des choses de façon essentiellement philosophique. La seconde définition nous apprend qu'une ontologie est objective et non subjective. La troisième, qu'une ontologie peut contenir à la fois des concepts (abstrait) et des individus (concret), par exemple le concept d'être humain dont les individus, ou instances, sont des personnes en particulier, avec un nom, une date et un lieu de naissance, deux parents qui sont aussi des instances du concept d'être humain, etc. La quatrième définition nous apprend que l'ontologie théorise l'être et formalise des vérités fondamentales, donc qu'aucune contradiction ou information fausse ne peut y figurer. Ce que ces définitions ne nous disent pas, c'est ce qu'est une ontologie du point de vue de l'ordinateur. Nous espérons que ce mémoire aidera le lecteur à y voir plus clair, mais en voici une première définition, élaborée par Tom Gruber en 1993 :

> « Une ontologie est la spécification d'une conceptualisation. [...] Une conceptualisation est une vue abstraite et simplifiée du monde que l'on veut représenter dans un certain but. » [Gruber, 1993]

Si l'on s'intéresse aux formats des données incluses dans une structure ontologique, il s'agit d'un ensemble de triplets de type <sujet, prédicat, objet>, parmi lesquels certains servent à donner des propriétés aux entités de l'ontologie ou à les faire

5. https://www.cnrtl.fr/definition/ontologie
6.)
7. Basic Formal Ontology, voir [Otte et al., 2022]
8. Descriptive Ontology for Linguistic and Cognitive Engineering, voir [Borgo et al., 2022]

intéragir entre elles (relations entre entités), d'autres servent à créer une taxonomie de concepts ou à instancier ces concepts [9], et enfin les derniers définissent des axiomes, ou des règles qui limitent l'usage des concepts et permettent une certaine logique et un raisonnement sur l'ontologie. Enfin, les ontologies, qui sont symboliques et discrètes, apportent des solutions aux probèmes non encore résolus par les LLMs, numériques et probabilistes, en voulant garantir la véracité et la non-contradiction des informations, et en apportant une dimension spatio-temporelle et hiérarchique des choses.

La nécessité des ontologies augmente depuis l'apparition des LLMs (Large Language Model), ces réseaux de neurones capables d'emmagasiner une quantité colossale de connaissances grâce à des milliards de paramètres. De nombreuses entreprises leur font confiance pour créer des agents conversationnels personnalisés. Étant entraînés sur des données générales, un des défauts des LLMs est leur manque de compréhension d'un domaine spécifique [Kandpal et al., 2023]. Une solution alternative à ce problème est de les ré-entraîner sur des données de spécialité, afin de leur apprendre les notions manquantes. Cependant, malgré ce ré-entraînement, ils ne sont pas à l'abri de confondre leurs nouvelles connaisances avec d'anciennes connaissances, faisant surgir un phénomène d'hallucinations [Zhang et al., 2023]. La deuxième solution consiste à guider la réponse du LLM avec un prompt qui contient à la fois la question et des documents dans lesquels il peut trouver la réponse et les formuler de façon à répondre à la question. Ce type de système s'appelle un RAG (*Retrieval Augmented Generation* [Gao et al., 2024]). Néanmoins, il se peut que la réponse n'existe pas dans les documents renvoyés par une requête, dans quel cas le LLM en inventera une à partir de ses connaissances et des informations similaires mais fausses issues des documents fournis par la requête. La troisième solution, est d'alimenter le système de questions-réponses avec une ontologie, qui représente des connaissances de manière formelle et guarantit leur véracité, et d'utiliser un système de RAG basé sur des requêtes de graphe [Peng et al., 2024]. Cette technique est appelée « Graph RAG » dans la littérature.

Depuis la naissance du web sémantique (fin des années 1990, début des années 2000), divers travaux visant à représenter toutes les connaissances de notre monde dans des ontologies généralistes ont vu le jour, tels que Wikidata et DBpedia.Ces ontologies couvrent divers domaines allant de la culture générale aux connaissances académiques comme la structure des molécules, mais aucune d'entre elles ne peut se vanter d'être exhaustive. Wikidata, par exemple, est une ontologie collaborative. Malgré des années de travail par des gens du monde entier, elle est loin de représenter tous les domaines. D'autre part, dans le monde professionnel, on peut avoir besoin d'une ontologie représentant les spécifications internes d'une entreprise, dans le cas d'une entreprise souhaitant mettre à disposition un agent de questions-réponses pour ses employés, par exemple. Pour développer ce genre d'agents, il y a une nécessité de créer des ontologies plus spécifiques et personnalisées. Ce processus fait souvent appel à un échange entre un ingénieur des connaissances, dont le rôle est de formaliser une ontologie, et des experts qui connaissent et sont capables d'expliquer les connaissances d'un domaine. Cet échange est long et ardu, d'autant plus que ces experts sont de plus en plus rares.

Afin de remédier à ces difficultés, nous pensons qu'il est possible de faire appel aux corpus de spécialité. Rédigés en langue naturelle par les clients ou les experts domaine, ils contiennent des connaissances utiles à la construction de l'ontologie de

9. L'instanciation des concepts s'appelle aussi la « population de l'ontologie ».

domaine. Grâce à un système d'acquisition des connaissances, le corpus pourrait être transformé en ontologie automatiquement. Cette science s'appelle l'apprentissage d'ontologies (*Ontology Learning*). Il existe de nombreux travaux essayant d'achever ce résultat, dont nous discuterons dans l'état de l'art. Dans l'idéal, un tel système est capable de remplacer le travail de l'ingénieur des connaissances, et concrètement, et dans l'absolu, d'assister son travail pour réduire la charge cognitive et le temps passé à échanger avec des experts domaine pour produire une ontologie spécialisée. Dans le cas d'un assistant, l'ingénieur des connaissances peut confirmer ou infirmer les connaissances extraites au lieu de les écrire lui-même. Il peut ainsi se concenter sur des tâches moins rébarbatives, telles que l'ajout des connaissances manquantes et la définition d'axiomes. C'est la problématique à laquelle ce mémoire tente de répondre : **comment automatiser l'acquisition des connaissaces à partir de corpus de spécialité ?** Dans une première partie, nous allons présenter le cadre de ce mémoire, avant de présenter les méthodes qui aboutissent à la création d'une ontologie, puis nous présenterons les expériences réalisées. Enfin, nous discuterons des forces et des faiblesses de notre méthode, avant de proposer des perspectives d'amélioration.

Plan de lecture

Ce mémoire s'articule autour de cinq chapitres. Le premier chapitre fera un état de l'art des systèmes d'*Ontology Learning* et d'autres technologies d'extraction d'informations. Le deuxième chapitre présentera le contexte général et ancre ces travaux dans deux projets de recherche : KESAIO et SUMINO, avant de présenter les corpus utilisés. Le troisième chapitre explicitera la méthodologie employée et les différentes étapes de notre système d'apprentissage d'ontologies, allant de l'extraction des connaissances au référencement de l'ontologie. Le quatrième chapitre présentera les métriques d'évaluation les résultats obtenus pour les différentes étapes de la constitution de l'ontologie. Enfin, dans le dernier chapitre, nous discuterons de nos réalisations et des pistes d'amélioration avant de conclure.

CHAPITRE 1

ÉTAT DE L'ART

Sommaire

Cette partie fait un état de l'art des techniques d'apprentissage d'ontologie, d'extraction d'informations en contexte ouvert (OIE), de méthodes de référencement d'ontologie (*Ontology Referencing*) et enfin, de l'évaluation des ontologies.

1.1 Ontology Learning

Dans ce chapitre, nous présentons des systèmes d'apprentissage d'ontologies complets. Un système d'apprentissage d'ontologie (*Ontology Learning*, ou OL) est un système qui regroupe toutes les étapes, techniques et procédés nécessaires à générer automatiquement une ontologie à partir de connaissances non structurées telles que dans un corpus en langue naturelle. Ces procédés incluent la reconnaissance d'entités nommées, la sélection des entités pertinentes, l'extraction des relations entre les entités, la découverte de la taxonomie, le référencement d'ontologie (*Ontology Referencing*), le liage d'entité (*Entity Linking*), la découverte d'axiomes et l'évaluation de l'ontologie. Des systèmes complets proposent différentes approches et mettent en place un processus dont la sortie est une ontologie. Dans l'*Ontology Learning*, il existe des approches traditionnelles (statistiques et symboliques) et des méthodes récentes basées sur l'apprentissage automatique dont les LLMs. [Du et al., 2024] propose un état de l'art de ce domaine en 2024 en mettant en avant l'utilisation des LLMs. Mais dans cet état de l'art, nous mettrons aussi l'accent sur les méthodes statistiques et symboliques.

DOG4DAG

[Wächter et al., 2011] ont mis à disposition un outil fonctionnant sous l'éditeur d'ontologies Protégé [1]. Cet outil a pour objectif d'assister la création des ontologies de façon semi-automatique, avec confirmation intermédiaire de l'utilisateur. DOG4DAG

1. Protégé est un logiciel d'édition d'ontologies largement utilisé dans le domaine de l'ingénierie des connaissances.

se base sur des méthodes symboliques et statistiques pour extraire des termes, générer leurs définitions ainsi que des relations hiérarchiques entre les concepts rencensés. Les termes sont extraits des phrases nominales signifiantes déterminées par des méthodes statistiques. Les concepts sont référencés grâce à la distance de Hamming, mesurant la similarité entre deux termes. Les chercheurs évaluent chaque étape de leur processus à partir d'une référence « gold » annotée manuellement. Après évaluation, les chercheurs montrent que 78 % des définitions et au moins 54 % des relations hiérarchiques sont valides.

RelEx

Le système proposé par [Fundel et al., 2006] extrait des noms de protéines d'un corpus médical, puis classifie les relations entre ces protéines à l'aide d'un algorithme à base de règles et une liste de relations prédéfinie. Les relations candidates sont filtrées pour, par exemple, retirer les relations qui ont une négation. Cette méthode d'extraction et de classification de relations obtient un rappel de 78 % sur le corpus MEDLINE (domaine médical). Un tiers des erreurs de relations non annotées alors qu'elles auraient dû l'être (faux négatifs) sont causées par des motifs non pris en compte dans l'heuristique, et deux tiers sont dûes à de mauvaises performances de l'annotateur en parties du discours et en dépendances syntaxiques.

REBEL

Dans [Huguet Cabot and Navigli, 2021], les chercheurs transforment la tâche d'extraction de triplets en tâche de traduction. Les chercheurs entraînent un modèle BERT à prédire des expressions suivies aux balises <subj>, <pred>, <obj> correspondant à ces expressions sur un corpus annoté en anglais. Les prédicats ne sont pas forcément des verbes. En interne, nous avons testé cette méthode sur un corpus en français et avec le modèle BARThez. L'avantage de cette méthode est sa simplicité, mais son désavantage est son manque d'adaptabilité à des nouvelles données formattées différemment : en effet, les relations que le modèle BERT arrive à extraire sont les 200 relations Wikidata les plus fréquentes : « *pays* », « *lieu de fondation* », etc. Pour le français, le modèle entraîné par notre équipe de recherche obtient un score BLEU à 0,75 et un score METEOR à 0,87.

LLMs4OL

Dans [Giglou et al., 2023], les chercheurs font l'hypothèse qu'utiliser les LLMs pour l'*Ontology Learning* augmentera les performances globales, car les LLMs sont capables de s'adapter à un nouveau problème grâce au *Prompt Engineering*, et de s'adapter à tout type de corpus grâce à la richesse de leurs données d'entraînement et de comprendre les relations entre les mots. L'objectif de l'article est de comparer les performances de huit LLMs sur différentes tâches. D'après cette recherche, pour compléter l'apprentissage d'ontologie, les tâches que les LLMs doivent être capable de résoudre incluent la préparation du corpus, l'extraction de la terminologie, le typage des termes (conceptualisation), la construction de la taxonomie, l'extraction de relations non taxonomiques et la découverte d'axiomes. D'après les résultats présentés, la découverte de relations taxonomiques est la tâche fonctionnant le mieux, avec un score de précision (MAP@1) maximal de 78,1 % par GPT-4 sur le jeu de données UMLS. La découverte de relations non taxonomiques obtient une précision de 49,5 %

par Flan-T5-XL sur UMLS. Enfin, la découverte de types obtient un score de 39,4 % par GPT-4 sur Geonames (entités géographiques).

1.2 Extraction d'informations en contexte ouvert

L'extraction d'informations en contexte ouvert (*Open Information Extraction*, ou OIE) est le fait d'extraire des triplets <sujet, prédicat, objet> à partir de textes en langue naturelle, où le sujet et l'objet sont des entités, et le prédicat est une relation entre eux ou une action de l'un sur l'autre. Ce domaine est étroitement lié à l'apprentissage d'ontologies, puisque les connaissances extraites de corpus forment les instances de l'ontologie regroupées au sein du graphe de connaissances. La plupart des articles d'OL abordent la question de l'OIE, qui est une sous-catégorie de l'OL cherchant à extraire des informations disparates, tandis que l'OL cherche non seulement à les extraire, mais aussi à les organiser. Comme pour l'apprentissage d'ontologies, nous allons explorer des techniques basées sur les réseaux de neurones et les méthodes traditionnelles, statistiques et symboliques. [Kamp et al., 2023] passe en revue toutes les applications de l'OIE, parmi lesquelles le résumé automatique de texte et les systèmes de questions-réponses, et présente l'état de l'art en 2023 pour chacune de ces techniques. Dans cet état de l'art, nous allons nous focaliser sur l'extraction de triplets.

BRMiner

L'extraction d'informations en contexte ouvert à partir des dépendances syntaxiques a été remise au goût du jour par BRMiner [Prakash et al., 2021]. L'objectif est de générer des règles et une terminologie de domaine à partir de corpus décrivant de des règles professionnelles en langue naturelle contrôlée (*Controlled Natural Language*, ou CNL) limitant les structures syntaxiques complexes. Les chercheurs mettent l'accent sur l'importance de la non-ambiguité et la clarté des faits dans les corpus. Chaque phrase est analysée itérativement pour extraire une terminologie de concepts et de faits, alias prédicats. L'extraction des concepts obtient une F-mesure variant entre 0,96 et 0,98 en fonction du corpus, et l'extraction des triplets obtient une F-mesure variant entre 0,72 et 0,92. L'extraction des règles à partir des triplets obtient un score de précision variant entre 0,64 et 0,85. Chaque étape dépend des performances de l'étape précédente, ainsi, il est logique de voir les scores diminuer au plus la tâche se complexifie. En termes de perspectives, l'équipe aimerait effectuer une reformulation des phrases longues et complexes au préalable, mettre en place une résolution automatique des coréférences, et de créer une taxonomie pour les termes extraits. Enfin, ils envisagent de fine-tuner [2] un modèle pour automatiser l'extraction de triplets.

Relext

La méthode proposée par [Schutz and Buitelaar, 2005] se base également sur l'analyse syntaxique pour l'extraction des connaissances. Relext identifie les verbes en tant que prédicats des triplets. Ensuite, il extrait leurs arguments syntaxiques (compléments, sujet, adverbe...), puis filtre les termes non pertinents à l'aide de mesures statistiques. Relext vise les corpus de spécialité et a pour objectif de créer un

2. Fait de ré-entraîner un modèle neuronal déjà entraîné avec des données complémentaires.

système de questions-réponses sur un domaine en particulier. Dans cette étude, la précision de l'extraction des triplets atteint 23,9 % et le rappel 36,0 % dans le cas où un seul triplet est extrait de chaque verbe, et une précision de 20,2 % et un rappel de 43,0 % dans le cas où tous les triplets possibles sont extraits de chaque verbe (correspondant aux différents arguments du verbe).

DualOIE

L'approche abordée par [Chen et al., 2024] utilise un prompt qui demande à un LLM génératif d'extraire les triplets d'un texte. L'article met l'accent sur les triplets difficiles à extraire avec des méthodes traditionnelles, notamment 4 types de triplets qui sont les triplets superposés (partageant un même élément), les triplets discontinus (prédicat séparé des autres mots), les triplets imbriqués (deux triplets partagent un mot de la phrase) et les triplets implicites (dont certaines informations sont manquantes). Les LLMs sont capables de compléter ces informations manquantes grâce à leurs connaissances préalables et à leur capacité générative. Il est possible d'extraire des prédicats au préalable et de demander au LLM de les compléter avec le sujet et l'objet des triplets. L'article compare différents LLMs génératifs servant à l'OIE, tels que ChatGPT, Gen2OIE, IMoJIE, et NOIE + BERT, ainsi que DualOIE, le modèle développé pour cette étude. DualOIE, par rapport aux autres LLMs, a été entraîné non seulement à extraire les triplets, mais aussi à réécrire le texte d'origine à partir des triplets. La F-mesure maximale est estimée à 59,5 pour le modèle DualOIE sur la tâche d'extraction de triplets et sur les jeux de données CaRB et SAOKE.

1.3 Référencement d'ontologies et liage d'entités

Le référencement d'ontologies (ou *Ontology Referencing*) est étroitement lié au liage d'entités (*Entity Linking*) puisque ces deux domaines consistent à trouver des liens entre concepts issus de deux ontologies ou entre une ontologie et du texte. La différence est que le liage d'entités se consacre à relier les mentions dans un corpus à une ontologie, tandis que le référencement d'ontologies consiste à détecter que deux classes dans deux ontologies différentes sont en réalité le même concept. Le liage de relations (*Relation Linking*) est la même chose que le liage d'entités, mais pour les relations. Dans cette partie, nous passons en revue l'état de l'art du liage d'entité et du référencement d'ontologies.

Embeddings de graphe pour l'alignement d'ontologies

Afin de référencer deux ontologies entre elles, nous avons remarqué que la technique qui fonctionne le mieux est d'utiliser des embeddings. De nombreuses recherches tentent de trouver le meilleur moyen de vectoriser un graphe, ce que nous allons voir ci-après. Dans notre méthode, nous nous inspirons des techniques existantes mais en utilisant des modèles de NLI (*Natural Language Inference*), dont nous reparlerons partie 3.4. Parmi ces méthodes, on compte JAPE, qui tente de connecter les concepts de DBPedia entre eux dans plusieurs langues ; OWL2Vec*, qui renvoie une version vectorisée d'une ontologie ; BERTmap, qui utilise un modèle BERT pour encoder l'ontologie. Enfin, en réaction aux avancées récentes des modèles de langues, ChatEA s'appuie sur la puissance des LLMs pour effectuer la tâche d'alignement d'ontologies. Cette partie détaille le fonctionnement et les découvertes de chacun de ces travaux.

OWL2Vec*

Le modèle présenté par [Chen et al., 2021] prend en entrée une ontologie et produit un vecteur par entité. Il se base sur l'algorithme *Random Walk* pour générer des embeddings des instances de concepts. Ces embeddings représentent trois aspects de l'ontologie : la structure du graphe, les informations lexicales et les relations logiques (axiomes). Après alignement et classement des entités candidates pour une entité de la première ontologie, un score MRR et hit@k sont calculés sur trois ontologies (HeLis, FoodOn et GO) et avec des paramètres de *Random Walk* différents. Ils montrent que, pour chaque ontologie, OWL2Vec* dépasse l'état de l'art dans l'utilisation des embeddings de graphes.

JAPE

Dans [Sun et al., 2017], les chercheurs présentent une méthode d'alignement d'ontologies multilingues par embeddings en employant la bibliothèque TensorFlow. Ils évaluent leur méthode sur DBpedia. Deux embeddings sont générés : l'un représentant les attributs de l'entité, l'autre représentant la structure de son voisinage. Ils atteignent un hit@1 à 41,18 sur l'alignement DBpedia du chinois vers l'anglais, mais des scores inférieurs sur d'autres couples de langues. En revanche, en combinant la méthode JAPE à un traducteur automatique, le hit@1 pour ces deux langues passe à 73,09, mais l'alignement devient dépendant de l'outil de traduction. Ainsi, leur méthode d'embeddings d'ontologie reste à améliorer.

BERTMap

Dans [He et al., 2022], un modèle BERT est fine-tuné de façon non supervisée pour l'alignement d'ontologies. Il exploite trois aspects des entités : leur contexte sémantique, structurel et logique. Après évaluation sur des ontologies médicales, les chercheurs constatent l'efficacité du modèle BERT à repérer des similarités entre des concepts dont le point commun est un trait sémantique secondaire, moins évident.

Modèles de langues

ChatEA

Dans [Jiang et al., 2024], les chercheurs estiment que les connaissances préalables des LLMs peuvent aider à l'alignement d'entités. Ils développent un système de conversion entre le code de graphe de connaissances et un vecteur de l'entité. Pour une liste d'entités candidates, ChatEA estime quatre scores de similarité : le nom de l'entité, la description, la structure et les informations temporelles de l'entité. La description est générée par le LLM. Si aucune entité ne convient, la recherche est effectuée sur une plus grande échelle dans le graphe de connaissances.

1.4 Hiérarchisation

COBWEB

COBWEB [Fisher, 1987] est un algorithme apparu en 1987 qui permet de construire des classes et une taxonomie de ces classes à partir d'individus et de leur voisinage dans un graphe de connaissances. Il utilise des méthodes non supervisées

à base de clustering. Pour ce faire, il se base uniquement sur les paramètres d'un individu (par exemple, s'il a une taille, une couleur, et si oui, quelles sont sa taille et sa couleur). Trestle [Maclellan et al., 2016] utilise l'algorithme de COBWEB en introduisant sa propre mesure du score d'utilité de catégorie (*Category Utility*) qui prend en compte des valeurs numériques au sein des paramètres, en plus de tester la présence et l'égalité d'un paramètre et de sa valeur. Pour catégoriser une nouvelle instance sous un concept ou créer un nouveau concept, COBWEB et Trestle font quatre hypothèses par nœud et mesurent le score de Category Utility des enfants du concept parent : l'hypothèse où l'instance est ajoutée à un concept existant (*adding*), l'hypothèse où le nœud doit être associé à un nouveau concept héritant d'un autre concept (*creating*), l'hypothèse où le nœud doit devenir un parent intermédiaire entre une autre relation parent-enfants (*merging*) et l'hypothèse où une relation intermédiaire doit être supprimée (*splitting*).

