Peuplement d'ontologie à partir de petites annonces immobilières

Céline Alec[1]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

celine.alec@unicaen.fr

Résumé

Le peuplement d'ontologie à partir de textes vise à transformer du contenu textuel en assertions ontologiques. Cela permet d'obtenir une représentation structurée du contenu d'un texte et d'automatiser sa compréhension. Cet article traite d'une approche de peuplement automatique d'une ontologie à partir de petites annonces immobilières. Celle-ci s'appuie à la fois sur une analyse textuelle et une analyse des connaissances du domaine. Des expérimentations, réalisées sur des annonces françaises de ventes de maisons, sont discutées et donnent des résultats encourageants.

Mots-clés

Peuplement d'ontologie, Traitements et raisonnement sur des connaissances, OWL

Abstract

Ontology population from texts transforms textual contents into ontological assertions. A text content has then a structured representation, enabling its understanding. This article deals with an approach to automatically populate an ontology from real estate classified ads. This approach is based on both a text-based and a domain knowledge-based analysis. Experiments, carried out on French house sale ads, are discussed and give encouraging results.

Keywords

Ontology population, Knowledge processing and reasoning, OWL

1 Introduction

Les ontologies [18] permettent de stocker et de partager les connaissances d'un domaine. Elles comportent une hiérarchie de concepts et des relations entre ces concepts. Le processus d'ajout d'instances à une ontologie est appelé peuplement d'ontologie. Une ontologie peuplée peut également être appelée base (ou graphe) de connaissances.

L'approche proposée dans cet article fait partie du projet DECA (Détection d'Erreurs et Correction d'Annotations). Ce projet s'intéresse à des descriptifs annotés, c'est-à-dire des descriptions textuelles, auxquelles sont ajoutées des annotations. Par exemple, c'est le cas des petites annonces, annotées avec les critères auxquels elles répondent. Les annotations sont théoriquement censées décrire les caractéristiques de l'objet ou de l'événement décrit dans la description. Cependant, ce n'est pas toujours le cas. En effet, on observe fréquemment des annotations erronées, soit à cause de fautes de frappe, soit à cause d'un « détournement d'usage » : les descriptions sont délibérément mal annotées afin d'augmenter leur visibilité. Par exemple, on peut trouver une annonce immobilière dont la ville annotée est X, alors que la description indique « à 30 minutes de X » ; ou dont l'annotation de la superficie est de 150 m^2 alors que la description indique « 147 m^2 ». Ces mauvaises annotations font perdre un temps considérable aux utilisateurs, qui doivent trier les multiples réponses répondant en théorie à leurs critères de recherche. Le projet DECA s'attaque à ce problème. Son objectif est de détecter et de corriger automatiquement les annotations erronées grâce aux incohérences qui peuvent être trouvées en confrontant les annotations et la description textuelle. Une telle confrontation nécessite d'assimiler les éléments essentiels du texte, autrement dit, de comprendre le texte comme le ferait un humain. C'est sur ce sous-problème que se concentre notre contribution. Notre objectif est de représenter le contenu des descriptions textuelles d'objets dans une ontologie de domaine. Il s'agit donc d'un problème de peuplement d'ontologie à partir de descriptions textuelles. Notre premier cas d'utilisation concerne des annonces de vente de maisons, mais l'approche proposée doit être aussi générique que possible, c'est-à-dire qu'elle doit être capable de peupler une ontologie de domaine à partir de descriptions dans de nombreux domaines, qu'il s'agisse de petites annonces (vente de véhicules, mode, etc.), ou de descriptions d'un certain type d'objet (restaurants, hôtels, etc.). Une ontologie du domaine concerné est considérée en entrée ; sa conception n'est pas l'objet de notre travail.

Le reste du document est organisé comme suit : la Section 2 présente les travaux connexes et positionne notre contribution. La Section 3 décrit notre approche. Les expérimentations sur notre cas d'utilisation sont détaillées dans la Section 4. La Section 5 conclut et suggère des travaux futurs.