Modèles de langues

La méthode présentée par [He et al., 2024] ré-entraîne des modèles de langues à encoder la hiérarchie d'une ontologie. En entrée du modèle, un embedding de chaque entité est fourni en se basant sur les phrases d'origine de l'entité. En sortie, ils récupèrent des triplets composés de 2 entités et d'une relation de subsomption. Les tests ont été effectués avec les modèles de langue all-MiniLM-L6-v2 et all-mpnet-base-v2. Ils montrent que la mise en place d'un fine-tuning du modèle, puis de l'apprentissage de l'encodage de la hiérarchie permettent d'augmenter de façon significative les scores de précision, rappel et F-mesure de la découverte de la taxonomie (scores 7 fois supérieur par rapport aux modèles de langue de base), avec une F-mesure atteignant 0,9 en moyenne pour les deux modèles dans différentes configurations.

1.5 Évaluation des ontologies

Évaluer une ontologie est un travail difficile, étant donné qu'il n'existe pas de format optimal pour la représentation des connaissances. Avoir une ontologie de référence « gold standard » et comparer les triplets entre les deux ontologies n'est pas forcément la façon la plus pertinente de faire, tout comme nous ne comparons pas deux traductions mot à mot. Il est nécessaire de comprendre la sémantique qui a été capturée par l'ontologie pour pouvoir l'évaluer, cela faisant souvent appel aux embeddings de graphes, auxquels on donne alors le rôle de capturer le sens d'une entité. Mais des mesures sur la structure, sur la couverture de la terminologie etc. ont aussi leur place dans l'évaluation des ontologies, puisque l'évaluation peut non seulement s'effectuer sur l'ontologie complète, mais aussi sur les étapes de sa fabrication et des aspects individuels. Dans cette partie, nous allons faire l'état de l'art des différentes stratégies d'évaluation qui existent.

Les quatre stratégies d'évaluation

Dans [Raad and Cruz, 2015], les chercheurs répertorient les aspects évaluables d'une ontologie et donnent quelques pistes d'évaluation, mais sans être spécifiques quant à la procédure, laissant la liberté d'adapter les métriques au cas par cas. Quatre grandes stratégies d'évaluation sont proposées : celles basées sur une comparaison avec une ontologie de référence (*« gold standard »*), celles basées sur une

Niveau	Gold standard	Tâche	Données	Critères humains
Lexique, vocabulaire, concepts	X	X	X	X
Hiérarchie et taxonomie	X	X	X	X
Autres relations sémantiques	X	X	X	X
Contexte et application		X		X
Syntaxique	X			X
Structure, architecture et design				X

TABLE 1.1 – Tableau des aspects de l'ontologie évaluables en fonction de la méthode proposé par [Asim et al., 2018].

comparaison avec un corpus (*« data driven »*), celles orientées sur une tâche (*« task-oriented »*), et enfin, celles basées sur des critères humains. Selon l'article, combiner au moins deux de ces quatre stratégies d'évaluation permet d'avoir un aperçu des performances générales de l'ontologie. Parmi les aspects à évaluer, il y a l'exactitude (vérifiant que les classes, les propriétés, etc sont correctes), la complétude ou couverture de domaine, la concision (ne pas avoir d'informations non pertinentes au domaine), l'adaptabilité à plusieurs tâches, la clarté, l'efficacité computationnelle et la consistence (absence de contradictions).

Dans [Asim et al., 2018], l'évaluation des ontologies est également divisée en quatre catégories (*gold standard*, *data-driven*, *task-oriented* et sur des critères humains), mais les chercheurs proposent en plus un état de l'art avec des implémentations spécifiques de ces stratégies d'évaluations pour chacune des quatre approches. Ils dressent un tableau des aspects évaluables en fonction de la méthode d'évaluation, voir tableau 1.1.

Mesure de la couverture de domaine

La mesure de couverture de domaine proposée dans [Spyns and Reinberger, 2005] est basée sur des calculs statistiques sur des tokens, notamment le score de spécificité et la loi de Zipf, pour déterminer l'importance d'un token dans un corpus. Une fois les tokens importants extraits, ils mesurent la précision du modèle d'extraction de triplets sur les tokens présents dans l'ontologie. La réflexion se concentre sur le vocabulaire, et non pas sur les triplets ou les entités, mais il est intéressant de mesurer cette information afin de savoir si les termes les plus représentatifs ont bien été capturés dans l'ontologie. Les chercheurs fixent un seuil d'importance : si, dans une classe de Zipf, il y a plus de 60 % de mots qui sont spécifiques au corpus, alors les mots de cette classe sont significatifs. Leur méthode d'évaluation leur permet d'obtenir des scores de couverture, de rappel et précision sur leur modèle d'extraction.

Dans [Zaitoun et al., 2023], les chercheurs proposent des métriques visant à mesurer l'exactitude et la couverture d'une ontologie de domaine en utilisant un modèle d'extraction fine-tuné sur un corpus du même domaine.

Évaluation avec une ontologie de référence

[Zavitsanos et al., 2011] est une méthode d'évaluation de type *gold standard* qui utilise la représentation vectorielle des différents concepts présents dans l'ontologie, permettant un calcul de similarité entre l'ontologie apprise et l'ontologie de référence. Les chercheurs expérimentent sur des opérations visant à modifier la taxonomie pour

mesurer à quel point ces opérations affectent sa qualité. Cette stratégie d'évaluation par altérations permet de savoir à quel point la taxonomie est bien organisée.

Évaluation par critères

Delta [Kondylakis et al., 2021] est un outil qui propose une interface en ligne pour l'évaluation d'ontologies. La plateforme affiche des métriques, statistiques et erreurs de l'ontologie. Parmi les statistiques, on trouve le nombre de propriétés, le niveau de détail sémantique, le nombre de classes et d'individus, etc. Ces statistiques de base reflètent une certaine qualité de l'ontologie. Des métriques, telles que le nombre d'entités, la richesse d'attributs, la population moyenne, etc. sont calculés avec diverses formules. Les erreurs sont divisées en niveaux de gravité. Elles aident à détecter et régler des problèmes de l'ontologie, comme des annotations manquantes, des classes isolées, etc.

OntoMetrics [Lantow, 2016] met à disposition des métriques d'évaluation automatiques. Les chercheurs font une distinction entre les différents cycles de vie de l'ontologie (population, taxonomie...) et reprennent les concepts d'évaluation des ontologies de [Raad and Cruz, 2015] et de la conception d'ontologies de [Fernández-López et al., 1997].

Conclusion

Dans cet état de l'art, RelExt[Schutz and Buitelaar, 2005] propose la méthode d'extraction de triplets la plus proche de celle présentée dans ce mémoire. Dans notre méthode, nous mettons en œuvre la sélection d'entités, l'extraction des relations, la découverte de la taxonomie, le référencement d'ontologie et l'évaluation d'ontologies. Il nous semblait important de faire le tour des technologies existantes, en passant par les méthodes les plus anciennes (symboliques) au plus récentes (LLM), tout en gardant en tête nos contraintes en termes de puissance de calcul et de taille de corpus. Bien que notre méthode exploite des modèles de langue pour les tâches de base du TAL, nous n'utiliserons pas les outils présentés ci-dessus en raison de leur difficulté d'accès ou de leur incompatibilité avec nos données ou nos contraintes.

CHAPITRE 2

CONTEXTE

Sommaire

Cette partie présente le cadre de ce mémoire. Nous commencerons par une brève description de l'entreprise d'accueil et de ses activités, avant de détailler les deux projets de recherche, KESAIO et SUMINO, qui ont fait l'objet d'études et d'expérimentations dans le cadre de ce travail. Enfin, nous replacerons le sujet du mémoire dans le contexte de ces projets et de l'entreprise et présenterons les deux corpus de spécialité utilisés.

2.1 Entreprise d'accueil

Airudit est une SAS française fondée en 2013 et basée sur la métropole bordelaise. L'activité principale d'Airudit est le développement de systèmes de questions-réponses personnalisés et pilotés par la voix. Ces systèmes de questions-réponses s'appuient sur l'exploitation d'ontologies spécialement conçues et possèdent une chaîne de traitement de la langue naturelle pour interpréter la question et trouver les bonnes informations dans l'ontologie. Aujourd'hui, deux solutions sont proposées par l'entreprise : Prometh.ai et Valer.ia.

Prometh.ai est un système de questions-réponses composé de diverses briques de traitement automatique du langage, allant de la reconnaissance de la parole à la tokenisation, l'annotation en parties du discours et en dépendances syntaxiques, au liage d'entités et à la génération de la réponse à partir des informations trouvées dans l'ontologie ou à défaut, dans une source de données externe pointée par l'ontologie.

Valer.ia est une version légère de Prometh.ai. Elle est conçue pour fonctionner comme système intégré au matériel client, avec des capacités de traitement limitées.

L'entreprise est divisée en quatre pôles : le pôle R&D, le pôle projet, le pôle produit et le pôle innovation. Le pôle R&D se concentre principalement sur la recherche en TAL et en reconnaissance automatique de la parole. Le pôle projet et le pôle produit s'occupent de maintenir Prometh.ai et Valer.ia tout en travaillant avec les clients pour créer des outils adaptés à leurs besoins.

2.2 Projets de recherche

Les deux projets de recherche auxquels sont rattachés ce projet, KESAIO et SUMINO, sont financés par l'Agence de l'Innovation de Défense (AID).

KESAIO (*Knowledge Empowerment Software to Automate Improvements of Ontologies*) est un projet de recherche dont l'objectif est l'enrichissement semi-automatique d'ontologies. Il est effectué en collaboration entre Airudit et l'équipe Orpailleur du LORIA, dont l'objet de recherche est l'extraction et la représentation de connaissances.

SUMINO (*Summarization in Open Context*) a pour objectif le résumé extractif et abstractif multi-document. Il est effectué en collaboration entre Airudit et l'équipe SyNaLP du LORIA, dont l'objet de recherche est le traitement automatique des langues naturelles par méthodes statistiques et symboliques.

2.3 Cadre du mémoire

Ce mémoire s'intéresse à l'automatisation de l'acquisition des connaissaces à partir de corpus de spécialité de façon non supervisée. Plus précisément, il porte sur l'extraction des connaissances à partir de corpus et la structuration de ces connaissances au sein d'une ontologie, ce qui s'inscrit pleinement dans le projet KESAIO.

Le lien avec le projet SUMINO réside dans l'extraction d'informations multi-documents. Une information stockée dans une ontologie est non redondante et considérée comme importante. L'ontologie joue le rôle de résumé structuré d'un domaine donné. Par ailleurs, le corpus Catastrophes 2.4.1, que nous présentons ci-dessous et sur lequel reposent plusieurs expériences menées dans ce mémoire, est aussi le corpus du projet SUMINO. Il est partagé entre Airudit et l'équipe du LORIA.

Du point de vue d'Airudit et des pôles projet et produit, la semi-automatisation de la création des ontologies vise à améliorer l'efficacité de cette tâche à partir des corpus fournis par les clients. Cette étape est longue mais nécessaire au début de chaque début de projet. À terme, Airudit aimerait créer un système entièrement autonome, capable de créer des ontologies de manière automatique, sans intervention d'un ingénieur des connaissances, et qu'il sache s'adapter à tous types de corpus afin de permettre aux clients de fournir leurs données pour voir l'ontologie se générer et/ou se mettre à jour sous leurs yeux.

2.4 Corpus

Dans le cadre de la constitution d'un corpus servant à créer une ontologie, nous devons d'abord définir un besoin spécifique auquel l'ontologie doit répondre [Fernández-López et al., 1997]. Par exemple, nous avons besoin d'un corpus large sur un domaine spécifique si nous voulons être capables de synthétiser ce domaine, ou d'un corpus technique décrivant le fonctionnement d'une entreprise si nous souhaitons créer une ontologie abordant des scénarios professionnels. Afin de répondre à ces deux objectifs, nous avons utilisé deux corpus en langue française distincts, que nous présentons ci-dessous : le corpus Catastrophes et le corpus Solaris.

2.4.1 Corpus Catastrophes

Le corpus Catastrophes a été constitué dans le carde du projet SUMINO. La constitution de ce corpus à partir du web se divise en deux étapes : l'extraction de noms de catastrophes industrielles et l'extraction de noms de catastrophes naturelles, ces deux extractions suivant un protocole similaire. Le corpus a été extrait principalement d'articles de presse relatant de catastrophes. Nous avons établi deux versions : une version courte contenant les résumés des articles dans le moteur de recherche, sur laquelle nous ferons nos expériences ; et une version longue contenant le corps des articles en entier. Le corpus Catastrophes (version courte) fait 53 189 tokens répartis sur 1155 documents, tandis que le corpus Catastrophes entier (articles) fait un peu plus d'un millions de tokens répartis sur 1158 documents.

Catastrophes industrielles

Pour l'extraction de catastrophes industrielles, nous avons d'abord extrait le nom d'une centaine de catastrophes industrielles depuis Wikipedia. Notre premier objectif est d'obtenir une liste de requêtes web pour pouvoir extraire les pages concernant ces catastrophes, ou une liste de noms de catastrophes. Pour trouver ces noms, nous avons récupéré la liste des URLs sur la page Wikipedia anglaise des catastrophes industrielles automatiquement (List of industrial disasters). Puis, nous filtrons manuellement les articles Wikipedia qui parlent de catastrophes industrielles, car certains liens pointent vers des articles décrivant non pas la catastrophe en question, mais l'endroit où elle a eu lieu ou l'objet de cette catastrophe (usine, pétrolier...). Enfin, pour chaque lien pointant vers les articles en anglais, nous récupérons le lien vers la version française de cette page si elle existe, puis nous récupérons le titre de la page, qui donne généralement l'intitulé de la catastrophe. La liste des catastrophes industrielles extraites de Wikidata anglais a été complétée par celles issues de la page Wikipedia listant les catastrophes industrielles en français : (Chronologie de catastrophes industrielles), qui contient des liens absents dans son homologue anglaise, par exemple la catastrophe intitulée « Naufrage du pétrolier Prestige ». La page étant moins structurée que sa version anglaise, il a fallu recopier les liens à la main. Un échantillon de noms de catastrophes récupérées avec cette méthode est présenté en annexe A.1.

Après fusion, nous obtenons 94 noms de catastrophes industrielles, dont 42 provenant de la liste en anglais, et 77 de celle en français, avec intersection entre les deux ensembles.

Catastrophes naturelles

Pour les catastrophes naturelles, nous avons également récupéré une liste de noms de catastrophes avant de procéder à l'absorption du corpus. Voici les pages Wikipedia listant des catastrophes naturelles en anlais et en français. Bien que ces deux pages soient une traduction l'une de l'autre, la page anglaise a reçu quelques actualisations supplémentaires. Nous n'avons utilisé que la page française, mais il aurait été aussi pertinent d'utiliser celle en anglais, de plus la majorité des pages recensées ont une version française.

Depuis la page Wikipedia de la liste des catastrophes naturelles en français, nous avons récupéré une liste de 70 noms de catastrophes naturelles. Un échantillon de noms de catastrophes naturelles est disponible en annexes A.2.

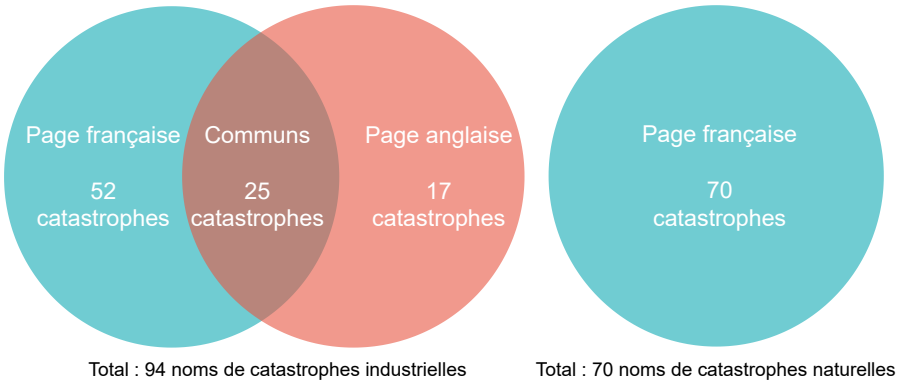

FIGURE 2.1 – Nombre de noms de catastrophes issus des pages Wikipedia listant les catastrophes et ayant servi au requêtage pour la collecte du corpus Catastrophes.

Collecte du corpus

L'étape suivante consiste, à partir de ces 94 industrielles + 70 naturelles = 164 noms de catastrophes (voir figure 2.1), récupérer le contenu des articles issus de la presse qui relatent les faits de ces catastrophes. L'objectif de cette démarche est d'avoir des sources diverses qui décrivent le même événement pour tester la capacité de l'ontologie à synthétiser un sujet. Ainsi, pour 164 noms de catastrophes, à l'aide du scraper Selenium, nous avons absorbé un total de 1158 pages web, soit en moyenne 7,1 pages par catastrophe. Ce nombre varie en fonction de la popularité des catastrophes dans la presse. Le corpus étant issu du web, il contient beaucoup de bruit. Nous avons tenté de diminuer le bruit avec un modèle NLI (trop permissif) et un extracteur à base de statistiques sur la longueur des paragraphes par type de balise HTML (trop strict).

2.4.2 Corpus Solaris

Le corpus Solaris est un corpus métier qui a été constitué dans le but de répondre à des questions techniques d'une entreprise. Ce corpus se spécialise dans la gestion des risques chimiques. Le corpus Solaris étant un corpus métier, toutes ses informations sont importantes, contrairement à un corpus moins technique de type presse, littéraire ou issu du web, dans lesquels beaucoup plus d'informations parasites vont être présentes, nécessitant un filtrage des informations extraites. Ce corpus a été recueilli dans le but d'aider les ingénieurs des connaissances à construire une ontologie pour un client. Il contient une liste de questions et de réponses. Seules les réponses y sont informatives, ainsi, les questions ne sont pas utilisées dans l'extraction d'informations. Il est représentatif des corpus que l'on retrouve généralement en entreprise car il contient des descriptions techniques, des consignes pour les salariés, etc. Il n'est constitué que de 2366 tokens pour 171 phrases après nettoyage. Il contient beaucoup de vocabulaire de métier absent du dictionnaire, parmi lesquels quelques noms propres et noms de codes, qui n'existent pas dans une ontologie générale telle que Wikidata et nécessitent donc leur enregistrement dans une ontologie métier. Le contenu de ce corpus étant privé, les exemples tirés de ce corpus ont été anonymisés.

CHAPITRE 3

MÉTHODES

Sommaire

3.1 Présentation générale

Dans un premier temps, nous proposons une définition d'une ontologie dans le contexte de l'informatique. Une ontologie est une représentation qui permet d'organiser et de stocker des connaissances. Toutes les connaissances sont retranscrites en triplets <sujet, prédicat, objet> dans un format tel que RDF, OWL ou Turtle. La plupart du temps, le prédicat est un verbe, ou le nom d'un attribut que l'on veut donner au sujet, par exemple <Angela Merkel, date naissance, 17 juillet 1954>. Il existe quatre niveaux de données que l'on peut stocker. Le premier niveau est la taxonomie, ou hiérarchie des classes. Le deuxième niveau correspond aux instances de ces classes, souvent associées aux individus peuplant l'ontologie. Le troisième niveau correspond aux relations entre les entités, qui sont soit des classes, soit des instances, soit des valeurs telles que des chaînes de caractères, entiers, dates... Le dernier niveau sont les axiomes, ou règles générales. La taxonomie définit des relations d'héritage entre les classes de type <A, sous classe de, B>, par exemple <chat, sous classe de, félin> et <félin, sous classe de, mammifère>. Cela s'oppose aux instances, qui dans cet exemple, vont décrire un chat physique, lui associer un nom, une date de naissance spécifique, une couleur de pelage etc. L'instance d'un chat nommé Mallow est définie par la relation « est un » : <Mallow, est un, chat>. Enfin, les relations entre les instances peuvent prendre n'importe quelle valeur : <Mallow, est le frère de, Pitto>. Il est possible de donner des valeurs à des instances, comme <Mallow, a pour âge, 10>, où 10 n'est pas une entité, mais une valeur numérique. Ceci

est la méthode la plus courante pour représenter des connaissances dans une ontologie. Les ontologies du point de vue de l'informatique ont été largement définies dans les années 1990, comme dans les trois ouvrages suivants : [Guarino, 1998] qui définit la globalité des concepts liés aux ontologies, [Horrocks et al., 2003] qui décrit le web sémantique dans OWL, et [Studer et al., 1998] qui discute de l'apparition des ontologies dans le domaine de l'ingénierie des connaissances.

Cependant, chaque ontologie a sa propre façon de représenter les connaissances. En général, les ingénieurs des connaissances en charge de créer ces ontologies vont d'abord cerner le besoin et définir un schéma de représentation qui convient le plus au problème étudié, conformément aux principes de [Fernández-López et al., 1997]. Dans notre cas, nous souhaitons tendre vers un schéma qui serait adapté à n'importe quel corpus de domaine. De plus, l'usage qui sera fait de l'ontologie résultante doit être pris en compte, soit un système de questions-réponses et/ou de résumé de texte. Étant donné l'aspect langue naturelle présent dans les corpus et dans l'utilisation ciblée, conserver un format adapté à la langue naturelle devrait, en théorie, permettre de représenter de façon fidèle les connaissances extraites d'un corpus, tout en étant exploitable par un modèle génératif tel qu'un LLM. Dans la partie 3.2.2, nous détaillerons le procédé de réification [1], qui a permis d'achever ce résultat, ainsi que les relations introduites pour rendre compatibles langue naturelle et ontologies. Le système de génération de taxonomie sera décrit dans la partie 3.3. En parallèle de cela, notons qu'introduire un certain équilibre entre langue naturelle et abstraction de haut niveau serait idéal. Une ontologie de plus haut niveau est une ontologie qui a assimilé les informations et les a structurées, s'éloignant de la langue naturelle pour se rapprocher d'un modèle plus standardisé. Dans ce mémoire, les triplets sont transcrits de cette façon : <sujet, prédicat (adverbe) [adposition], objet>. « None » signifie que l'une des valeurs du triplet est laissée vide.

Afin de construire une ontologie à partir de corpus spécialisés, nous avons conçu une chaîne de traîtement faisant appel à des outils de TAL et des algorithmes de post-traitement. Le schéma 3.1 illustre les différentes étapes pour parvenir à la construction d'un graphe de corpus (étapes 1 à 4). Notre méthode part de l'arbre syntaxique pour extraire les nœuds et les arêtes du graphe [2], qui forment des triplets, pour aller vers un graphe de phrase (étape 2), un graphe de document pour lequel les graphes de phrases sont fusionnées grâce aux chaînes de coréférences (étape 3), et enfin, un graphe de corpus (étape 4), contenant les connaissances extraites du corpus entier. Le graphe obtenu n'est constitué que de nœuds non hiérarchisés et de relations entre ces nœuds. Mais en parallèle de ce processus, une taxonomie des classes (ou concepts) est construite à partir des entités du graphe (étapes 5 à 6). L'étape 5 vise à regrouper des instances statistiquement importantes sous une même classe. Finalement, la taxonomie (6) est fusionnée avec le graphe de corpus (4) pour constituer une ontologie.

Le schéma 3.2 montre l'ordre d'exécution des différents processus pour parvenir à l'ontologie. Une première étape permet de n'extraire que des classes et des relations entre ces classes à partir d'une description textuelle, ou spécification, en langue naturelle contrôlée. Il s'agit d'un prompt descriptif [3] que l'utilisateur rédige lui-même et qui a pour but de créer une taxonomie de base en imposant une liste de

1. Fait de transformer un prédicat en entité.

2. Les nœuds, ou entités, sont des regroupements de tokens sous un même groupe nominal, dont la tête est un nom commun, un nom propre ou un pronom. Les arêtes, ou relations, sont des groupes verbaux dont la tête est un verbe ou un auxiliaire.

3. Ce prompt est transformé en un ensemble de triplets par notre système et les triplets extraits sont interprétés comme des classes directement. Aucune instance n'est créée.

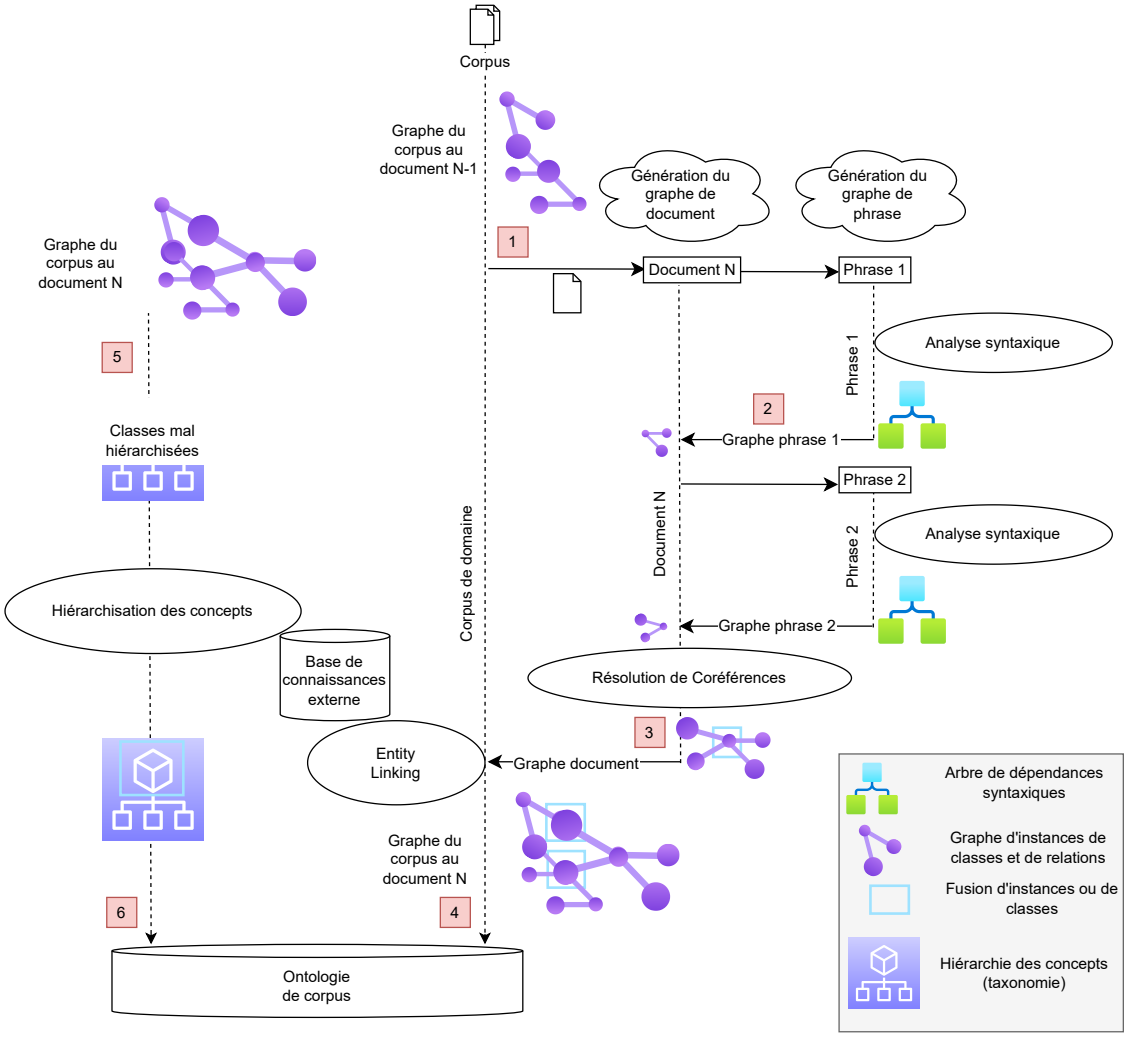

FIGURE 3.1 – Schéma du système de fonctionnement de la génération d'ontologie à partir de corpus. Étapes 1 à 4 : extraction des triplets et du graphe de connaissances ; étape 5 : extraction de concepts ; étape 6 : hiérarchisation des concepts à partir des entités.

classes, avec la possibilité de définir des relations hiérarchiques (par exemple : <incendie, rdfs:subClassOf, catastrophe naturelle>) ou non hiérarchiques (par exemple : <homme, pouvoir causer, incendie>) entre elles. Une fois ce préambule analysé, l'ensemble du corpus est présenté au système, duquel on extrait le graphe de corpus, constitué d'entités et de relations. L'étape optionnelle d'enrichissement de concepts correspond au référencement de nos entités avec des entités d'une ontologie externe, pouvant fournir des informations complémentaires à notre ontologie et aider à la fusion des classes et à la hiérarchisation, étape essentielle pour créer l'ontologie. Nous avons imaginé pouvoir enrichir les concepts avec différentes méthodes, via une intervention de l'utilisateur qui ajouterait des définitions aux concepts comme dans [Wächter et al., 2011], qui aboutit à une actualisation incrémentale de l'état de l'ontologie. Cela n'a pas été testé mais devrait théoriquement fonctionner. L'aspect « mise à jour de l'ontologie » est un des principes du projet KESAIO 2.2 et pourra être exploré davantage à l'avenir.

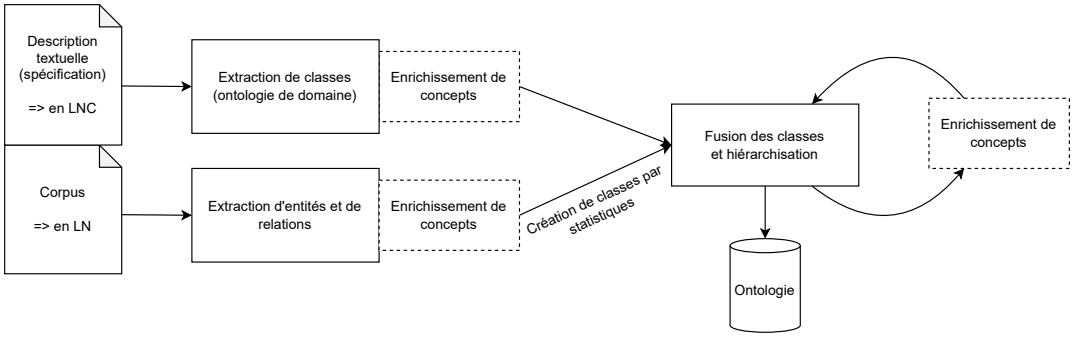

FIGURE 3.2 – Schéma du fonctionnement global du système d'apprentissage d'ontologies proposé.

3.2 Conversion de l'arbre de dépendances en triplets

Dans cette section, nous décrivons l'algorithme qui a permis de transformer l'arbre de dépendances syntaxiques en triplets, autrement dit, comment passer d'un graphe entre les tokens d'une phrase à un graphe de connaissances d'un degré supérieur d'abstraction.

3.2.1 EUD, UD et SUD et leur rôle ontologique

Chaque phrase du document en cours est traitée par un analyseur syntaxique l'une après l'autre. Les relations syntaxiques sont des relations entre deux tokens, appelés « tête » et « dépendant ». Nous avons choisi SpaCy [Honnibal and Montani, 2017] et son modèle transformers version 3.8.0 [4] car c'est le modèle le plus performant et open-source que nous avons testé (parmi les modèles testés : AiruNer, interne à Airudit et Stanza[Qi et al., 2020a]).

Transformation du graphe

L'objectif de cette partie est de convertir l'arbre de dépendances en triplets, pour créer un graphe de connaissances. La différence entre un graphe de connaissances et une ontologie est que l'ontologie donne un sens aux nœuds du graphe de connaissances en les classifiant et en les ancrant dans une taxonomie, tandis que le graphe de connaissances n'est qu'un ensemble de nœuds et d'arêtes. Les graphes de connaissances sont décrits dans [Hogan et al., 2021]. Dans cette partie, nous allons répondre à la problématique suivante : **comment les relations syntaxiques UD peuvent-elles être exploitées dans l'extraction d'informations en contexte ouvert pour un corpus en français ?** Nous allons passer en revue chacune d'entre elles et décrire brièvement leurs rôles dans la phrase et pour l'ontologie. À noter que certaines relations ont été ajoutées au schéma de l'ontologie :

— *hasSubject* pour le sujet du verbe,
— *hasObject* pour les arguments du verbe sauf adverbe,
— *hasAdverb* pour les adverbes,
— *hasModifier* pour les modifieurs de noms,
— *hasDeterminant* pour les déterminants du nom,
— *hasAdposition* pour les adpositions (ou prépositions),

4. fr_dep_news_trf version 3.8.0, fine-tuné à partir de CamemBERT.

Relation Ontologique	Relation UD
hasSubject	*nsubj, csubj, amod* (dans certains cas). *nsubj:pass, csubj:pass, expl:pass, obj:agent* (voix passive).
hasObject	*obj, obl:arg, obl:mod, iobj, iobj:agent, ccomp, xcomp* (dans certains cas), *cop* (dans certains cas). *nsubj:pass, csubj:pass, expl:pass, obj:agent* (voix active).
hasAdverb	*advmod*
hasDeterminant	*det*
hasConj	*conj*. La forme de la conjonction reliée par *cc* (et, ou...) devient une relation entre les membres de la conjonction.
hasAdposition	*case*
hasModifier	*amod, nmod, nummod, advmod, acl* (dans certains cas)
hasHead, *rdfs:subClassOf*	relation entre le groupe nominal et sa tête.
LogicalRelation	*advcl*, lorsqu'utilisés avec : *mark, case*
skos:altLabel	*appos*
dépendant du contexte	*acl:relcl, acl* : dépendent de la relation vers le pronom relatif.
Dans une entité (groupe nominal)	*det, nmod, amod, nummod, advmod, fixed, flat:name, flat:foreign, goeswith*
Dans une relation (groupe verbal)	*adv, aux, aux:caus, aux:tense, aux:pass, expl:pass, fixed, goeswith, obj:lvc* (SUD), *xcomp* (dans certains cas), *cop* (dans certains cas)

TABLE 3.1 – Tableau simplifié des correspondances entre le rôle ontologique (colonne de gauche) relations UD (colonne de droite). La version détaillée pour chaque relation UD est disponible en annexe A. Dans ce tableau, toutes les relations ontologiques en *"has"* et *LogicalRelation* ont été ajoutées au schéma de notre ontologie, tandis que *rdfs :subClassOf* et *skos :altLabel* proviennent de schémas largement utilisés en ingénierie des connaissances.

— *hasConj* entre les membres d'une conjonction.

Et des super-relation pour la logique :

— *Conjunction* pour préciser la conjonction de coordination entre les membres d'une conjonction,
— *LogicalRelation* pour préciser le marqueur logique,
— *Comparison* pour les superlatifs (le plus, le moins).

Les deux premières relations, *hasSubject* et *hasObject* ont été ajoutés afin de résoudre le problème des prédicats imbriqués et des arguments multiples du verbe. Elles correspondent au processus de réification, dont nous reparlerons partie 3.2.2. Le tableau 3.1 résume les transformation des relations UD en relations ontologiques. La liste des relations UD et leur rôle dans la phrase et pour l'ontologie est davantage détaillée annexe A.

Pour résumer, pour l'extraction des triplets <sujet, prédicat, objet>, deux grandes catégories de relations se dégagent. Les relations *ccomp*, *iobj*, *iobj:agent*, *obj:agent*, *obl:arg*, *obl:mod*, *xcomp*, *nsubj:pass*, *csubj:pass* sont transformées en relation *hasObject* entre leur tête et leur dépendant. Les relations *csubj*, *nsubj*, *nsubj:caus*, *obl:agent*

(causatif), *iobj:agent* (causatif) sont transformées en relation *hasSubject*. Pour la voix passive, deux triplets sont générés pour changer la voix : un triplet à la voix passive, et un à la voix active (à l'envers et sans le verbe être du passif). Toutes ces relations permettent de faire intéragir les entités entre elles et d'obtenir une ébauche de graphe de connaissances.

Conversion des relations UD vers SUD et EUD

Afin de simplifier l'extraction d'informations à partir de l'arbre syntaxique, nous avons effectué des conversions partielles depuis UD (*Universal Dependencies*) vers SUD (*Surface Syntactic Universal Dependencies*, voir [Gerdes et al., 2018]) et EUD (*Enhanced Universal Dependencies*, voir [Schuster and Manning, 2016]). Il s'agit de deux formalismes d'annotation en dépendances syntaxiques dont la conversion à partir d'UD est dépendante du langage.

SUD. Pour les conversions en SUD, nous avons remarqué que la relation *obl:lvc* est absente dans UD et donc dans la plupart des analyseurs syntaxiques performants. Cette relation est introduite par SUD et a une utilité pour la construction d'ontologies, puisqu'elle permet de distinguer les locutions verbales telles que « rendre visite » des structures verbe + objet direct. En effet, dans la locution « rendre visite », le verbe « rendre » ne veut pas dire « rendre quelque chose », il a un sens propre et donc devrait être considéré comme un prédicat à part entière dans l'ontologie.

Ensuite, dans la relation *cop* d'UD, l'objet du verbe « être » est considéré comme sa tête, peu importe si l'objet est un nom ou un adjectif, alors que pour tous les autres verbes et auxiliaires, c'est l'objet qui est dépendant du verbe. Pour des questions de cohérence, il est nécessaire d'inverser cette relation, tout en la renommant *comp:pred*, comme recommandé dans le guide de conversion de SUD, en prenant soin de transférer les autres dépendants et la tête de l'objet vers le verbe être.

Nous n'avons pas converti toutes les relations en SUD, uniquement celles qui sont gênantes pour la compréhension de la phrase d'un point de vue ontologique. Par exemple, bien que *nummod* (modifieur numérique) ait un rôle syntaxique similaire à *det* (déterminant) et bien qu'il ait été supprimé en SUD au profit de la relation *det*, il n'est pas gênant de le distinguer de *det*. Au contraire, il a un rôle dans l'ontologie, comme par exemple pour savoir combien d'instances du nom existent : dans l'expression « 7 passagers », nous pouvons choisir d'instancier un groupe de sept passagers directement (en tant que concept) ou de créer sept instances de passagers. Les déterminants sont, quant à eux, ignorés.

EUD. Enhanced Universal Dependencies (EUD) est une version plus proche de la conception ontologique des relations. [Heinecke, 2020] décrit EUD comme crucial pour le TAL à base d'exploitation d'arbres syntaxiques :

> « EUDs provide syntactic information which can be crucial for any NLP processing based on syntactic analysis. »

Par exemple, une des opérations d'EUD vise à relier tous les membres d'une conjonction aux attributs de la tête de la conjonction. Une conversion de conjonction vers EUD est illustrée figure 3.3.

Après conversion, les deux membres de la conjonction sont à profondeur égale dans l'arbre, ils ont donc la même influence dans la phrase, alors qu'avant la conversion, le deuxième membre (et les suivants dans le cas d'une énumération plus longue) sont dépendants du premier membre et ne sont pas reliés au sujet. Il s'agit de la plus grande modification apportée à l'arbre syntaxique : en effet, avec l'introduction de

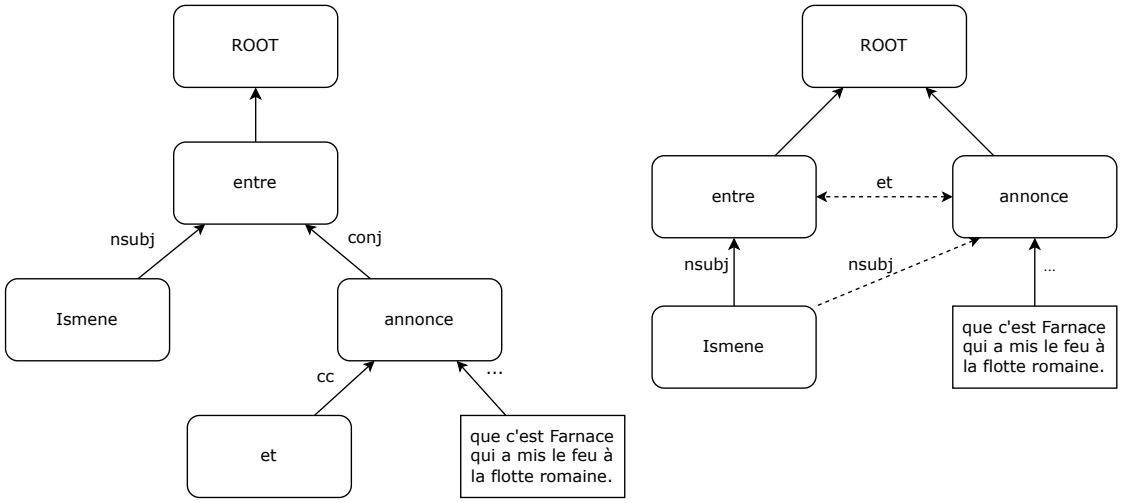

FIGURE 3.3 – Exemple de modification de l'arbre syntaxique pour un partage de sujet par deux verbres membres d'une conjonction dans la phrase « Ismene entre et annonce que c'est Farnace qui a mis le feu à la flotte romaine. » (source : corpus French-GSD). À gauche, la version UD ; à droite, la version EUD, qui relie le sujet « Ismene » au deuxième verbe de la conjonction et utilise la conjonction « et » comme relation bilatérale entre ses membes.

cette conversion, l'arbre, qui est un graphe acyclique orienté, peut devenir un graphe cyclique orienté, puisqu'un token peut désormais avoir plusieurs têtes avec des dépendances différentes. Cela peut causer quelques difficultés algorithmiques, mais permet à l'extraction de triplets de fonctionner sans se préoccuper de la résolution des conjonctions (i.e. récupérer les membres dépendant de la tête de la conjonction) à chaque étape de l'extraction. Cependant, cette conversion n'est pas un problème résolu à l'heure actuelle et elle génère donc quelques erreurs dans le partage des arguments du verbe [Heinecke, 2020].

La résolution des propositions relatives (*acl:relcl*) est aussi proposée par EUD. Le verbe principal de la proposition relative prend un nouvel argument, qui était à l'origine la tête du verbe principal de la proposition, reliée par la relation *acl:relcl*. La relation utilisée est la relation reliant le pronom relatif à la phrase. Un algorithme récursif permet de le retrouver, car il n'est pas toujours dépendant du verbe principal de la proposition. Un exemple de cette conversion est montré figure 3.4.

D'autres conversions existent dans EUD. Un exemple est la résolution des relations *orphan*, dont l'objectif est de retrouver les termes ellipsés dans la phrase pour compléter la structure grammaticale. Toutes les conversions EUD sont utiles pour l'ontologie et, à terme, si un outil de conversion d'UD vers EUD fiable émerge pour le français, l'utiliser pourrait améliorer la qualité de l'extraction de triplets.