2 Travaux proches et positionnement

Le peuplement d'ontologie a été étudié dans divers articles scientifiques. L'étude [12] présente un aperçu des travaux sur ce sujet. Dans cette section, nous nous concentrons uniquement sur les approches visant à extraire des informations de documents textuels non structurés et d'une ontologie de domaine donnée en entrée, en permettant l'ajout d'instances dans cette dernière, en particulier l'ajout d'assertions de propriétés. Les systèmes existants sont basés

sur diverses méthodes à base de règles utilisant des modèles lexico-syntaxiques (cf. paragraphe suivant) ; ou à base d'apprentissage automatique, [9, 19] ; ou encore sur des méthodes hybrides [3]. Certaines approches récentes utilisent l'apprentissage profond. Pour cela, les données textuelles sont exploitées pour produire un modèle de langage basé sur des plongements de mots. C'est le cas de [2] qui vise à peupler une ontologie dans le domaine biomoléculaire, ou de [6] qui utilise un LSTM pour peupler une ontologie traitant de cybersécurité. De façon générale, l'utilisation de techniques d'apprentissage automatique nécessite de disposer en amont d'une quantité suffisante de phrases et de leur correspondance ontologique, ce qui n'est pas le cas de nos données. Nous nous concentrons donc ici sur les approches qui exploitent des patrons lexico-syntaxiques.

L'approche ArtEquAKT [1] s'intéresse à du peuplement d'ontologie à partir du Web dans le domaine des artistes. Elle peuple l'ontologie avec des assertions de propriétés. Pour cela, le verbe trouvé dans une phrase entre deux instances de concept de l'ontologie est exploité. Sur le même principe, Makki [13] se concentre également sur les verbes afin de peupler l'ontologie avec des assertions de propriétés, mais l'approche est semi-automatique et indépendante du domaine. Une liste de verbes est extraite du corpus d'entrée pour chaque propriété de l'ontologie en utilisant Wordnet. Un ensemble de sept règles écrites manuellement est utilisé pour reconnaître les sujets et les objets d'une assertion de propriété potentielle. Les résultats sont ensuite validés par un expert. Dans [5], un framework est proposé pour instancier un concept et les relations qui le concernent. Tout d'abord, les entités nommées sont identifiées dans le texte (en exploitant également les co-références). Ensuite, des déclencheurs sont pris en compte, c'est-à-dire les noms de propriétés ainsi que leurs synonymes ; et des règles sont construites sur la base des phrases nominales précédées ou suivies d'un déclencheur. L'application de ces règles conduit au peuplement de l'ontologie. [16] présente une approche pour peupler une ontologie d'évènements criminels et leurs causes à partir de tweets de journaux espagnols. L'approche se base sur des patrons linguistiques. [11] a pour but d'extraire des assertions de propriété dans des documents règlementaires entre des incidents et des mesures (propriété « hasMeasure » et ses sous-propriétés) en se basant sur des règles. Les co-occurrences d'incidents et de mesures au sein d'une même phrase, paragraphe ou chapitre, sont utilisées ; mais cela ne permet pas de distinguer les sous-propriétés. Des patrons lexicaux sont employés dans ce cas. [15] présente une approche à base de règles pour extraire des relations à partir d'anecdotes musicales et peupler une ontologie. Elle exploite des règles basées sur des étiquetages grammaticaux. Le framework T2KG [10] se base sur des règles pour traduire du texte en triplets et utilise une approche hybride (règles et similarité) transformant les prédicats textuels en ceux d'un graphe de connaissances.

L'étude de ces approches montre plusieurs obstacles scientifiques émergents. En général, les méthodes à base de règles présentent une bonne précision au détriment du rappel [4]. Soit les règles sont propres à un domaine (cas des patrons lexicaux et de certains patrons syntaxiques très précis), soit elles sont génériques. Dans ce dernier cas, elles ne peuvent pas être aussi précises que des règles définies pour un seul domaine. Beaucoup de ces travaux nécessitent une intervention humaine pour valider les propositions trouvées. De plus, le verbe présente en général une grande importance dans le peuplement des propriétés, car il est fortement caractéristique d'une relation (par ex., « est marié à »). Enfin, la plupart des approches s'intéressent à des entités nommées en sujet et objet des propriétés, et se focalisent sur le peuplement des propriétés objet (object properties) au détriment des propriétés typées (data properties).