3.2.2 Problème de compatibilité entre ontologie et langue naturelle

Réification des verbes

La réification des verbes est une étape qui permet de considérer les prédicats comme des entités et non plus comme des prédicats au sens ontologique du terme, c'est-à-dire comme une relation entre deux entités. Cette étape est nécessaire pour faire le pont entre langue naturelle et ontologie. En effet, en langue naturelle, il n'y a

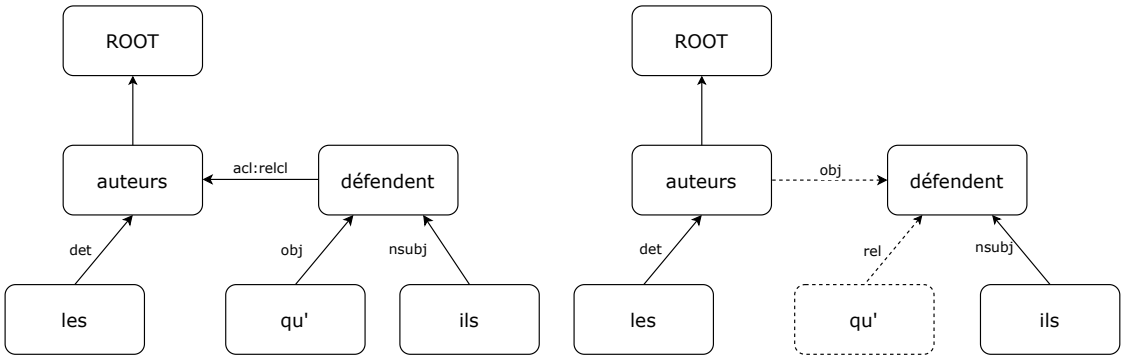

FIGURE 3.4 – Schéma du graphe d'une phrase avant (UD) et après (EUD) la résolution de la proposition relative dans la phrase « [...] les auteurs qu'ils défendent [...] » (source : corpus French-GSD). La relation *obj* est transférée depuis le pronom relatif (« qu' ») vers la tête de la proposition relative (« auteurs »). Le triplet qui sera alors extrait est <ils, défendent, les auteurs>.

pas de contraintes quant à la structure des phrases : il est possible d'enchaîner plusieurs verbes (relation *xcomp*), et un verbe peut servir de pivot à plus d'une information en plus de l'objet direct, par exemple une information spatiale, temporelle, un objet indirect, un adverbe qui modifie sa valeur, etc. Certains chercheurs recommandent d'utiliser la réification sur les « faits », comme dans l'article [Nguyen et al., 2014], dans lequel un fait est relié à un sujet, un prédicat et un objet par des relations héritant de *rdf:subject*, *rdf:predicate* et *rdf:object*. Puis, ce fait est relié à une source d'information et une date d'extraction du fait. Dans notre système, ce n'est pas cette structure qui a été choisie, car les triplets extraits ne sont pas uniquement des faits <sujet, prédicat, objet>, mais aussi <sujet, prédicat, complément circonstanciel (de temps, de lieu...)>, qui sont généralement les arguments obliques du verbe. Au lieu de cela, nous avons choisi le prédicat comme pivot pour tous ses arguments (sujet(s), objet(s), circonstanciel(s), ...), comme c'est le cas en langue naturelle, et comme recommandé par [Schutz and Buitelaar, 2005]. Pour distinguer les types d'arguments entre eux, nous avons gardé les adpositions avant les objets afin d'aider le raisonneur à comprendre lui-même de quel type d'objet il s'agit, par exemple l'adposition « dans » + un complément circonstanciel de lieu sera annoté de la sorte :

— Assertion :
  « Le petit chat dort dans la pièce. »
— Après extraction de triplets :
  <le petit chat, dort [dans], la pièce>
— Après réification :
  <dort, *hasSubject*, le petit chat>
  <dort, *hasObject*, dans>
  <dans, *isAdpositionOf*, la pièce>

Contrairement à un arbre syntaxique en UD, l'adposition n'est plus le dépendant du nom qu'elle introduit, mais elle est au-dessus, facilitant le déplacement dans le graphe. Garder « dans » comme nœud intermédiaire entre le sujet et le complément de lieu est justifié par l'usage questions-réponses que nous voulons faire de l'ontologie. En effet, si la question posée concerne un endroit, le modèle en charge de trouver la réponse pourra chercher à se déplacer dans le graphe en direction des adpositions

qu'il juge être spatiales, telles que « à », « en », « dans », « sur », « sous ». Un prétraitement sur le corpus ou un post-traitement sur l'ontologie permettrait de classifier les entités en tant que lieu, durée, etc., et la relation *hasObject* unique pour tous les arguments du verbe sauf le sujet et l'adverbe pourra être affinée avec des relations plus précises telles que *location*, *duration*, *beginDate*, etc. en fonction du type de l'argument. Nous n'avons pas été jusqu'à ce niveau de compréhension dans le cadre de ce mémoire, sachant que le référencement d'ontologies (3.4) permet aussi de savoir à quel type d'entité on a affaire.

D'autres relations ont été introduites dans le schéma de l'ontologie (voir annexe A.7), comme *hasHead*, *hasModifier* ou *hasDeterminant* pour les structures internes des groupes nominaux :

— <le petit chat, *hasHead*, chat>
— <le petit chat, *hasModifier*, petit>
— <le petit chat, *hasDeterminant*, le>

La version dé-réifiée des verbes a été conservée grâce aux *Annotation Properties*, un moyen plus libre d'enregistrer des triplets, mais qui n'est pas pris en compte par le raisonneur. Voici un exemple extrait du corpus Catastrophes : <usine pétrochimique Jilin Chine, Entraîner, pollution fleuve Songhua>, où "Entraîner", une *Annotation Property*, est la réalisation du concept "entraîner". Un autre exemple de graphe de phrase constitué de triplets réifiés avec la version dé-réifiée en légende est illustré en annexe A.4.

Un autre problème rencontré est que certaines entités n'ont de lien entre elles que le fait qu'elles apparaissent dans le même document, puisqu'elles se retrouvent dans des phrases isolées, sans prédicat ni chaîne de coréférence. Nous avons alors introduit une relation *ontology:ExtractedFrom*, avec un identifiant unique pour chaque document d'origine. La figure 3.5 montre le diagramme Entité/Association de la représentation utilisée dans l'ontologie. Chaque entité ou relation provenant du corpus est en réalité une instance [5] de *DocumentNode*, et chaque *DocumentNode* a un *Document* d'origine. De plus, chaque instance de *Document* a un titre, une date d'extraction et un identifiant unique, qui constituent ses métadonnées. D'autres métadonnées peuvent être enregistrées en fonction du corpus et des besoins (auteur, date de publication, source...).

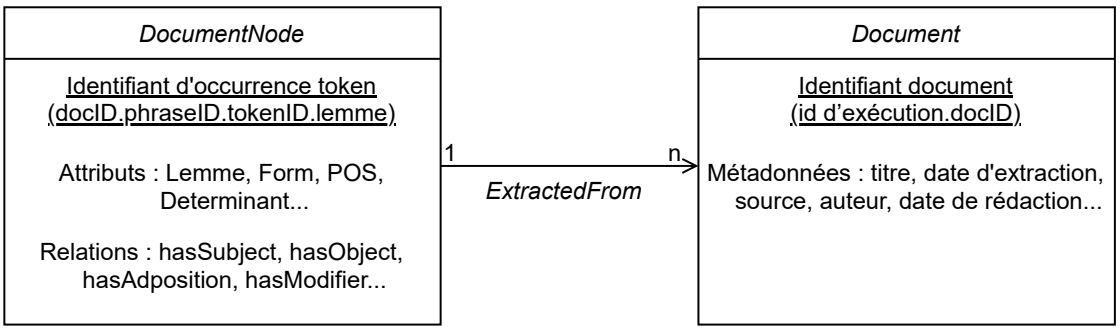

FIGURE 3.5 – Diagramme Entité/Association de la structure *DocumentNode* et *Document* utilisée dans notre ontologie. Chaque entité occurrant dans un document est une instance de *DocumentNode* reliée à un seul *Document*.

5. Pour instancer une classe, on utilise la relation *rdf:type* de RDF.

3.3 Découverte de la taxonomie et conceptualisation

Une taxonomie est une hiérarchie de classes, sans instances, servant à donner des relations parent-enfant à des concepts généraux ou d'un domaine particulier. Elle ne se compose que de triplets <A, est un, B>, qui sont des relations d'hyperonymie/hyponymie. La taxonomie, en sciences de l'information, est inspirée de la taxonomie en biologie qui classifie le vivant. La taxonomie sert à inférer le fait que si tel concept hérite d'un autre concept, il hérite aussi de toutes ses caractéristiques et doit être traîté de la même manière. Par exemple, si l'on sait que le jus d'orange est une sous-catégorie de jus de fruit, et que « tous les jus de fruits sont conservés au frais », alors le jus d'orange, qui hérite des propriétés du concept de jus de fruit, est aussi conservé au frais. Dans le cas d'un système de questions-réponses, la question « comment conserver le jus d'orange ? » aura alors une réponse grâce à cette inférence.

Pour la découverte de la taxonomie à partir des instances de classes (*Taxonomy Discovery*), on manipule directement les nœuds du graphe de connaissances pour décider ou non de leur créer une classe (conceptualisation), et si oui, on décide où placer cette classe dans la taxonomie (découverte de la taxonomie). La découverte de la taxonomie se base sur les concepts et les relations entre eux sous forme de graphe, mais pourrait aussi se baser davantage sur des ressources extérieures, comme la ressource dictionnairique GLAWI qui permet d'enrichir la plupart des concepts généraux en synonymes, antonymes, définitions et traductions, ou encore Wikidata, une ontologie possédant déjà une taxonomie qu'il est possible de récupérer. C'est ce que fait [Wächter et al., 2011], qui réutilise la taxonomie d'une autre ressource, en plus de se baser sur des motifs de type « A is a B with property C ». Dans le graphe syntaxique, il faut savoir que grâce à la résolution des coréférences (voir section 3.4.1), chaque « concept » a déjà plusieurs variantes orthographiques. En se basant à la fois sur l'arbre syntaxique, la structure du graphe autour d'un concept et sur des ressources dictionnairiques (GLAWI) et ontologiques (Wikidata), plusieurs stratégies de hiérarchisation peuvent être mises en place, mais nous ne les avons pas toutes explorées dans ce mémoire.

Une stratégie possible de découverte de taxonomie est de partir de quelques concepts prédéfinis au sommet de la taxonomie pour en dériver les classes du corpus au fur et à mesure que le graphe s'agrandit. En adoptant cette approche, les entités nommées reconnues par un modèle de détection des entités nommées (Flair ou SpaCy par exemple) peuvent être exploitées comme sommet provisoire de la taxonomie, en choisissant d'utiliser les classes d'entités nommées comme classes racines de la taxonomie [6] ; cependant les résultats de ces deux modèles de détection d'entités nommées sur le corpus Solaris ne sont pas suffisamment précis pour baser le sommet d'une ontologie sur ces classes d'entités nommées. Il est aussi possible d'entraîner un modèle à reconnaître des entités nommées d'un corpus de spécialité avec ses propres classes, ce que nous n'avons pas fait car nous préférons une méthode générale et non supervisée, qui fonctionne hors ligne et avec une puissance de calcul limitée.

Au lieu de cela, pour générer les classes racines, nous laissons le choix à l'utilisateur de créer une taxonomie de base qui servira à dériver les concepts du corpus. Pour ce faire, l'utilisateur peut rédiger une liste de mots ou une brève description du fonctionnement de son domaine en langue naturelle contrôlée. Par exemple, pour le corpus Catastrophes, nous avons créé un prompt descriptif, qui liste et décrit les types de catastrophes et d'autres concepts connexes, voir annexe A.3. Grâce à l'algorithme

6. Par défaut on trouve les quatre classes suivantes : personne, organisation, lieu et divers.

qui sera présenté ci-dessous, ce prompt produit la taxonomie présentée annexe A.5. D'autres propriétés sont extraites, tels que <homme, PouvoirCauser, feu> et <séisme, Avoir, magnitude>. Les relations d'équivalence proviennent de la relation *appos* et peuvent être séparées ou fusionnées en post-traitement.

3.3.1 À partir de l'arbre syntaxique

Nous avons déjà vu que plusieurs relations syntaxiques d'UD dénotaient des relations hiérarchiques. Parmi celles-ci, il y a la relation *nmod* (noun modifier), qui fait référence aux compléments du nom en grammaire conventionnelle, et la relation *amod*, qui indique un adjectif modifiant un nom.

Exploiter ces relations permet de créer des dérivations de classe. La racine du groupe nominal devient ainsi la classe parent, et le groupe nominal avec ses modifieurs devient la sous-classe. Par exemple, « matériel informatique » est une sorte de « matériel », il est donc pertinent de le faire hériter de « matériel », même si aucune occurrence de « matériel » en tant qu'expression monolexicale n'existe dans le corpus. Si un autre type de matériel apparaît lors d'une mise à jour de l'ontologie postérieure, comme « matériel bureautique », il sera par défaut la classe sœur de « matériel informatique ».

3.3.2 Avec des motifs de phrases

Il est possible, dans un corpus, de rencontrer directement des définitions des concepts. Ces définitions servent à situer un concept inconnu par rapport à un concept connu pour le lecteur, en fournissant des synonymes, des classes parentes, etc. C'est le cas des phrases suivant une formulation en « A est un B », « A est une sorte de B », etc [Prakash et al., 2021].

Nous effectuons un post-traîtement des triplets pour détecter les triplets de type <Sujet, être, Objet> sans adposition devant l'objet. Ces triplets sont exploités pour la génération de la taxonomie : Objet devient la superclasse de Sujet, car Sujet est un sous-type d'Objet. Notons que, grâce à l'extraction de triplets et contrairement à une approche basée sur les expressions régulières, il suffit de regarder le lemme du verbe, et les mots tels que « sorte de », « type de » ou encore « genre de », « espèce de », n'influencent pas l'extraction de la relation d'hyperonymie. Par exemple, dans la phrase issue du corpus GSD « Le Monde médiatique est une sorte de théâtre [...] », « sorte de théâtre » deviendrait la superclasse de « monde médiatique ».

L'autre condition est que le sujet et l'objet doivent avoir une partie du discours compatible avec cette approche, c'est-à-dire un nom commun pour l'objet, et un nom commun ou un nom propre pour le sujet. Il arrive que l'objet soit un adjectif, comme dans « la voiture est bleue », dans ce cas il faut réussir à modéliser le fait que l'objet est un modifieur du sujet [7]. La dernière condition est que l'objet doit être précédé d'un déterminant indéfini, car un déterminant défini ne montre pas la même intention. Un déterminant défini devant l'objet peut être une marque du superlatif (« le plus », « le moins »). Les relations superlatives (voir figure A.7) ont été extraites de cette façon. Un déterminant indéfini affiche souvent une intention de définir un terme pour le classifier, comme dans la structure « A est un B ».

Pour finir, la définition d'un terme commence souvent par donner sa superclasse. Par exemple, si l'on prend la définition de « volcan », on trouve : « Un volcan est une

7. Pour ce faire, nous utilisons notre relation *hasModifier* pour relier le sujet à l'objet du verbe être : <la voiture, *hasModifier*, bleue>.

structure géologique [...] » (source : Wikipédia), ou « Montagne qui émet ou a émis des matières en fusion. » (source : dictionnaire Le Robert). En exploitant ces définitions, il est possible de déduire que les classes parentes de « volcan » sont « montagne » et « structure géologique ». Pour aller plus loin, il est possible de continuer de chercher une classe parente à « montagne », qui est elle-même une structure géologique, afin d'augmenter la taxonomie d'un niveau supplémentaire.

En conclusion, grâce à l'exploitation de la structure « A est un B avec la propriété C », nous pouvons non seulement extraire une taxonomie à partir du corpus, mais également venir l'enrichir grâce à des définitions trouvées dans des ressources extérieures ou fournies par l'utilisateur. Cette étape correspond à l' « enrichissement des concepts » incrémentale présenté sur le schéma 3.2.

3.3.3 Découverte des concepts importants

La population de l'ontologie, soit le fait d'ajouter des instances à une classe, s'oppose à la conceptualisation, le fait de créer une classe à partir d'instances. La population de l'ontologie nécessite d'avoir un panel de classes à disposition, puis, il suffit d'identifier le concept auquel se réfère un token ou une expression du corpus avec un modèle de classification ou d'autres méthodes d'*Entity Linking*. Il est aussi possible de simplement se baser sur la forme du mot dans un premier temps. Pour la conceptualisation, la tâche est plus compliquée, puisqu'en plus de devoir regrouper des termes entre eux, il faut aussi en dégager un concept commun. Cette tâche demandant beaucoup d'efforts [Fisher, 1987], nous nous sommes concentrés sur l'extraction de concepts significatifs du point de vue du corpus. En effet, tous les termes d'un corpus n'ont pas forcément besoin d'être transformés en classes et peuvent exister dans le graphe de connaissances comme nœuds non classifiés.

L'utilité de créer ces concepts pour regrouper les instances est d'utiliser ces classes comme pivot dans un système de questions-réponses. Par exemple, si nous demandons « Quels volcans sont entrés en éruption entre 2010 et 2020 ? », il est plus simple d'identifier une liste de volcans si les instances de volcans sont conceptualisées sous une classe commune. Cependant, nous ne pouvons pas transformer chaque terme en classe, le nombre de concepts serait alors trop important dans le cas d'un corpus large. Dans cette sous-partie, nous avons défini quatre critères de sélection de termes qui formeront des classes : le score de spécificité minimal du terme dans le corpus par rapport à un corpus plus large ; la fréquence token minimale (tf min) ; la fréquence token minimale relative (tf min relative) et la fréquence document minimale (df min), le dernier critère jouant le rôle le plus discriminatoire parmi les quatre.

Score de spécificité minimal

Afin d'identifier les individus de l'ontologie qui méritent d'être conceptualisés (i.e. transformés en concepts de l'ontologie), nous calculons un indice de spécificité [8], qui mesure à quel point un terme de notre corpus est spécifique par rapport à la langue concernée grâce à un corpus plus large dont notre corpus est un sous-corpus. L'indice de spécificité se fonde sur la loi hypergéométrique :

$$P(X = k) = \frac{\binom{n}{k}\binom{M-n}{N-k}}{\binom{M}{N}}$$

8. Calculé de la même manière que l'indice de spécificité de TXM.

où, pour un terme candidat, on a :
— n, le nombre de tokens dans le corpus ;
— k, le nombre d'occurrences du terme dans le corpus ;
— M, le nombre total de tokens dans le corpus plus large ;
— N, le nombre d'occurrences du terme dans le corpus plus large.

Nous utilisons une fonction de distribution cumulative, qui indique qu'en dessous d'un certain seuil se situent x % des valeurs. Ainsi, pour ne garder que les termes qui sont parmi les 1 % des termes les plus spécifiques, on élimine ceux dont la spécificité est en dessous du seuil de 0,99.

En général, un score de spécificité se mesure sur un sous-corpus par rapport à un corpus l'englobant. On dit que le corpus plus grand est représentatif du genre, et que le sous-corpus possède des termes spécificiques. Afin d'éviter de recréer un corpus, nous utilisons un dictionnaire de fréquence des termes du français, le Lexique v3.86 [New et al., 2004], dans lequel les termes du français sont classés par lemmes et parties du discours. Des fréquences relatives d'occurrence sont calculées à partir de quelques 50 millions de tokens, et sont indiquées en nombre d'occurrences pour un million de tokens, ainsi dans la formule ci-dessus, nous avons N = fréquence relative d'occurrence du terme recherché dans la ressource Lexique v3.86, M = 1 million.

Fréquence token minimale

Dans le lexique (ou corpus de référence), il se peut que la fréquence relative d'un mot soit si faible (< 1 pour 1 million) qu'une seule occurrence dans notre sous-corpus suffise à dépasser le seuil de spécificité. C'est pour cela que nous devons fixer, pour les termes présents dans le lexique, un seuil fixe de fréquence de token, par exemple le token doit apparaître minimum 3 fois dans tout le corpus. Ce seuil n'est pas pris en compte lorsque la fréquence document minimale est supérieure à la fréquence token minimale.

Fréquence token minimale relative

Lorsqu'il s'agit d'expressions hors vocabulaire (ou OOV en anglais pour *out-of-vocab*), qui sont absentes du corpus de référence ou du lexique, nous n'avons d'autre choix que d'observer leur fréquence relative par rapport à la taille de notre corpus. Par exemple, si le terme représente 0,01 % du corpus, alors il est important et doit être conceptualisé. Ce seuil doit varier en fonction de la taille du corpus. Sur un corpus de 50 000 tokens, comme la version courte du corpus Catastrophes, le seuil de 0,01 % impose que les mots OOV apparaissent 5 fois, ce qui est relativemant haut dans le cas d'un corpus varié qui n'a qu'une dizaine de documents par sujet. Souvent, les termes OOV sont des expressions polylexicales qui ont été regroupées lors de l'étape d'extraction des triplets en exploitant les relations syntaxiques, voir 3.2. Ces derniers ne sont pas conceptualisés, car il s'agit souvent d'entités nommées, et devraient par la suite pouvoir être instanciés (instance de personne, de lieu...).

Fréquence document minimale

En fonction du nombre de documents dans notre corpus, il peut être intéressant de fixer un seuil de fréquence document minimale (df min) à atteindre. La fréquence document se compte en « dans combien de documents apparaît l'expression ». Pour un corpus monodocument, on peut effectuer un découpage en paragraphes ou en parties.

Dans le cas d'un corpus dont les sujets sont variés, augmenter le df min permettra de détecter des concepts récurrents, communs à plusieurs sujets.

3.4 Liage d'entités interne et externe

Le référencement d'ontologies (*Ontology Referencing*) et le liage d'entités (*Entity Linking*) servent soit à relier des concepts d'une ontologie de domaine à ceux d'une ontologie générale (externe), soit à relier des concepts internes à notre ontologie entre eux (interne). Ce processus remplit deux fonctions principales. La première, est qu'il permet de désambiguïser des homonymes et regrouper des synonymes. Si deux classes distinctes sont en réalité synonymes, comme « survivant » et « rescapé » dans le corpus Catastrophes, alors elles sont reliées à la même classe de Wikidata (ici *Q6136036*), nous savons dès lors qu'il faut les fusionner. La seconde est que la ressource externe apporte des connaissances utiles pour la compréhension des concepts : définitions, structure, voisinnage, ascendants hiérarchiques, etc.

3.4.1 Avec les coréférences et l'arbre syntaxique

Chaînes de coréférences

La résolution des coréférences est une étape nécessaire pour l'extraction et le liage d'entités [Sukthanker et al., 2018]. Elle permet de relier différentes mentions d'une entité dans un document pour en faire une entité unique. Au sein d'un corpus, soit sans faire appel à une ressource extérieure, une liaison d'équivalence entre deux termes peut ainsi être déduite. Ces liaisons sont appelées des chaînes de coréférences. Les chaînes de coréférence sont une suite de pronoms personnels (je, tu, il...) ou relatifs (qui, que, quoi...), de déterminants possessifs (mon, ta, ses...), et de groupes nominaux (dont la tête est un nom commun ou un nom propre) qui font référence à une seule et même entité. Nous proposons trois étapes pour résoudre les coréférences.

D'après [Joshi et al., 2019], ré-entraîner un modèle de langue BERT à résoudre des coréférences est plutôt efficace. Dans leur étude, les chercheurs ont fine-tuné un modèle BERT pour lui apprendre à renvoyer tous les tokens qui sont coréférés sous forme de liste. Le modèle BERT-large fine-tuné a permis d'augmenter la F-mesure de 3,9 points par rapport à l'état de l'art sur le corpus OntoNotes, et de 11,5 points sur le corpus GAP, qui sont utilisés pour les tâches de résolution de coréférence en anglais. Le modèle fonctionne sur de l'anglais uniquement, mais un modèle CamemBERT a été fine-tuné pour la résolution des coréférences en français : Easter-Island/coref_classifier_ancor. Il s'agit de la solution qui fonctionne le mieux et qui est la plus simple à mettre en place, par rapport aux autres outils de résolution de coréférence sur le Français tels que DeCOFre [Grobol, 2020] sur le français parlé ; CoFR ; neuralcoref, fonctionnant avec SpaCy-experimental ; ou encore coreferee. Des solutions utilisant les LLMs sont présentées ici : [Gan et al., 2024], ou encore ici : [Zhu et al., 2024].

Nous avons choisi d'utiliser le modèle fine-tuné Camembert pour détecter les chaînes de coréférence. Comme mentionné dans l'article [Joshi et al., 2019], il est possible de dépasser la limite de caractères imposée par le modèle BERT en utilisant la technique de l'*overlapping* : on commence par découper le texte en segments qui se superposent, puis on relie les chaînes de coréférence en s'appuyant sur les tokens en commun dans la partie qui est superposée. En entrée du modèle HuggingFace, on encode un texte de 512 caractères maximum, pour lequel on a au préalable ajouté des

chevrons <> autour du terme pour lequel on veut extraire les coréférences. En sortie, le modèle renvoie une liste de tous les mots coréférés au mot entre chevrons, qu'il faut relier au texte d'origine. Malheureusement, le modèle ne renvoie que la tête des groupes nominaux, ce qui nécessite quelques post-traitements pour réunir les tokens qui font partie de la même entité d'après l'arbre syntaxique de la phrase.

Exploitation de l'arbre syntaxique

La seconde méthode consiste à exploiter l'arbre syntaxique directement. Nous avons vu que certaines relations syntaxiques avaient un rôle dans la résolution des coréférences, comme la relation UD *appos*, qui est directement transformée en chaîne de coréférence entre sa tête et son dépendant.

La résolution des propositions relatives d'EUD a pour objectif de remplacer les pronoms relatifs par l'entité à laquelle ils se réfèrent, c'est-à-dire la tête de la proposition relative. Elle est effectuée au préalable des traitements et modifie l'arbre syntaxique directement. Un exemple de modification est proposé diagramme 3.4. En revanche, le pronom relatif est « détaché » de l'arbre (remplacé par une relation neutre *rel*) et n'est pas ajouté à la chaîne de coréférences car il ne porte ni d'information sur le genre de son coréféré comme un pronom personnel, ni sur son nombre.

Une fois toutes les coréférences trouvées, nous créons des sets de tokens qui vont fusionner pour ne produire qu'une instance unique dans le graphe de connaissances. Le lemme qui est le plus informatif sémantiquement [9] est sélectionné comme label principal de l'instance, et les autres lemmes sont gardés comme labels alternatifs du concept grâce à l'annotation *skos:altLabel*. Les déterminants possessifs jouent le rôle de modifieurs : « sa fille » va générer un triplet <sa fille, *hasModifier*, Constance>, si « sa » est dans la même chaîne de coréférences que « Constance ». Si l'on avait émis l'énoncé "la fille de Constance", le triplet extrait aurait été <la fille, *nmod*, Constance>. Puisque *nmod* hérite de *hasModifier* (voir figure A.7), ces deux informations sont équivalentes pour le raisonneur.

3.4.2 Liens des entités vers d'autres ressources

GLAWI est une ressource morphosyntaxique du français assez complet, sauf qu'il ne propose pas de sets de synonymes avec un identifiant ni de liens vers Wikidata, comme c'est le cas dans WordNet, une ressource en anglais à mi-chemin entre linguistique et ontologie. GLAWI propose des synonymes, des traductions dans diverses langues, des définition et des exemples, ce qui la rend intéressante pour enrichir notre ontologie. Dans GLAWI, chaque entrée peut avoir plusieurs définitions, et chaque définition concerne un concept différent, qu'il faut choisir avec une désambiguïsation. Cependant, nous n'avons pas mise en place cette désambiguïsation sur les définitions de GLAWI, car nous avons préféré concentrer nos efforts sur la ressource plus ontologique qu'est Wikidata.

Quelques outils de liage d'entités visant à relier un concept (ou classe) à une ontologie externe ont été présentés dans l'état de l'art (3.4). D'autres outils tels que Clocq ont été testés, mais aucun d'entre eux ne permet d'interroger Wikidata en français. En absence d'un outil de référencement fonctionnant bien sur le français, nous avons décidé d'utiliser l'API de Wikidata pour interroger directement le label français des entités.

9. On garde en priorité un nom propre, un nom commun en l'absence de nom propre, ou à défaut un pronom. Puis, on sélectionne le label le plus long.

L'API de Wikidata renvoie une liste ordonnée de candidats, en fonction de la correspondance entre le label français de l'entité et le label demandé dans la requête. Nous questionnons l'API sur des mots-clés, donc les synonymes ne sont pas détectés, ni les concepts qui n'ont pas de label en français, par exemple une recherche sur le mot « matériel » ne permet pas de retrouver pas le concept Wikidata référencé par l'URI *Q111398534* (« hardware ») à l'heure où ce mémoire est écrit, car il n'est pas traduit dans Wikidata. Cette requête renvoie des instances d'armées françaises ou de matériel d'armée. Il est possible d'enrichir notre liste de mots-clés avec des synonymes ou des traductions dans d'autres langues issus de GLAWI par exemple, ce qui permettrait de renvoyer davantage de concepts candidats.

3.4.3 Désambiguïsation par *Random Walk*

L'objectif de la désambiguïsation est de relier un mot qui peut posséder plusieurs homonymes avec le bon concept. Dans le cadre de l'alignement d'entités entre deux ontologies, la problématique est similaire, sauf que l'on manipule des entités issues d'une ontologie pour la relier avec l'entité correspondante dans l'ontologie de référence.

Une fois la liste de candidats de références Wikidata récupérée, il faut choisir quel concept candidat correspond le mieux à notre entité. Par exemple, si nous avons le terme « mineur », nous voulons savoir s'il s'agit d'une personne non majeure ou d'un travailleur à la mine. Pour connaître le bon candidat, il est possible de s'arrêter au classement de l'API Wikidata obtenu précédemment. Mais nous souhaitons refaire un classement en prenant davantage en compte la similarité sémantique entre les classes de Wikidata et les instances des classes dans notre graphe de connaissances. Pour cela, nous avons décidé de mettre en place une désambiguïsation à l'aide d'embeddings et d'un algorithme de *Random Walk* [10]. Une relation *owl:equivalentClass* est ajoutée pour la classe qui obtient le plus haut score de similarité.

Il faut espérer que grâce à cette étape de désambiguïsation, nous pourrons détecter qu'aucun des concepts candidats n'est satisfaisant, avec un seuil de similarité minimal à atteindre par exemple, soit parce que le concept est absent de l'ontologie de référence, soit parce que la requête Wikidata n'a pas renvoyé le bon candidat.

Nous avons décidé de désambiguïser avec un modèle NLI (*Natural Language Inference*), plus précisément un modèle NLI multilingue (XNLI) étant donné que Wikidata est une ontologie multilingue. Un modèle NLI est entraîné à déterminer si deux phrases sont en accord, contradictoires ou ne parlent pas de la même chose. Nous avons choisi un modèle HuggingFace fine-tuné à partir de DeBERTa-V3 mDeBERTa-v3-base-xnli-multilingual-nli-2mil7. Le rôle de ce modèle est de prédire le sens d'un énoncé parmi un ensemble de clusters candidats. Ces clusters sont, dans notre cas, des classes Wikidata, pour lesquelles nous devons également trouver une représentation textuelle afin de proposer des noms de clusters pertinents au modèle XNLI. L'énoncé est, quant à lui, le résultat du *Random Walk*, soit le contexte des instances

10. Le *Random Walk* consiste en un déplacement aléatoirement dans un graphe en passant d'un nœud à l'autre de façon aléatoire. Chaque système peut décider d'une stratégie de *Random Walk* qui lui est propre. Dans notre stratégie de *Random Walk*, il est possible de se déplacer sur les relations *hasSubject*, *hasObject*, *LogicalRelation* (relations logiques, alias prépositions ou conjonctions de coordination), *hasModifier*, ou en passant par la classe, mais avec une priorité donnée aux deux premières. La direction du triplet n'a pas d'importance. L'algorithme est répété plusieurs fois (largeur) à partir d'une des instances de la classe pour obtenir plusieurs chemins, et le parcours de graphe s'arrête au bout d'un certain nombre de sauts (profondeur) ou s'il arrive dans une impasse. Un nœud ne peut pas être visité deux fois.

d'une même classe dans le graphe de connaissances. Wikidata dispose de plusieurs champs en langue naturelle qui permettent de capturer la sémantique de l'entité : un champ « label » avec le nom de l'entité dans de nombreuses langues et un champ « description » avec une brève définition de cette entité. Le champ « label » en français n'est pas discriminant, étant donné que toutes les classes candidates ont été sélectionnées sur le label français, mais utiliser ce champ dans d'autres langues pourrait aider à trouver le bon concept. Ce qui peut potentiellement servir à la désambiguïsation est le label de la classe parente. En utilisant la classe parente sur l'exemple du mineur, les deux clusters proposés au modèle seraient « profession / travailleur manuel » et « personne en vie / capacité juridique ». À l'aide d'un troisième paramètre du modèle XNLI, le prompt, nous pouvons signaler au modèle que les classes qui vont être proposées sont le sujet de ce dont le contexte parle, par exemple « Ce texte mentionne une sous-catégorie de ». Dans la partie 4.2.2, nous allons tenter d'optimiser la représentation de ces clusters et du contexte de la classe à relier pour augmenter la précision de la désambiguïsation.

CHAPITRE 4

ÉVALUATION ET RÉSULTATS

Sommaire

Introduction

Dans ce chapitre, nous évaluons différents aspects de l'ontologie à différentes étapes de sa création. Chaque section sera divisée en une partie « Expériences et métriques » qui présente la méthodologie employée, une partie « Résultats » et une partie « Conclusion », qui discute des résultats obtenus. Nous évaluons les aspects suivants : l'extraction des triplets, la sélection des classes, la sélection des candidats dans Wikidata, la désambiguïsation par *Random Walk* et la taxonomie générée à partir du corpus Catastrophes et du prompt descriptif.

4.1 Évaluation de l'extraction des connaissances

4.1.1 Évaluation des triplets extraits

Expériences et métriques

Pour évaluer la qualité des triplets extraits, la stratégie la plus simple est d'annoter chaque phrase avec les triplets que l'on attend à la main et vérifier que les triplets extraits correspondent. Cela nous permet de mesurer un rappel, une précision et une F-mesure, et d'estimer quelles sont les relations que l'outil d'analyse syntaxique a mal annotées. Pour cette expérience, nous avons choisi d'annoter le corpus Solaris en triplets en raison de sa taille limitée. Nous comparerons trois outils d'annotation : SpaCy (modèle français transformers version 3.8.0 [1]), Stanza [Qi et al., 2020b] (modèle français version 1.10.0 [2]) et Nersa, modèle interne d'Airudit. Ces trois modèles ont été entraînés sur des corpus annotés au format UD.

1. fr_dep_news_trf version 3.8.0
2. stanza-fr version 1.10.0

Outil d'annotation	Rappel	Précision	F-1
SpaCy	**0,800**	**0,825**	**0,812**
Stanza	0,452	0,445	0,449
Nersa	0,488	0,386	0,431

TABLE 4.1 – Tableau de résultats de l'extraction de triplets par les outils d'analyse syntaxique Spacy, Stanza et Nersa.

Résultats

Le tableau 4.1 donne les résultats obtenus pour les trois outils d'annotation. SpaCy est l'outil qui permet d'obtenir le meilleur rappel, précision et F-mesure. Il y a plusieurs raisons à cela. Tout d'abord, parce que les post-traitements et tests au moment du développement ont été effectués sur la sortie de SpaCy, ainsi que le corpus GSD français (annotations UD gold). Ensuite, parce qu'il existe des disparités importantes entre les annotateurs, bien que tous les trois entraînés sur UD. En effet, Stanza a tendance à utiliser des relations *appos* entre un nom commun et un nom propre, alors que SpaCy utilise plutôt une relation *nmod*. Ce problème est assez impactant pour l'ontologie. Par exemple, dans l'expression « l'outil XY », « XY » est considéré comme *appos* d' « outil », tandis que Spacy considère « XY » comme son modifieur nominal (nmod). Cela fait que Stanza crée une coréférence entre « outil » et « XY » au lieu de créer un concept nommé « outilXY » :

Stanza :

```
ontology:m_XY a ontology:DocumentNode,
        owl:NamedIndividual ,
        ontology:outil ;
    rdfs:label "XY" ;
    syntax:Form "XY" ;
    syntax:Pos "PROPN" ;
    skos:altLabel "XY",
        "outil" .
```

Spacy :

```
ontology:0.0.2.outilsXY a ontology:DocumentNode,
        owl:NamedIndividual ;
        ontology:outil ;
    rdfs:label "outil XY" ;
    ontology:ExtractedFrom ontology:0.doc ;
    syntax:Pos "NOUN" ;
    syntax:hasHead ontology:0.0.2.outil .
```

Le label choisi par Stanza est le nom propre, parmi les deux labels alternatifs (*skos:altLabel* "outil" et *skos:altLabel* "XY"). En tant que nom propre, nous le considérons comme une entité nommée unique, sans lui accorder d'index numérique correspondant à son ordre d'apparition au sein du corpus. En revanche, le concept « outilsXY » extrait par Spacy a pour tête un nom commun (NOUN) et n'a pas de labels alternatifs. En tant que nom commun, nous ne pouvons pas nous assurer qu'il est unique et qu'il n'existe pas plusieurs outils XY distincts, alors il faut lui ajouter un identifiant unique de *Document Node* 3.5. Le problème apparaît au moment où on a

deux objets différents composés d'un nom commun et d'un nom propre : « l'outil XY » et « l'onglet XY », et c'est le cas dans le corpus Solaris. Alors, l'ontologie les considère comme le même concept avec deux parents différents, qui sont « outil » et « onglet » :

Stanza :

```
ontology:m_XY a ontology:DocumentNode ,
        ontology:outil ,
        ontology:onglet ,
        owl:NamedIndividual ;
 rdfs:label "XY" ;
    syntax:Form "XY" ;
 syntax:Pos "PROPN" ;
 skos:altLabel "XY" ,
        "outil" ,
        "onglet" .
```

Conclusion

En conclusion, il vaut mieux utiliser SpaCy avec notre chaîne de traitement, mais adapter la chaîne de traitement à la sortie d'un autre outil d'annotation est toujours possible à condition de garder à l'esprit le format de triplets que l'on veut obtenir. Une des erreurs les plus récurrentes de SpaCy observée sur le corpus Solaris est le choix de la tête, surtout dans les phrases longues et avec des conjonctions imbriquées. Une relation vers la mauvaise tête engendre forcément un triplet erroné, faisant diminuer la F-mesure.

À titre de comparaison, dans [Schutz and Buitelaar, 2005], le rappel de l'extraction de triplets n'étaient pas supérieurs à 0,45 en 2005, avec une précision bien moins élevée (< 0,14), démontrant l'évolution des outils d'annotation en dépendances syntaxiques. Bien que notre méthode n'emploie pas les dernières technologies de TAL telles que les LLMs, nous sommes persuadés que la chaîne de traitement sur un arbre syntaxique permet un contrôle plus précis des connaissances extraites.

4.1.2 Évaluation de la sélection des classes

Expériences et métriques

Pour cette expérience, nous avons utilisé le corpus Catastrophes (court), qui contient un nombre élevé de tokens et permet de faire des statistiques.

Avec une liste de classes que nous aimerions voir apparaître dans l'ontologie, nous tentons de trouver les valeurs idéales de fréquence document minimale et de specificité minimale pour optimiser le rappel tout en ayant le moins de classes générées possibles. Cette liste de classes comprend le nom des types de catastrophes naturelles et industrielles, ainsi que d'autres concepts centraux tels que « secouriste », « habitant », « décès », « mine ». Nous avons dressé cette liste manuellement et de façon subjective suite à une extraction moins stricte.

Une courbe ROC (Receiver Operating Characteristic) montre la sensibilité (taux de vrais positifs) en fonction de la spécificité (taux de faux négatifs) sur plusieurs expériences, qui correspondent aux points sur la courbe.

$$sensibilité = \frac{VP}{VP + FP}$$

$$spécificité = \frac{VN}{VN + FP}$$

Elle permet de montrer à quel point un modèle de classification est efficace. Pour la tracer, on fait varier un seuil de sélection et on observe le nombre de vrais positifs et faux négatifs obtenus pour chaque seuil. Dans notre cas, le « modèle » de classification correspond aux calculs statistiques qui donnent un score aux tokens, comme le score de spécificité et le score de fréquence document minimal.

Dans notre expérience, nous avons fait varier deux seuils : le score de spécificité minimal (valeurs testées : 0,9 ; 0,95 ; 0,99 ; 0,999), et le score de fréquence document minimal (valeurs testées : 1, 3, 5, 7, 9, 11, 13, 15), qui sont les deux scores les plus discriminants sur le corpus Catastrophes. La fréquence token minimale (pour les termes non OOV) et la fréquence token minimale relative (pour les termes OOV) sont fixés à 3 et 1/10000 respectivement. Nous montrerons quelle configuration a obtenu la F-mesure la plus élevée.

Résultats

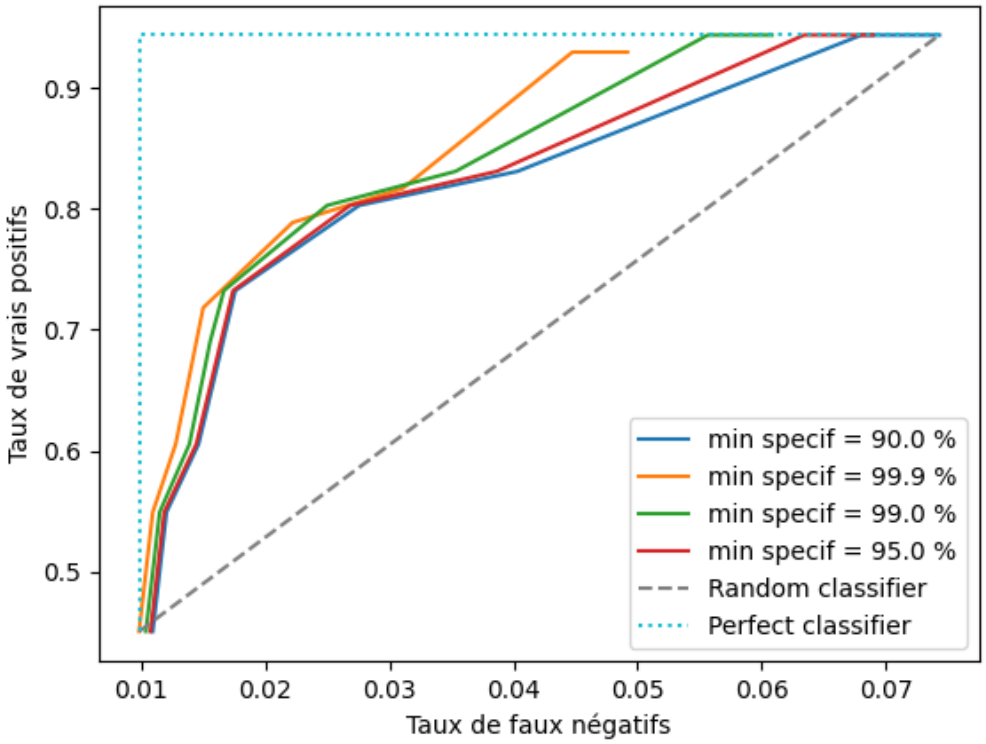

FIGURE 4.1 – Courbe ROC des concepts créés par rapport aux concepts que l'on souhaite voir apparaître selon la variation des critères de sélection (spécificité minimale et fréquence document minimale).

Les courbes ROC obtenues sont affichées figure 4.1. Un classifieur parfait atteindrait le coin en haut à gauche, avec la plupart des seuils s'approchant de ce point (zéro faux négatifs et un taux maximal de vrais positifs). Comme on peut observer, la classification est relativement positive par rapport au hasard mais non parfaite. Il

df min	specif min	Rappel	Précision	F-1	Nb classes
1	0,90	**0,94**	0,14	0,25	469
9	0,95	0,73	0,36	0,48	146
9	0,999	0,72	0,39	**0,50**	132
13	0,999	0,55	**0,40**	0,46	98
15	0,999	0,45	0,38	0,41	85

TABLE 4.2 – Tableau de résultats de l'extraction de classes en faisant varier les seuils de fréquence document minimale (df min) et spécificité minimale (specif min). Toutes les expériences ne sont pas affichées. La dernière colonne montre le nombre de classes extraites pour chaque configuration.

y a un taux de faux négatif très faible car très peu de classes sont sélectionnées. La variation du seuil de spécificité minimal n'affecte que très peu le résultat. On peut en conclure qu'il n'y a qu'en faisant varier ces seuils que l'on peut tenter d'améliorer la qualité des classes sélectionnées de façon expérimentale, et donc l'utilisation de la spécificité et du score de fréquence document sont des critères qui sont pertinents mais non « parfaits ».

D'après le tableau 4.2, l'expérience qui a obtenu la meilleure F-mesure (0,50) est celle avec une spécificité à 0,999 et une fréquence document minimale à 9. L'expérience qui a obtenu la meilleure précision (0,40) est celle avec une spécificité à 0,999 et une fréquence document minimale à 13. Enfin celle qui obtient le meilleur rappel (0,94) est celle avec les seuils les moins stricts : spécificité minimale à 0,9 et 0,95 et fréquence document minimale à 1, pour laquelle 469 classes ont été extraites. La configuration la plus stricte (df min = 15 et specif min = 99,9) a obtenu une F-mesure à 0,41 avec 85 classes extraites.

En parallèle de ces résultats, nous observerons le nombre de classes diminuer lorsque l'on fait varier la fréquence document minimale et la spécificité minimale figure 4.2.

Conclusion

En conclusion, il est difficile de définir à quel degré on veut restreindre les concepts qui seront présents dans l'ontologie. Le mieux est de fixer des seuils de façon empirique et tester différentes métriques, en ayant en tête une liste de classes que l'on veut voir apparaître, voire de filtrer manuellement celles qui nous semblent superflues. En effectuant l'extraction automatiquement et en filtrant manuellement, on garantit un bon rapport entre qualité des classes et représentativité. En effet, on peut découvrir des concepts qui sont présents dans le corpus mais que nous n'avions pas envisagés et qui s'avèrent utiles pour représenter le domaine. Ainsi, cette étape de conceptualisation sélective joue un rôle important dans la création automatique d'ontologies synthétiques à partir d'un corpus large. Cependant, elle n'est pas nécessaire dans un contexte où chaque mot est important et doit être conceptualisé, comme pour un corpus technique tel que Solaris.

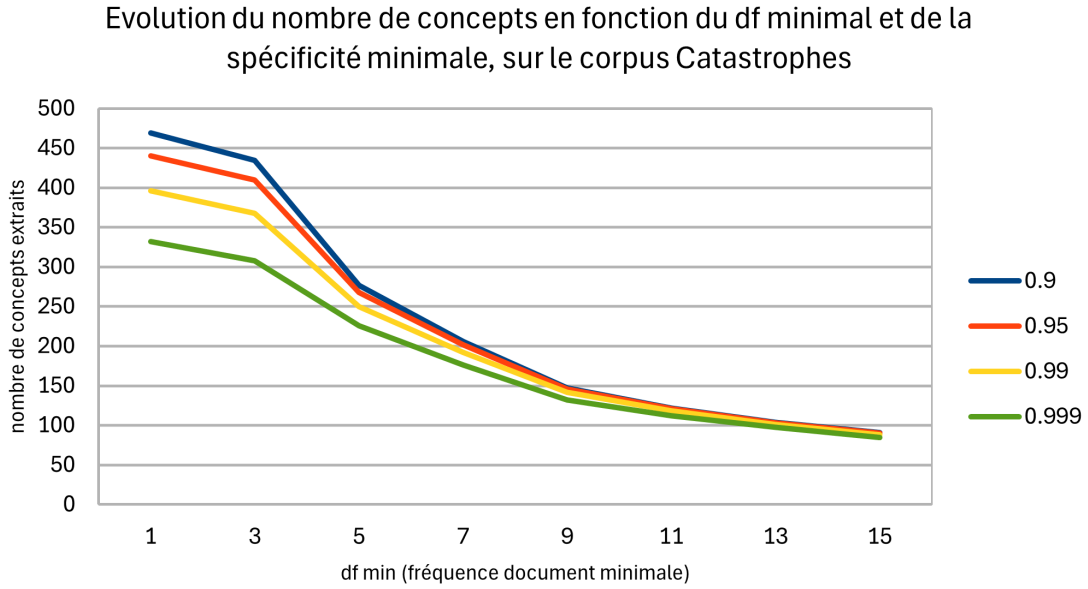

FIGURE 4.2 – Courbes de l'évolution du nombre de classes extraites en faisant varier la spécificité minimale (couleurs) et la fréquence document minimale.

4.2 Évaluation du référencement d'ontologies

4.2.1 Évaluation de la sélection des candidats

Expériences et métriques

Afin d'évaluer le référencement d'ontologies (section 3.4), nous avons annoté les classes de l'ontologie Catastrophes qui a été générée automatiquement à partir du corpus Catastrophes (version court). Un total de 188 classes ont été annotées avec une référence Wikidata au format « Q + identifiant numérique » avec une relation *References*, dans le but de déterminer si le système de référencement d'ontologies est capable de retrouver la bonne classe Wikidata. Pour chaque classe à annoter, des relations *CandidateClass* pour les 10 premières références candidates renvoyées par l'API de Wikidata sont produites.

Pour cette expérience, nous utilisons des métriques qui mesurent la qualité d'un classement étant donné une classe vraie et des classes fausses. Voici les deux métriques utilisées :

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rang_i}$$

Le MRR (*Mean Reciprocal Rank*) calcule l'inverse du rang du bon candidat dans le classement, et fait la moyenne de cette valeur pour les 188 classes. Si le bon élément n'est pas parmi la liste des résultats, son rang est considéré comme infini, alors son inverse est 0. Ce score donne une indication sur le rang du résultat : plus le score est proche de 1, plus le rang moyen est petit, donc meilleur est le classement moyen.

Une autre métrique envisageable est le Hit@k, qui compte combien de fois le bon résultat est renvoyé parmi les k premiers résultats de la requête, sans tenir compte de

sa position exacte dans la liste. Il s'agit d'une évaluation binaire : soit le bon résultat est présent dans les k premiers, soit il ne l'est pas.

$$Hit@k = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 & \text{si } rang_i \leq k,\\ 0 & \text{sinon}\end{cases}$$

Cette mesure a tendance à donner un score plus élevé que d'autres métriques lorsque k est élevé, car elle ne pénalise pas le fait que le bon résultat soit plus bas dans le classement, ce qui permet d'évaluer la fiabilité générale du modèle pour capturer le bon candidat parmi une liste. Le Hit@10, par exemple, détermine à quel pourcentage le bon candidat est parmi le résultat de la requête de l'API Wikidata avec une limite de 10 items renvoyés, et le Hit@1 détermine à quelle fréquence le bon candidat se retrouve dans le top 1.

Résultats

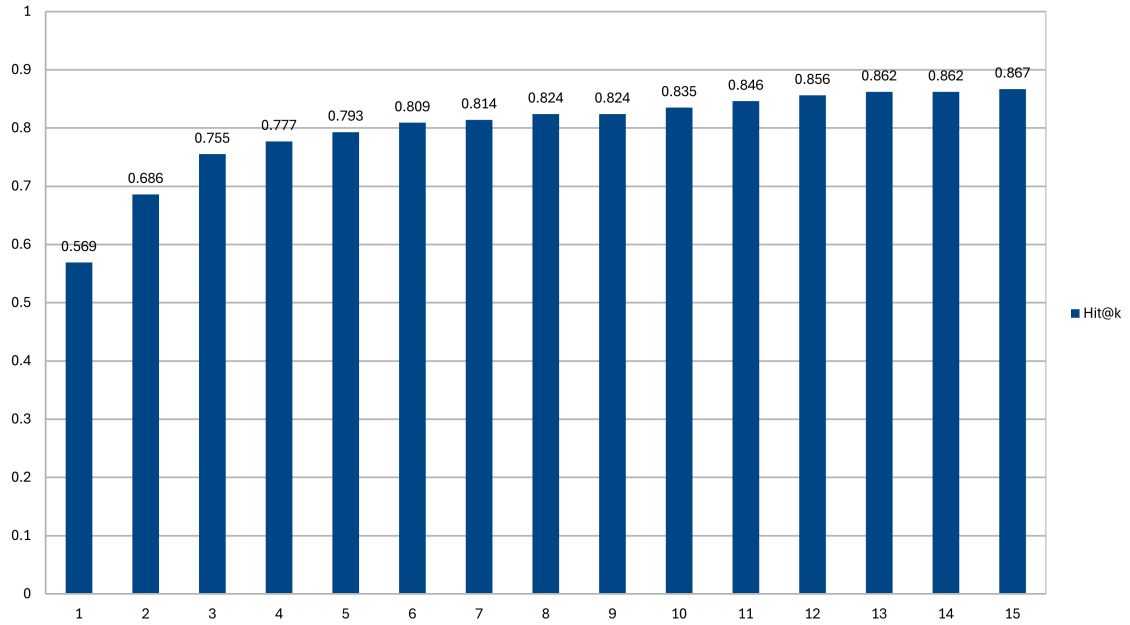

FIGURE 4.3 – Évolution du hit@k sur 188 classes du corpus Catastrophes annotées manuellement, pour k allant de 1 à 15, après avoir récupéré le classement proposé par l'API de Wikidata.

Sur la figure 4.3, on observe que le hit@1 en sortie de l'API Wikidata est déjà supérieur à 50 %, ce qui signifie que dans plus de la moitié des cas, le premier résultat de Wikidata est le bon candidat. Puis, le hit@k augmente de plus en plus lentement. Un plateau est trouvé à partir du hit@13 autour de 0,86. Le score MRR s'élève à 0,665, en adéquation avec la courbe hit@k.

Conclusion

Dans cette partie, nous avons annoté une ontologie et mesuré les scores de MRR et Hit@k obtenus pour le référencement de 188 classes avec l'API de Wikidata. Nous avons obtenu des scores encourageants, mais ces scores élevés reflètent sans doute

le fait que les concepts de l'ontologie Catastrophes ne sont pas rares, et donc ont de fortes probabilités d'être renvoyés par l'API Wikidata.

Garder le moins de candidats possibles peut s'avérer utile pour la désambiguïsation, puisque diminuer le nombre de candidats laissera moins de place à l'erreur. Le dilemme est donc de trouver à partir de combien de candidats s'arrêter. Pour la suite des expériences, nous avons gardé un k maximal à 10 afin de montrer ce qu'il se passe avec la limite du nombre de clusters du modèle de XNLI choisi.

4.2.2 Évaluation de la désambiguïsation

Expériences et métriques

En sortie du modèle XNLI, nous obtenons un re-classement des candidats de l'API Wikidata. Afin de déterminer si le *Random Walk* est efficace ou non avec XNLI, et si non, comment améliorer les performances de la sélection de candidat, nous avons mis en place une série d'expériences en changeant les trois paramètres d'entrée du modèle : 1) le prompt utilisé, 2) le format du contexte[3], 3) les informations dans les clusters pour chaque classe Wikidata candidate en français ou en anglais, parmi lesquelles on peut trouver le label de la classe, la description de la classe, les labels des superclasses, séparés par des points.

Résultats

Ci-dessous, l'explicitation des hyperparamètres utilisés pour les expériences présentées figure 4.4. Seule l'expérience 4 utilise le *Random Walk*, les autres utilisent le contexte dans le corpus avec une fenêtre de 10 tokens à gauche et à droite. Tous les résultats utilisant le *Random Walk* donnaient des scores similaires, alors seul celui-ci est présenté. Dans le prompt, le modèle XNLI remplace les accolades par le nom des clusters et calcule un score de similarité pour renvoyer un classement des candidats.

- Aléatoire : en choisissant aléatoirement une classe candidate.
- API WIkidata : classement renvoyé par Wikidata, sans désambiguïsation.
- Expérience n°1 : clusters en anglais contenant la superclasse et la description. Contexte issu du corpus. Prompt : « Dans ce context, 'label' est défini par {}. »
- Expérience n°2 : clusters en anglais contenant le label, la superclasse et la description. Contexte issu du corpus. Pas de prompt.
- Expérience n°3 : idem, mais avec le prompt : « Dans ce context, 'label' est défini par {}. »
- Expérience n°4 : idem, mais utilise l'algorithme *Random Walk* pour générer un contexte aléatoirement en se déplaçant dans le graphe.
- Expérience n°5 : avec un modèle NLI français (distilcamembert-base-nli). Utilise les labels français de Wikidata : labels, superclasses et description. Contexte issu du corpus. Pas de prompt.

Conclusion

L'expérience qui a les moins bons hit@k et MRR est l'expérience effectuée sur le modèle NLI français. Les autres expériences utilisant le modèle XNLI multilingue ont toutes un hit@1 inférieur à celui du classement renvoyé par Wikidata, mais les courbes ont une tendance similaire.

3. Le contexte est soit extrait directement du corpus autour d'un token avec une fenêtre de 10 tokens avant et après, soit généré à partir du graphe de connaissances et algorithme de *Random Walk*.

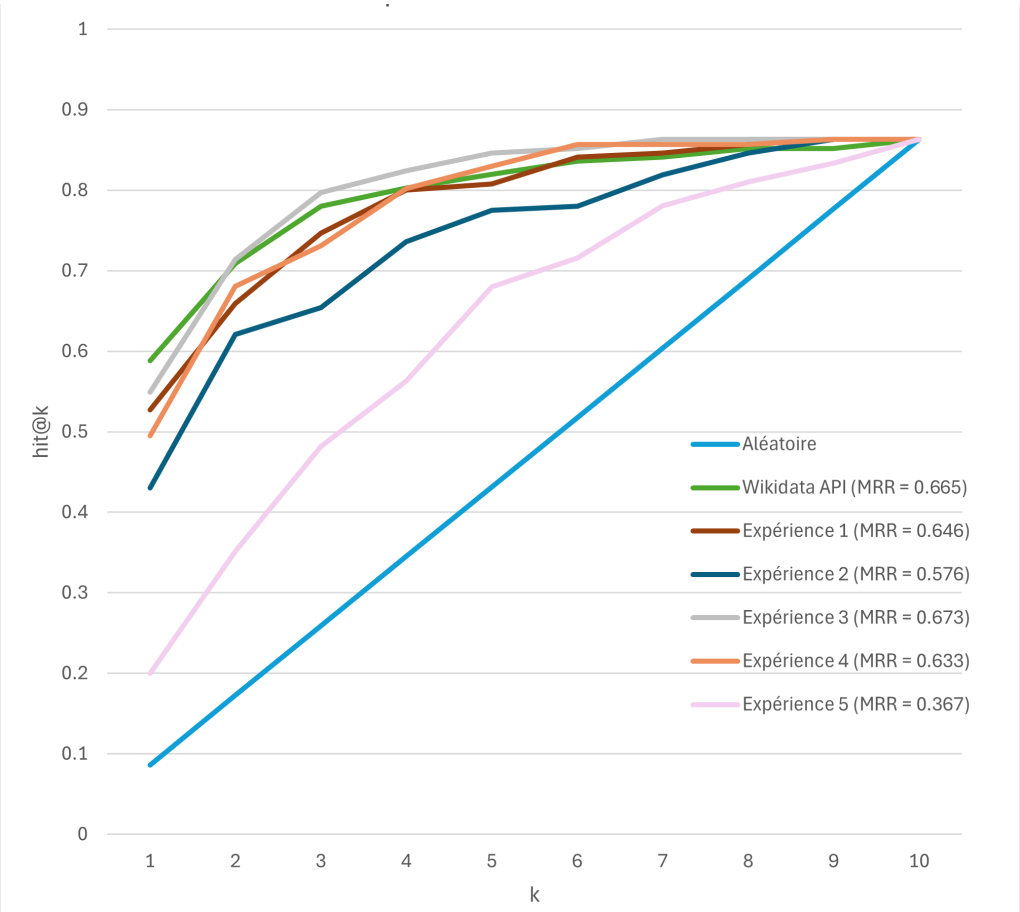

FIGURE 4.4 – Courbes d'évolution du hit@k pour 5 expériences de désambiguïsation par rapport au hasard (courbe bleue) et au hit@k du classement renvoyé par la requête Wikidata (courbe verte).

Pourquoi le MRR obtenu par XNLI n'est pas meilleur que celui de la requête Wikidata ? Le résultat de la requête Wikidata obtient le meilleur hit@1 (0,588) et un MRR supérieur à la quasi-totalité des expériences basées sur NLI : sans avoir effectué de désambiguïsation, une simple requête basée sur des statistiques et le label français permet de trouver la meilleure classe la plupart du temps. Ceci s'explique par le fait que les classes que nous recherchons ne sont pas rares, le sujet du corpus Catastrophes touchant à des notions assez répandues. La deuxième explication concerne l'état de l'art : en effet, en comparant les résultats de diverses études, on observe que les systèmes de désambiguïsation obtiennent des scores très variables en fonction de l'ontologie et de la technologie utilisée ([Chen et al., 2021]). Ainsi, réussir à dépasser 0,5 au hit@1 est déjà une preuve que l'utilisation d'un modèle de NLI peut résoudre la tâche de désambiguïsation.

Qu'en est-il du modèle en français ? Le fait que le modèle NLI français (expérience n°5) obtienne de si mauvais résultats est peut-être dû au manque d'informations pour le français dans Wikidata, diminuant le taux de présence du bon candidat dans les clusters, ainsi qu'aux performances générales du modèle NLI choisi sur cette tâche en particulier. Un modèle NLI plus adapté à cette tâche pourrait être fine-tuné à l'avenir.

Quels critères affectent la qualité ? Le fait d'utiliser un algorithme de *Random*

Walk par rapport à utiliser le contexte du corpus n'affecte que très peu la qualité de l'analyse. Il n'y a pas de vraie explication à cela, mais nous supposons que le modèle XNLI n'arrive pas à véritablement capter le sens des phrases dans leur ensemble, affectant la qualité de façon similaire des deux côtés, peu importe si la phrase est grammaticalement correcte ou non. De plus, le contenu des clusters (descriptions, labels, superclasses) est différent du contenu de l'entité à relier (contexte) car ils se rapprochent davantage d'une définition du concept que d'une utilisation de celui-ci. Dans l'état actuel, nous avons la possibilité d'utiliser un graphe de connaissances qui permet de faire du *Random Walk* sur le contexte des instances ou de simplement utiliser la phrase d'origine, mais utiliser d'autres informations telles que la classe parente de la classe recherchée serait potentiellement efficace, si la taxonomie était plus aboutie. Si on prend l'exemple de la tornade, si on sait déjà que « tornade » est une sous-classe de « catastrophe naturelle » ou « catastrophe météorologique » dans notre ontologie, il sera plus simple de la relier avec le bon concept Wikidata (Q8081), qui a pour classe parente « natural disaster » (Q8065) dans Wikidata.

4.3 Évaluation de la qualité de l'ontologie

Précédemment, nous avons évalué trois étapes intermédiaires de l'ontologie : l'extraction des triplets, la conceptualisation et le liage d'entités. La qualité de l'ontologie résultante dépend évidemment de ces trois évaluations, mais nous allons tenter d'évaluer l'ontologie en sortie, en nous intéressant particulièrement à la taxonomie, qui n'a pas été évaluée.

Pour rappel, afin d'évaluer la qualité d'une ontologie, il est possible de se baser sur des critères d'évaluation, une ontologie « gold standard », ses performances sur une tâche donnée ou en se référant au corpus d'origine [Raad and Cruz, 2015]. Dans cette partie, nous utilisons des techniques de type « gold standard » afin d'évaluer la qualité de la taxonomie.

4.3.1 Évaluation de la taxonomie

Expériences et métriques

Dans cette expérience, nous allons évaluer la qualité de la taxonomie de façon arbitraire en calculant la distance d'édition (nombre de clics) entre la taxonomie obtenue et une taxonomie idéale pour les classes héritant de la classe « Catastrophe ».

Avec le prompt (A.3), notre système a généré la taxonomie figure A.5. En revanche, la taxonomie obtenue après simple lecture du corpus est bien différente (figure 4.5).

Résultats

L'avantage de la taxonomie obtenue par extraction depuis le corpus est qu'elle apporte un niveau de découpage supplémentaire aux catastrophes (météorologique, sanitaire, Seveso), mais l'inconvénient est que certaines catastrophes spécifiques, avec un lieu ou une date, ont été conceptualisées, en raison d'un nombre élevé d'instances de ces catastrophes dans le corpus. Un petit défaut à corriger est de glisser « Catastrophe Sanitaire Mondial » sous « Catastrophe Sanitaire ». On observe aussi que «

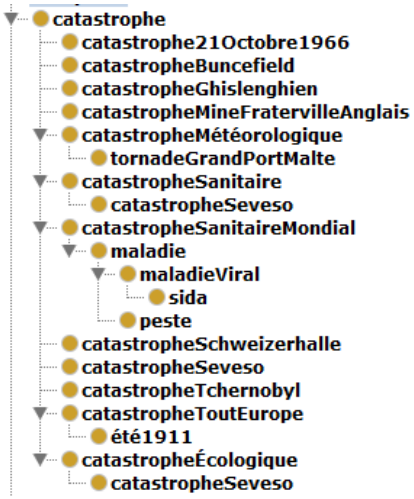

FIGURE 4.5 – Capture d'écran dans Protégé de la taxonomie des catastrophes extraites du corpus, sans prompt.

	Post-édition Catastrophes	Création complète Catastrophes	Corpus technique (Solaris)
Taille du corpus	~50k tokens	~50k tokens	171 énoncés
Temps d'édition dans Protégé	~25 min	~25 min	~2 jours
Temps de lecture du corpus	0h	~5 heures	~1 heure
Nombre de clics dans Protégé	150 clics	150 clics	-
Nombre de classes créées	0	50+nommage	~50+nommage
Nombre de classes supprimées	15	0	0
Nombre de classes déplacées	15	0	0
Nombre de superclasses ajoutées à une classe existante	8	-	-

TABLE 4.3 – Tableau récapitulatif du temps et des efforts nécessaires à la post-édition de la taxonomie des Catastrophes (colonne 2) par rapport à sa création manuelle complète (colonne 3) et la création d'une taxonomie pour un corpus technique tel que Solaris (colonne 4). La plupart des chiffres sont estimés, non mesurés avec précision. Le nommage correspond au fait de devoir entrer au clavier le nom de la classe.

Catastrophe Seveso » a deux classes parentes : « Catastrophe Écologique », et « Catastrophe », qui est la tête du groupe nominal. Le dernier défaut est que les types de catastrophes (« Tornade », « Explosion », etc) n'ont pas été placés sous « Catastrophe » ni « Catastrophe naturelle », mais sous « Entity »[4], car le corpus n'a pas défini une « tornade ». Afin de régler ce problème, nous avons fusionné les taxonomies issues du prompt descriptif et du corpus, voir annexe A.6. Nous avons édité cette taxonomie des catastrophes pour l'améliorer au mieux avec uniquement des classes déjà présentes dans la taxonomie entière.

La taxonomie obtenue après édition est présentée figure 4.6. Parmi les modifica-

4. Entity est la classe parente de toutes les entités dans notre ontologie, s'opposant à la classe Action pour les verbes, et à la classe Value pour les adjectifs et les adverbes. Toutes les instances de concepts héritant de ces classes sont des *DocumentNode*.

tions apportées, on compte la suppression des classes-instances, qui contiennent un lieu ou une date ; ou encore le déplacement des catastrophes de type météorologique telles que « Tempête » ou « Canicule » sous « Catastrophe Météorologique ». Parmi les classes déplacées, il y en a qui n'héritaient pas de « Catastrophe », comme « éboulement », qui héritait d' « Entity ». Le tableau 4.3 met en contraste le temps et les efforts nécessaires à la post-édition de la taxonomie des catastrophes par rapport à la création complète de cette taxonomie et la création d'une taxonomie pour un corpus technique, d'après les données d'Airudit.

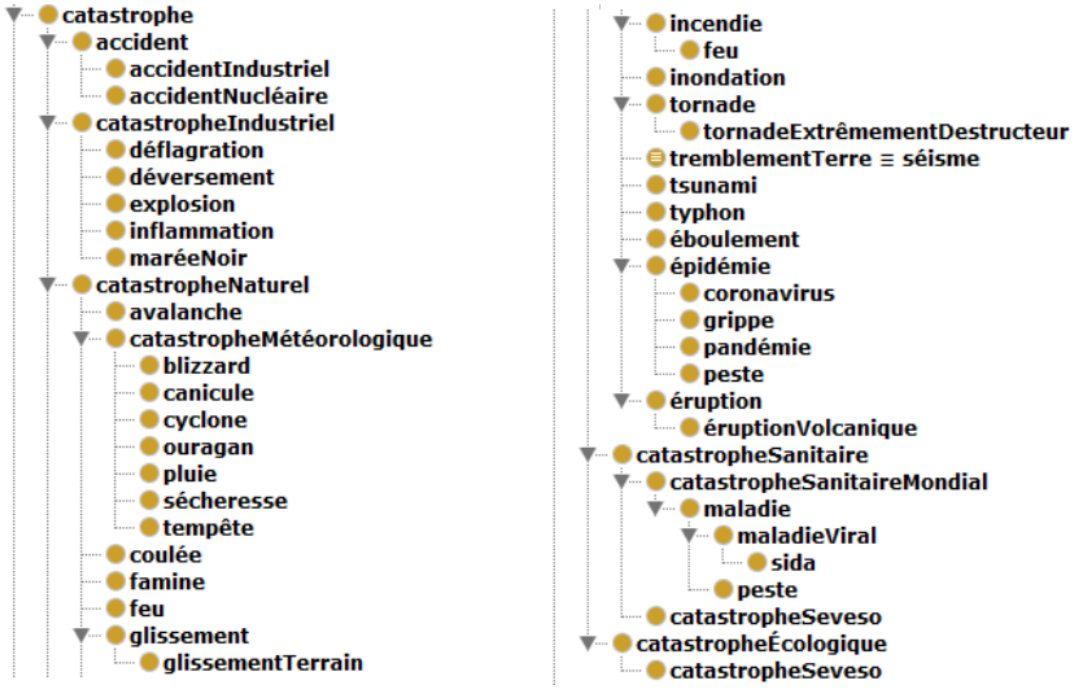

FIGURE 4.6 – Capture d'écran dans Protégé de la taxonomie des catastrophes après fusion de la taxonomie prompt et de la taxonomie issue du corpus et après une édition manuelle de la taxonomie résultante.

Conclusion

En moyenne, 6 clics par minute pendant 25 minutes ont été nécessaires pour passer de la taxonomie de sortie de l'algorithme à la taxonomie des catastrophes idéale. Mais la taxonomie des catastrophes n'est qu'une petite partie de la taxonomie entière, et pour post-éditer toute la taxonomie, quelques efforts supplémentaires de l'ordre de quatre fois le travail fourni doit être ajouté, par exemple : l'ajout d'équivalence entre les concepts « événement » et « évènement », car la taxonomie des catastrophes représente environ 1/4 du nombre de classes total.

Enfin, nous remarquons que grâce à la dualité de l'origine de la taxonomie, nous diminuons la charge travail de l'ingénieur des connaissances, puisqu'il n'est plus que de l'ordre de déplacements de classes et suppressions de classes dans un outil d'édition d'ontologies.

CHAPITRE 5

DISCUSSION

Sommaire

5.1 Introduction

Dans ce chapitre, nous allons d'abord replacer ce travail dans le contexte de la recherche en apprentissage d'ontologies. Puis, nous allons discuter d'éventuels biais et problèmes rencontrés dans les expériences réalisées. Ensuite, nous présenterons des perspectives d'amélioration pour de prochains travaux, avant de conclure.

5.2 Contenu

5.2.1 Comparaison avec les travaux précédents

Notre travail d'extraction de triplets se rapproche de RelExt [Schutz and Buitelaar, 2005], qui place le prédicat au cœur de l'extraction d'informations, mais nous obtenons des scores nettements supérieurs que ceux annoncés, sans doute grâce aux progrès des outils d'annotation en dépendances syntaxiques depuis 2005 et au fait que la chaîne de traitements ait été conçue à partir d'UD mais aussi de la sortie de SpaCy, pouvant potentiellement constituer un biais, tout comme le fait que le corpus utilisés dans RelExt, composé de commentaires en direct d'actions de matchs de football, ne possède pas le même niveau de difficulté que le corpus Solaris sur lequel nous avons fait nos tests.

Quelle est la particularité de nos triplets ? [Prakash et al., 2021] dresse un tableau comparatif du format des triplets issus de quatre systèmes d'OIE, parmi lesquels BRMiner, StanfordOIE, Supervised OIE et ClausIE. En sortie de norte système d'OIE, les adpositions sont détachées ; les triplets sont décomposés pour chaque prédicat, relation logique et entité membre d'une conjonction ; et chaque entité est décomposée en interne sur ses modifieurs, poussant la logique de décomposition de BRMiner à l'extrême. Par exemple, le triplet en sortie de BRMiner, <scheduled booking, of, assigned car>, deviendrait, si on adaptait notre outil à l'anglais : <scheduled

booking, *nmod*, assigned car> + <scheduled booking, *hasHead*, booking> + <booking, *amod*, scheduled> + <assigned car, *hasHead*, car> + <car, *amod*, assigned>. L'intérêt est de pouvoir répondre à des questions comme « How is the booking ? » ou « How is the car ? » en s'appuyant sur les modifieurs, qui jouent le rôle du verbe être : <car, is, assigned>.

Comment améliorer le référencement d'ontologies ? Le référencement d'ontologies par *Random Walk* est la technique la plus utilisée pour relier une ontologie de domaine à une ontologie généraliste. Mais notre méthode utilise des mots-clés pour pré-sélectionner une dizaine d'entités afin de ne pas prendre en compte toutes les entités de Wikidata dans le calcul d'embeddings. D'autres approches utilisent un embedding pré-calculé pour chaque entité, comme RDF2Vec [Christensen et al., 2022] sur DBpedia, ce qui permet de ne pas avoir à générer à nouveau ces embeddings et de calculer la distance entre les entités à relier plus rapidement.

5.2.2 Problèmes rencontrés

Chaînes de coréférence. La génération des chaînes de coréférence a posé quelques problèmes lors de la réalisation de ces travaux. Le premier problème est le manque de fiabilité des modèles de résolution des coréférences, le second est le temps d'exécution du modèle Camembert utilisé. En effet, sur un corpus comme le nôtre, cette étape représente plus de 98 % du temps d'exécution (10 minutes sans contre 20 heures avec). C'est pour cette raison que les tests ont été effectués sur la **version courte du corpus Catastrophes**, des tests supplémentaires sur la version complète du corpus donneraient des résultats différents de ceux présentés dans ce mémoire.

Liage des relations. Les relations du graphe de connaissances n'ont pas été « conceptualisées » à la manière des entités, ainsi les relations extraites ne sont pas d'assez haut niveau et trop dépendantes du corpus. La structure du graphe de connaissances reste donc à formaliser, notamment avec le liage de relations (ou *Relation Linking*).

5.2.3 Perspectives

Non-contradiction et découverte d'axiomes. Une des étapes manquantes à notre méthode est de vérifier la consistence des informations avant de les fusionner entre elles. L'autre étape manquante est la découverte d'axiomes, ou la généralisation des règles qui affectent le fonctionnement de l'ontologie. Ce processus s'ancre dans une méthodologie de type « bottom-up » : partir des instances pour en dégager des principes fondamentaux. Ces étapes sont ouvertes à la réflexion dans de futurs travaux.

Clusterisation. Parmi les méthodes d'extraction d'informations non supervisées, il existe le clustering, par exemple les graphes de contrastes proposés par le LORIA [Cuxac and Lamirel, 2013], qui permettraient d'ébaucher une ontologie en proposant des clusters, ou thèmes, qui sont connectés entre eux via des termes. Avec Jean-Charles Lamirel, nous avons testé l'extraction d'un graphe de contraste sur la version complète du corpus catastrophes, et les grands thèmes qui s'en dégagent sont effectivement les principaux types de catastrophes (incendie, tempête...). Un exemple d'usage pourrait être de récupérer ces thèmes et d'en faire des concepts dans l'ontologie, puis de regarder quels termes sont associés à la plupart d'entre ces thèmes et d'en faire des attributs ou d'autres concepts. Cette vision est très opposée à la nôtre,

puisqu'il s'agit d'une approche « sac de mots », alors que nous avons mis l'emphase sur les arbres de dépendances syntaxiques et l'extraction des connaissances.

Compatibilité Les moteurs de Prométh.ai et Valer.ia sont basés sur BFO (*Basic Formal Ontology*), qui procure à la machine des capacités de raisonnement logique. Afin d'intégrer le résultat de ces recherches aux projets de l'entreprise, un travail manuel visant à relier les classes de la taxonomie aux classes de BFO est nécessaire. Nous pensons qu'automatiser cette tâche est une perspective intéressante. Pour l'instant, tous les concepts « nominaux » sont placés sous une classe temporaire « Entity », et les verbes sont places sous une classe temporaire « Action », mais dans BFO, il existe des classes qui permettent de spécifier davantage la fonction de chaque concept dans l'espace et dans le temps. Par exemple, dans BFO, « Realizable Entity » est utilisé pour les entités qui peuvent aussi être une action, comme nos verbes réifiés.

Détection d'intentions pour KESAIO Décrire les modifications à appliquer à l'ontologie en langue naturelle est une piste à explorer à l'avenir. Dans la partie 3.3, nous avons interprété chaque affirmation d'un corpus descriptif en langue contrôlée [1] comme étant une description d'une nouvelle relation hiérarchique : « X est une sous-classe de Y ». Un système plus complet serait capable de détecter l'intention de l'opération désirée parmi les opérations CRUD [2] existant dans un outil d'édition d'ontologies tel que Protégé, par exemple : consulter une entité, supprimer une classe, ajouter une instance à une classe, ajouter une relation entre deux classes, etc. Réussir à généraliser notre méthode de découverte de taxonomie à la langue naturelle serait une grande avancée, de même que de la généraliser à toutes les opérations possibles sur une ontologie grâce à la détection d'intentions. Cela pourrait permettre de réduire encore la charge cognitive quant au maintien de l'ontologie et de rendre les ontologies plus accessibles aux experts métiers.

5.3 Conclusion

Dans ces discussions, nous avons commencé par comparer notre travail aux travaux précédents, et avons remarqué des divergeances significatives avec ces derniers. Puis, nous avons synthétisé les difficultés rencontrées, que ce soit au niveau méthodologique, au niveau des outils utilisés ou au niveau des performances. De plus, nous avons discuté de pistes d'amélioration et de futures recherches qui viendraient combler les lacunes de ce mémoire, notamment l'emploi de la clusterisation, l'exploitation des classes de BFO et la détection d'intentions pour la constitution et la mise à jour automatique d'ontologies.

1. La langue contrôlée est une langue naturelle qui utilise une grammaire simplifiée.
2. *Create, Remove, Update, Delete.*

Conclusion générale

Dans ce mémoire, nous avons abordé la construction automatique d'ontologies à partir de corpus de spécialité, dont l'objectif final est d'aider l'ingénieur des connaissances dans sa tâche de conception d'ontologies. Afin de répondre à cette problématique, nous avons mis en place un processus capable d'extraire un graphe de connaissances et une taxonomie des concepts représentatifs du domaine. Notre système d'apprentissage d'ontologies s'appuie sur trois étapes principales : l'extraction d'informations en contexte ouvert, où des triplets sont extraits via l'arbre syntaxique des phrases du corpus ; la conceptualisation, permettant de regrouper des entités sous un même concept selon leur importance relative ; et la hiérarchisation, basée sur des motifs syntaxiques pour extraire des relations d'hyperonymie/hyponymie. Bien que cette méthode assiste efficacement l'ingénieur des connaissances dans tout le processus de création, elle ne remplace pas son travail. En outre, la découverte d'axiomes à partir des instances et la vérification de la non-contradiction des informations restent à explorer.

Les expériences réalisées sont globalement concluantes dans l'objectif d'automatiser en partie la construction de l'ontologie à partir de corpus de spécialité. Cependant, même si les méthodologies utilisées ont permis de répondre aux exigeances des différentes tâches d'extraction, chaque étape nécessite toujours une révision manuelle en raison de la complexité des tâches et du fait que l'apprentissage d'ontologies est un domaine encore en développement [Du et al., 2024]. Par exemple, l'emploi de SpaCy sur le corpus Solaris a permis d'atteindre une F-mesure de 0,812 sur la tâche d'extraction de triplets, mais des difficultés subsistent sur les phrases longues et complexes. Pour la taxonomie, nous avons préféré laisser le contrôle à l'utilisateur final, en lui proposant de décrire lui-même les concepts du domaine en langue contrôlée dans un prompt descriptif, plutôt que de faire appel à un modèle génératif. Nous avons observé que la combinaison des classes extraites du corpus avec les concepts définis dans le prompt descriptif offre un compromis entre qualité de la taxonomie et couverture de domaine. Nous avons montré que le post-traitement nécessaire sur cette taxonomie est rapide : moins de 2 heures pour un corpus de 50 000 tokens, contre deux jours sur un corpus d'une centaine de phrases pour une création manuelle complète.

Ces travaux soulèvent néanmoins des questions quant à l'usage pratique de l'ontologie en contexte réel. Par exemple, chez Airudit, les ontologies héritent de BFO (*Basic Formal Ontology*), ce qui implique une étape supplémentaire de liaison entre les entités et les relations extraites avec les concepts de BFO. Cette étape peut être résolue manuellement, ou nous pouvons envisager de fine-tuner un modèle pour effectuer cette connexion. Par ailleurs, une approche émergente, StructRAG [Li et al., 2024], propose une perspective nouvelle où les ontologies sont structurées en fonction des tâches qu'elles doivent servir, et non des concepts issus d'un corpus en langue naturelle, remettant en question la structure inspirée de la langue naturelle et des ontologies conventionnelles qui est développée dans ce mémoire. Explorer cette piste pourrait constituer une avancée prometteuse. Enfin, l'exploitation des ressources ex-

ternes permettrait d'enrichir l'ontologie et d'accélérer certaines phases du processus, mais cette piste n'a finalement pas été approfondie dans ce mémoire. Ces théories ouvrent la voie à de nouvelles recherches pour améliorer l'efficacité de la génération automatique d'ontologies à partir de corpus de spécialité dans le cadre d'une application industrielle.

BIBLIOGRAPHIE

[Asim et al., 2018] Asim, M. N., Wasim, M., Khan, M. U. G., Mahmood, W., and Abbasi, H. M. (2018). A survey of ontology learning techniques and applications. *Database (Oxford)*, 2018. – Cité page 19.

[Borgo et al., 2022] Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E. M., and Vieu, L. (2022). Dolce: A descriptive ontology for linguistic and cognitive engineering1. *Applied Ontology*, 17(1):45–69. – Cité page 9.

[Chen et al., 2021] Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O. M., Antonyrajah, D., and Horrocks, I. (2021). Owl2vec*: Embedding of owl ontologies. – Cité pages 17 et 51.

[Chen et al., 2024] Chen, Z., Liu, J., Yang, D., Xiao, Y., Xu, H., Wang, Z., Xie, R., and Xian, Y. (2024). Exploiting duality in open information extraction with predicate prompt. – Cité page 16.

[Christensen et al., 2022] Christensen, M. P., Lissandrini, M., and Hose, K. (2022). Dbpedia rdf2vec graph embeddings. – Cité page 56.

[Cuxac and Lamirel, 2013] Cuxac, P. and Lamirel, J.-C. (2013). Analyse des évolutions et des interactions entre domaines scientifiques: Grafsel, association de la sélection de variables et de la représentation graphique. – Cité page 56.

[Du et al., 2024] Du, R., An, H., Wang, K., and Liu, W. (2024). A short review for ontology learning: Stride to large language models trend. – Cité pages 13 et 59.

[Fernández-López et al., 1997] Fernández-López, M., Gomez-Perez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. *Engineering Workshop on Ontological Engineering (AAAI97)*. – Cité pages 20, 22 et 26.

[Fisher, 1987] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172. – Cité pages 17 et 36.

[Fundel et al., 2006] Fundel, K., Küffner, R., and Zimmer, R. (2006). RelEx—Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371. – Cité page 14.

[Gan et al., 2024] Gan, Y., Poesio, M., and Yu, J. (2024). Assessing the capabilities of large language models in coreference: An evaluation. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N., editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1645–1665, Torino, Italia. ELRA and ICCL. – Cité page 38.

[Gao et al., 2024] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. – Cité page 10.

[Gerdes et al., 2018] Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2018). SUD or surface-syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In de Marneffe, M.-C., Lynn, T., and Schuster, S., editors, *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium. Association for Computational Linguistics. – Cité page 30.

[Giglou et al., 2023] Giglou, H. B., D'Souza, J., and Auer, S. (2023). Llms4ol: Large language models for ontology learning. – Cité page 14.

[Grobol, 2020] Grobol, L. (2020). *Coreference resolution for spoken French*. Theses, Université Sorbonne Nouvelle - Paris 3. – Cité page 38.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220. – Cité page 9.

[Guarino, 1998] Guarino, N. (1998). *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy*. Frontiers in artificial intelligence and applications. IOS Press. – Cité page 26.

[He et al., 2022] He, Y., Chen, J., Antonyrajah, D., and Horrocks, I. (2022). Bertmap: A bert-based ontology alignment system. – Cité page 17.

[He et al., 2024] He, Y., Yuan, Z., Chen, J., and Horrocks, I. (2024). Language models as hierarchy encoders. – Cité page 18.

[Heinecke, 2020] Heinecke, J. (2020). Hybrid enhanced Universal Dependencies parsing. In Bouma, G., Matsumoto, Y., Oepen, S., Sagae, K., Seddah, D., Sun, W., Søgaard, A., Tsarfaty, R., and Zeman, D., editors, *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 174–180, Online. Association for Computational Linguistics. – Cité pages 30 et 31.

[Hogan et al., 2021] Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37. – Cité page 28.

[Honnibal and Montani, 2017] Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear. – Cité pages 7 et 28.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From shiq and rdf to owl: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26. – Cité page 26.

[Huguet Cabot and Navigli, 2021] Huguet Cabot, P.-L. and Navigli, R. (2021). REBEL: Relation extraction by end-to-end language generation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics. – Cité page 14.

[Jiang et al., 2024] Jiang, X., Shen, Y., Shi, Z., Xu, C., Li, W., Li, Z., Guo, J., Shen, H., and Wang, Y. (2024). Unlocking the power of large language models for entity alignment. – Cité page 17.

[Joshi et al., 2019] Joshi, M., Levy, O., Weld, D. S., and Zettlemoyer, L. (2019). Bert for coreference resolution: Baselines and analysis. – Cité page 38.

[Kamp et al., 2023] Kamp, S., Fayazi, M., Benameur-El, Z., Yu, S., and Dreslinski, R. (2023). Open information extraction: A review of baseline techniques, approaches, and applications. – Cité page 15.

[Kandpal et al., 2023] Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. (2023). Large language models struggle to learn long-tail knowledge. – Cité pages 7 et 10.

[Kondylakis et al., 2021] Kondylakis, H., Nikolaos, A., Dimitra, P., Anastasios, K., Emmanouel, K., Kyriakos, K., Iraklis, S., Stylianos, K., and Papadakis, N. (2021). Delta: A modular ontology evaluation system. *Information*, 12(8). – Cité page 20.

[Lantow, 2016] Lantow, B. (2016). Ontometrics: Putting metrics into use for ontology evaluation. In *International Conference on Knowledge Engineering and Ontology Development*. – Cité page 20.

[Li et al., 2024] Li, Z., Chen, X., Yu, H., Lin, H., Lu, Y., Tang, Q., Huang, F., Han, X., Sun, L., and Li, Y. (2024). Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. – Cité page 59.

[Maclellan et al., 2016] Maclellan, C., Harpstead, E., Aleven, V., and Koedinger, K. (2016). Trestle: A model of concept formation in structured domains. – Cité page 18.

[New et al., 2004] New, B., Pallier, C., Brysbaert, M., and Ferrand, L. (2004). Lexique 2: A new french lexical database. *Behavior Research Methods, Instruments, & Computers*, 36(3):516–524. – Cité page 37.

[Nguyen et al., 2014] Nguyen, V., Bodenreider, O., and Sheth, A. (2014). Don't like rdf reification? making statements about statements using singleton property. volume 2014, pages 759–770. – Cité page 32.

[Otte et al., 2022] Otte, J. N., Beverley, J., and Ruttenberg, A. (2022). Bfo: Basic formal ontology. *Appl. Ontology*, 17:17–43. – Cité page 9.

[Peng et al., 2024] Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., and Tang, S. (2024). Graph retrieval-augmented generation: A survey. – Cité page 10.

[Prakash et al., 2021] Prakash, C., Chittimalli, P., and Naik, R. (2021). Open information extraction using dependency parser for business rule mining in sbvr format. – Cité pages 15, 35 et 55.

[Qi et al., 2020a] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020a). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. – Cité page 28.

[Qi et al., 2020b] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020b). Stanza: A python natural language processing toolkit for many human languages. – Cité page 43.

[Raad and Cruz, 2015] Raad, J. and Cruz, C. (2015). A Survey on Ontology Evaluation Methods. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference*

on Knowledge Discovery, Knowledge Engineering and Knowledge Management , Lisbonne, Portugal. – Cité pages 18, 20 et 52.

[Schuster and Manning, 2016] Schuster, S. and Manning, C. D. (2016). Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA). – Cité page 30.

[Schutz and Buitelaar, 2005] Schutz, A. and Buitelaar, P. (2005). Relext: A tool for relation extraction from text in ontology extension. In Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., editors, *The Semantic Web – ISWC 2005*, pages 593–606, Berlin, Heidelberg. Springer Berlin Heidelberg. – Cité pages 15, 20, 32, 45 et 55.

[Spyns and Reinberger, 2005] Spyns, P. and Reinberger, M.-L. (2005). Lexically evaluating ontology triples generated automatically from texts. In Gómez-Pérez, A. and Euzenat, J., editors, *The Semantic Web: Research and Applications*, pages 563–577, Berlin, Heidelberg. Springer Berlin Heidelberg. – Cité page 19.

[Studer et al., 1998] Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: principles and methods. data knowl eng 25(1-2):161-197. *Data & Knowledge Engineering*, 25:161–197. – Cité page 26.

[Sukthanker et al., 2018] Sukthanker, R., Poria, S., Cambria, E., and Thirunavukarasu, R. (2018). Anaphora and coreference resolution: A review. – Cité page 38.

[Sun et al., 2017] Sun, Z., Hu, W., and Li, C. (2017). Cross-lingual entity alignment via joint attribute-preserving embedding. – Cité page 17.

[Wächter et al., 2011] Wächter, T., Fabian, G., and Schroeder, M. (2011). Dog4dag: semi-automated ontology generation in obo-edit and protégé. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, page 119–120, New York, NY, USA. Association for Computing Machinery. – Cité pages 13, 27 et 34.

[Zaitoun et al., 2023] Zaitoun, A., Sagi, T., and Hose, K. (2023). Automated ontology evaluation: Evaluating coverage and correctness using a domain corpus. *Companion Proceedings of the ACM Web Conference 2023*. – Cité page 19.

[Zavitsanos et al., 2011] Zavitsanos, E., Paliouras, G., and Vouros, G. (2011). Gold standard evaluation of ontology learning methods through ontology transformation and alignment. *IEEE Trans. Knowl. Data Eng.*, 23:1635–1648. – Cité page 19.

[Zhang et al., 2023] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., Shi, F., and Shi, S. (2023). Siren's song in the ai ocean: A survey on hallucination in large language models. – Cité pages 7 et 10.

[Zhu et al., 2024] Zhu, Y., Moniz, J. R. A., Bhargava, S., Lu, J., Piraviperumal, D., Li, S., Zhang, Y., Yu, H., and Tseng, B.-H. (2024). Can large language models understand context? – Cité page 38.

ANNEXE A

FICHIERS DU PROJET

FIGURE A.1 – Échantillon de noms de catastrophes industrielles utilisées dans le corpus Catastrophes

— Accident ferroviaire de Lac-Mégantic
— Catastrophe de Port Chicago
— Catastrophe minière de Coalbrook
— Explosion de la poudrière de Delft
— Inondation de Johnstown de 1889

FIGURE A.2 – Échantillon de noms de catastrophes naturelles utilisées dans le corpus Catastrophes

— Blizzard de 1972 en Iran
— Famine en Inde de 1876 à 1878
— Inondations de 1931 en Chine
— Peste noire
— Séisme de 2010 en Haïti

Un feu, un incendie, une canicule, un typhon, un glissement, un cyclone, une éruption, un ouragan, un tsunami, une tornade, un tremblement de terre (ou un séisme), une tempête, une sécheresse, une inondation, la pluie, une famine, une coulée, un blizzard, une avalanche et une épidémie sont des catastrophes naturelles. Les marées noires, les explosions, les déflagrations, les déversements sont des catastrophes industrielles. Un cataclysme et un accident sont des catastrophes. Un feu, un incendie ou une famine peuvent être causés par l'Homme. Un feu est un incendie. La peste, la grippe, une pandémie et le coronavirus sont des épidémies. La pollution est une conséquence de catastrophe industrielle. L'effondrement et les effets sont des conséquences de catastrophe. Les morts et les blessés sont des victimes d'une catastrophe. Une catastrophe a un bilan (bilan matériel et victimes). Les victimes sont des dommages humains, le bilan matériel est un dommage matériel ou d'infrastructure. Un séisme a une magnitude (ou ampleur).

FIGURE A.3 – Exemple de description textuelle en LNC pour le corpus Catastrophes

TABLE A.1 – Tableau des relations UD et leur usage ontologique

acl	**clausal modifier of noun**. En fonction des situations, *acl* a un rôle différent pour l'ontologie. D'un point de vue linguistique, il s'agit d'une relation allant d'un nom vers un verbe ou gérondif, comme *acl:relcl*. Si le verbe n'a pas de dépendant (sauf un adverbe), alors il peut être considéré comme le modifieur du nom avec *hasModifier*.
acl:relcl	**relative clause modifier**. Relation entre la tête de la proposition relative (dépendant) et le mot qu'elle modifie (tête). Pour savoir quel est le rôle de la relation entre la proposition et sa tête, il faut regarder la relation entre la proposition et son pronom relatif, c'est la relation qui doit être utilisée à la place de *acl:relcl* [1]. Un exemple de modification de l'arbre syntaxique est présenté diagramme 3.4.
advcl	**adverbial clause**. La relation *advcl* fait un lien logique entre deux clauses et est souvent accompagnée d'un marqueur logique *mark*. Il peut s'agir d'un lien de causalité, de temporalité (avec le marqueur "quand"), de condition (si A alors B)... Il en existe beaucoup, alors nous avons choisi d'utiliser directement le marqueur logique comme relation entre les deux verbes réifiés. Si aucun marqueur n'est présent, nous utilisons une relation *NoMark* dans l'ontologie à la place de *advcl*.
advmod	**adverbial modifier**. La plupart du temps, il s'agit du modifieur d'un adjectif ou d'un verbe, lui donnant une valeur. Dans l'ontologie, nous avons créé une relation *hasAdverb* qui relie sa tête à l'adverbe. Cependant, il existe un cas où l'adverbe est lui-même porteur d'un objet (*ccomp*, *obl:arg*...) [2]. Dans ce cas, on obtient la structure : <Predicat, *hasAdverb*, Adverbe>, suivi de <Adverbe, *hasObject*, Objet> où l'objet est bien le dépendant de l'adverbe dans l'arbre de dépendances.
amod	**adjectival modifier**. Modifie un nom par un adjectif. Nous avons gardé les relation de modifieurs UD dans l'ontologie car ils permettent de répondre à des questions telles que "de quelle couleur est X?". Les modifieurs en UD héritent d'une relation *hasModifier*. *amod* a un rôle particulier comparé aux autres modifieurs, puisqu'un adjectif peut lui-même porter un objet. Dans ce cas, le nom qu'il modifie devient son sujet, comme dans : "objets extérieurs à la Terre", où "extérieurs" est relié par *amod* à "objets", mais le triplet extrait sera <objets, extérieurs [à], la Terre>.
appos	**appositional modifier**. Sert à donner un nom alternatif à une entité, comme dans "Sam, mon frère, . . .". Ontologiquement parlant, il s'agit d'une coréférence et donc d'une forme alternative. L'étiquette gardée est Sam puisqu'elle est la plus informative lexicalement, tandis que "mon frère" devient un label alternatif de l'entité "Sam" avec l'attribut *skos:altLabel*.
aux:caus	**causative auxiliary**. Marque du causatif. Relation entre l'auxiliaire "faire" et le verbe. Nous créons un groupe verbal composé des deux (auxiliaire + verbe) pour en faire un prédicat unique.

1. Il peut s'agir d'une relation *nsubj* (sujet) pour un "qui" par exemple, ou *obl:comp* pour un "dont".
2. Des adpositions qui jouent le rôle d'adverbes telles que "autour de", "au fur et à mesure de", "à cause de", sont reliées à un verbe par *advmod* et leurs dépendants sont reliés à eux-mêmes par une relation de type objet : "la quantité de pluie"<-[*nsubj*]-"évolue" -[*advmod*]-> "au fur et à mesure de" -[*obl:arg*]-> "l'avancée des saisons".

aux:pass	**passive auxiliary**. Forme passive, où le sujet subit l'action. Relation entre le verbe être et l'action subie. Dans l'ontologie, on enregistre la voie active : "l'arbre a été arraché" devient <None, arracher, arbre>. Dans le cas où le sujet est explicité, "l'arbre a été arraché par le vent" devient <le vent, a arraché, l'arbre>. La voix passive est aussi enregistrée : <l'arbre, a été arraché [par], le vent>.
aux:tense	**tense auxiliary**. Relation entre l'auxiliaire "être" ou "avoir" et son verbe dans les temps composés. Comme pour *aux:caus*, on garde l'auxiliaire et le verbe dans la forme du prédicat.
case	**case marking**. Préposition qui introduit un groupe nominal, contrairement à *mark* qui introduit un groupe verbal. Dans l'ontologie, la relation *hasAdposition* a été ajoutée. L'adposition peut être un indice du type de l'objet, mais n'est pas catégorique (par exemple, "à" peut indiquer un lieu, un horaire ou un destinataire).
cc	**coordinating conjunction**. Chaque conjonction de coordination correspond à une relation logique particulière : énumération (et, ainsi que), alternative (ou, soit), contradiction (mais, tandis que), double négation (ni), cause et conséquence (car, donc) etc. Ainsi, les relations de conjonction héritent d'une de ces catégories. Chaque membre de la conjonction est reliée aux autres membres par cette relation logique.
ccomp	**clausal complement**. Relation entre deux verbes de clauses différentes. Il est transformé en relation *hasObject*.
conj	**conjunction**. Conjonction entre deux mots, peu importe leur partie du discours. En UD, la conjonction relie le premier mot et le deuxième mot entre eux, mais ne fait pas de lien entre les mots énumérés et les arguments de la tête de l'énumération. C'est un problème que EUD résout en ajoutant les liens manquants.
cop	**copula**. En UD uniquement, *cop* relie le verbe être (dépendant) à l'objet (tête) qui est soit un nom soit un adjectif, ce qui n'est pas cohérent avec le reste des relations de type "argument du verbe" en UD. SUD convertit la relation cop en relation *comp:pred* allant du verbe être à l'objet. La relation *comp:pred* de SUD est ensuite convertie en relation *hasObject* pour l'ontologie. Lorsque l'objet est un participe passé, il est détaché du verbe s'il n'y a pas d'autre argument : <il, est, né> mais reste attaché dans le cas contraire : <il, est né [en], 2001>.
csubj	**clausal subject**. Lorsqu'une clause est sujet d'une autre clause. Relie les verbes des deux clauses avec la relation *hasSubject*.
csubj:pass	**clausal passive subject**. Lorsqu'une clause est sujet d'une autre clause à la voix passive, devient objet de l'autre clause à la voix active. Utilise *hasObject* entre les verbes des deux clauses.
det	**determinant**. Déterminant d'un nom, utile pour connaître l'accord en genre et en nombre du nom. Utilise *hasDeterminant* entre l'entité et son déterminant.
expl:comp	**expletive**. Objet non-entité, comme dans la formulation "il y a". Il n'est pas utilisé dans l'ontologie.

expl:pass	**passive expletive**. Sujet explétif d'une forme passive, comme le "me" dans "je me lève". Il devient l'objet du verbe avec *hasObject* : <lève, *hasObject*, me>.
fixed	**fixed**. Indique que deux mots font partie d'une expression figée. Les deux mots sont regroupés sous le même nœud. Même chose avec *flat:foreign* pour les expressions étrangères polylexicales, *flat:name* pour les noms et prénom, ou encore *goeswith* pour les mots découpés par erreur lors de la tokenisation.
iobj	**indirect object**. Objet indirect du verbe. Est remplacé par *hasObject* dans l'ontologie.
iobj:agent	**agentive indirect object**. Agent clitique d'un verbe causatif. Il doit être coréféré pour être résolu car il s'agit d'un pronom. Par exemple, "X lui fait manger Y" devient, avec *iobj:agent*, <lui, manger, Y>. Cependant, le fait que "lui" ait bien effectué l'action de "manger" sur Y n'est pas certain, nous ne pouvons qu'affirmer que <X, fait manger, Y> et <X, fait manger [à], lui>. Utilise la relation *hasObject*.
mark	**marker**. Préposition qui introduit une clause verbale, qui peut être un pronom relatif ou une relation logique entre les deux clauses. La plupart du temps, il peut être converti en relation *hasAdposition*. Mais s'il s'agit d'une relation entre deux verbes, on conservera le marqueur logique comme la relation entre ces derniers : <était fatigué, alors.lr, est endormi>, où la relation "alors.lr" hérite de la relation *LogicalRelation*.
nmod	**noun modifier**. Complément du nom. Comme pour *amod*, la relation hérite de *hasModifier* et est directement utilisée dans sa forme UD dans l'ontologie.
nsubj	**nominal subject**. Sujet du verbe. Dans l'ontologie, il est remplacé par *hasSubject*.
nsubj:pass	**passive nominal subject**. Sujet d'une voix passive, qui devient donc l'objet de l'action à la voix active. On génère un triplet à la voix passive avec *hasSubject*, et un triplet à la voix active avec *hasObject*, sans le verbe "être" de la voix active.
nummod	**numeric modifier**. Donne une quantité à un nom. Dans notre ontologie, la relation *nummod* héritant de *hasModifier* sert à modifier le nom, mais il pourrait aussi être utilisée pour spécifier un nombre d'instances de l'entité qu'il modifie.
obj	**direct object**. Objet direct du verbe. Dans l'ontologie, il est remplacé par *hasObject*.
obj:agent	**agentive oblique**. Sujet du verbe dans la forme causative. Transformé en *hasSubject* à la voix passive et *hasObject* à la voix active.
obl:arg	**oblique argument**. Objet du verbe. Dans l'ontologie, il est remplacé par *hasObject*.
obl:mod	**oblique modifier**. Modifieur oblique du verbe. Dans l'ontologie, il est remplacé par *hasObject*.

xcomp	**open clausal complement**. Objet du verbe qui est lui-même un verbe. Si le deuxième verbe n'a pas de dépendants, alors il devient l'objet du premier verbe avec *hasObject*, comme dans "Alice aime manger" : <Alice, aime, manger> où "manger" est le fait de manger. Mais si le deuxième verbe a lui aussi des dépendants, alors ce dernier est fusionné avec le premier verbe, pour créer un triplet <Alice, aime manger, du chocolat>.

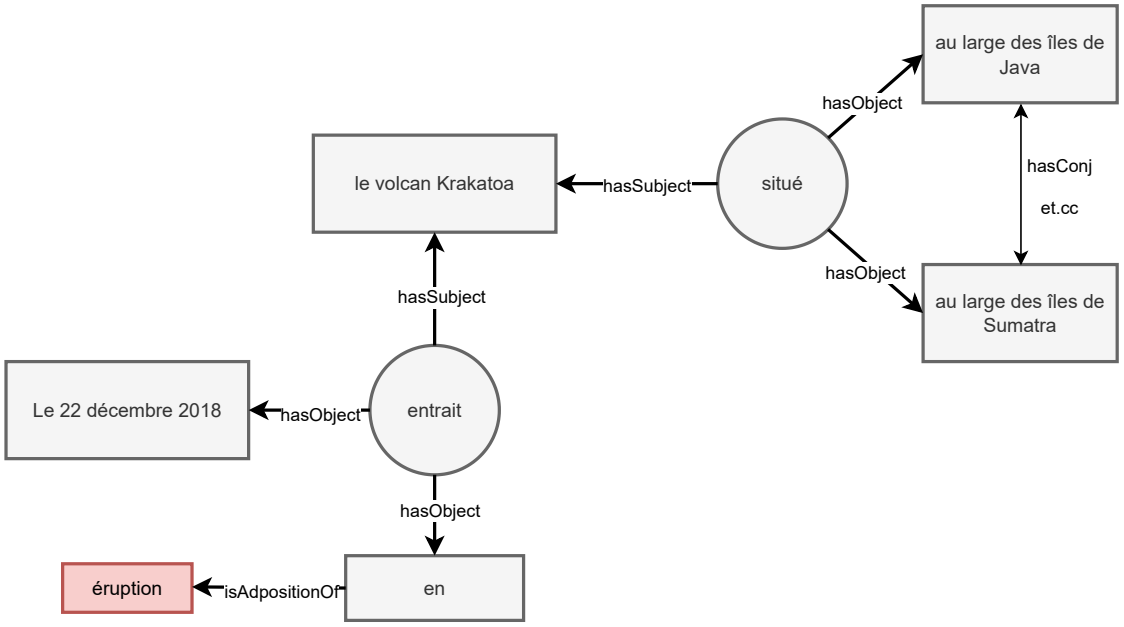

FIGURE A.4 – Exemple d'un graphe de connaissances extrait à partir de l'arbre syntaxique de la phrase "Le 22 décembre 2018, le volcan Krakatoa, situé au large des îles de Java et de Sumatra, entrait en éruption" (source : futura-sciences, phrase simplifiée). Les prédicats (cercles) sont réifiés et leurs arguments sont reliés avec des relations (flèches) spécialement créées pour notre ontologie. Les triplets non-réifiés sont également enregistrés : <le volcan Krakatoa, entrait, le 22 décembre 2018> ; <le volcan Krakatoa, entrait [en], éruption> ; <le volcan Krakatoa, situé, au large des îles de Java> ; <le volcan Krakatoa, situé, au large des îles de Sumatra>.

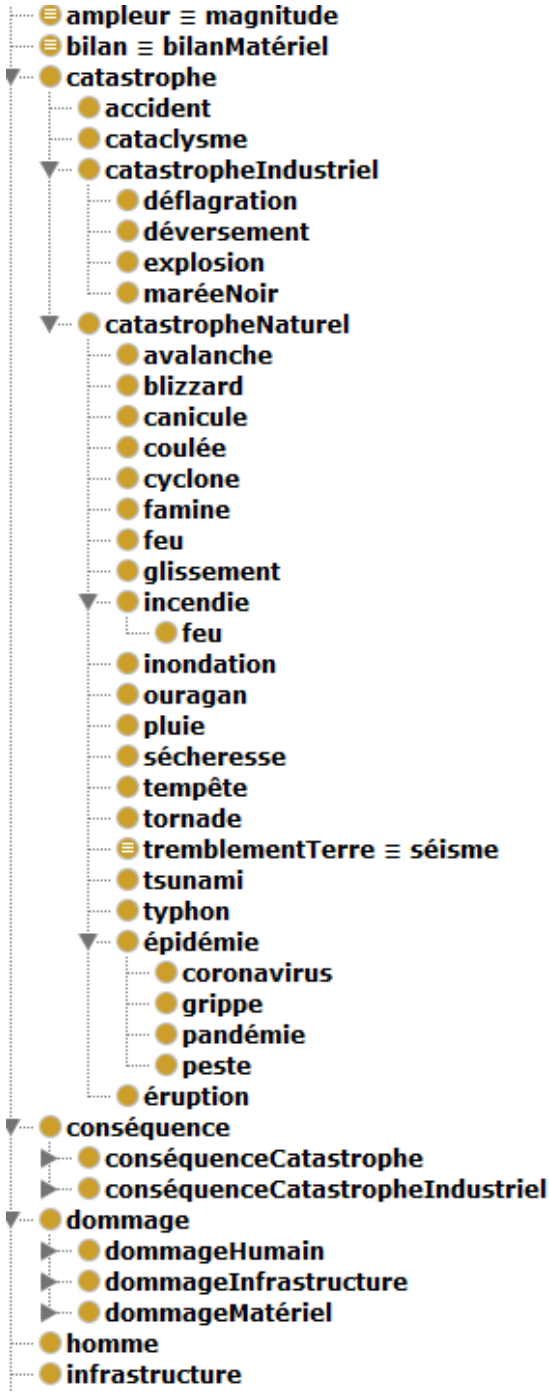

FIGURE A.5 – Capture d'écran dans Protégé de la taxonomie générée automatiquement par le prompt décrivant le corpus Catastrophes.

FIGURE A.6 – Capture d'écran dans Protégé de la taxonomie des types de catastrophes après la fusion des taxonomies générées par le corpus et par le prompt descriptif.

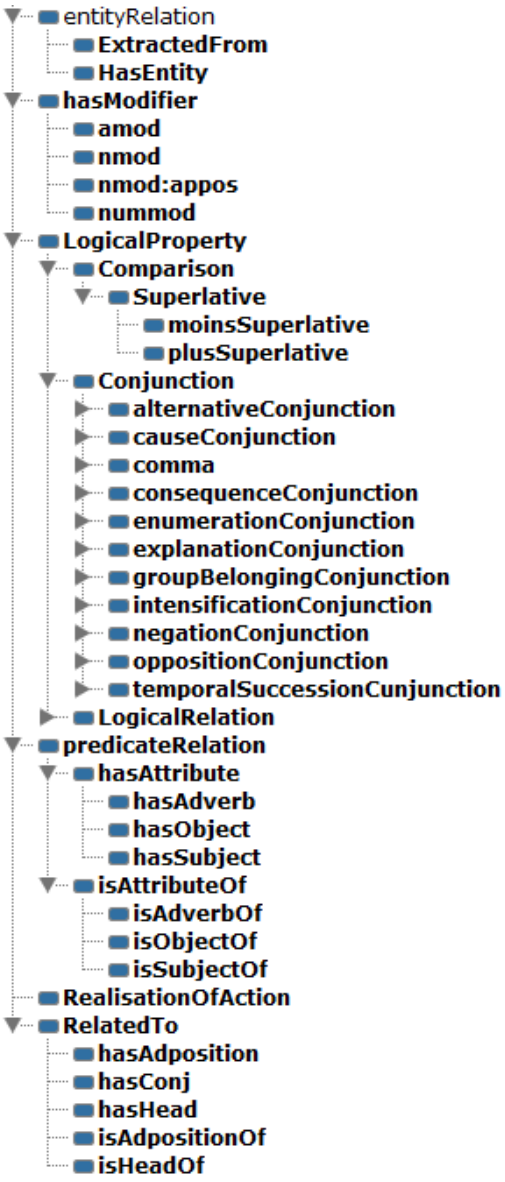

FIGURE A.7 – Capture d'écran dans Protégé des Properties (relations) utilisées dans notre ontologie.