Dans notre contexte, nous souhaitons pouvoir appliquer une même approche sur plusieurs domaines, en restant néanmoins toujours dans le cadre des descriptions textuelles d'objets. Comme l'approche proposée sera la première étape en vue d'une correction automatique d'annotations, elle se doit d'être automatique, sans aucune validation humaine, et suffisamment générique. En général, les verbes ne sont pas très caractéristiques d'une relation dans les descriptions d'objets (par ex., « a » ou « possède »). Parfois, il peut n'y avoir aucun verbe (par ex., « 2 chambres, 1 salle de bain. »), ce qui complique le peuplement. Les propriétés objet et typées sont importantes. Les sujets et objets ne sont pas nécessairement des entités nommées. Enfin, notre contexte peut présenter des propriétés n-aires [1], qui représentent des notions complexes difficiles à peupler. Pour toutes ces raisons, les travaux cités ne sont pas adaptés à notre problématique originale, qui nécessite l'établissement d'une nouvelle approche.

3 L'approche KOnPoTe

Nous présentons KOnPoTe (Knowledge graph/ONtology POpulation from TExts), une approche pour peupler une ontologie de domaine à partir de descriptions textuelles d'éléments de ce domaine. Elle prend en entrée un corpus de descriptions ainsi qu'une ontologie du domaine et peuple l'ontologie en représentant les descriptions du corpus.

3.1 Les données initiales

L'ontologie initiale définit le domaine. Plus formellement, elle peut être définie comme un tuple ($\mathcal{C}$, $\mathcal{P}$, $\mathcal{I}$, $\mathcal{A}$, $\mathcal{R}$) où $\mathcal{C}$ est un ensemble de classes, $\mathcal{P}$ un ensemble de propriétés (objet et typées) caractérisant les classes, $\mathcal{I}$ un ensemble d'individus et d'assertions (potentiellement vide), $\mathcal{A}$ un ensemble d'axiomes représentables en OWL2 [2] et $\mathcal{R}$ un ensemble de règles SWRL [3] [8] (potentiellement vide). Elle peut être existante, construite manuellement ou (semi-) automatiquement. Sa conception ne fait pas partie de notre contribution. Le domaine y est représenté par une classe nommée ci-après *classe principale*. Les propriétés typées prises en compte peuvent utiliser des valeurs booléennes, numériques ou des chaînes de caractères. Elle peut contenir des individus initiaux, qui sont génériques. Chaque entité

1. Par exemple, exprimant qu'un bien se situe à une distance d'un lieu.
2. Web Ontology Language
3. Semantic Web Rule Language

de l'ontologie (classe, propriété, individu) possède un identifiant (URI) et éventuellement une terminologie plus avancée, via rdfs:label, rdfs:isDefinedBy, ainsi qu'une propriété d'annotation « unité », spécialement créée pour associer une entité à son unité ou à une expression d'unité. Cette particularité est exemplifié dans le paragraphe suivant. Le corpus utilisé en entrée est composé de documents qui décrivent chacun une instance de la *classe principale*.

FIGURE 2 – Vision partielle des entités de l'ontologie

L'approche est conçue pour être applicable à différents domaines de descriptions textuelles. L'exemple déroulé dans ce papier considère le domaine des ventes de maisons. La Figure 2 représente une vision partielle des classes et propriétés de l'ontologie utilisée dans ce cadre. Celle-ci contient des classes telles que la classe principale *Bien* (désignant un bien immobilier) ; des classes désignant des pièces comme *PièceDeMaison*, *Cuisine*, *Chambre* ; etc. Parmi les propriétés, on peut citer la propriété objet *seSitueA* reliant un bien immobilier à la commune dans laquelle il est situé, ou la propriété typée *surfaceEnM2* reliant une partie de bien (terrain, maison, etc.) ou une pièce à une valeur numérique. Les individus initiaux (non représentés sur la figure) sont génériques, par exemple, des instances de la classe *Commune*, ou de la classe *SystèmeDeChauffage*. L'ontologie contient également des axiomes, par exemple, le fait qu'un *Bien* ne peut être situé que dans maximum une *Commune*, ou le fait qu'une *Chambre* est disjointe d'une *Cuisine*. Enfin, certaines règles SWRL sont définies, par exemple, pour exprimer le fait que la propriété booléenne *séparé* doit être peuplée avec la valeur opposée de la propriété booléenne *ouvert*. La propriété *surfaceEnM2* est associée à son unité « m^2 », et la propriété *pourcentageHonoraires* à une expression d'unité « honoraires : xxx % ». Un changement dans la façon de représenter les unités (en utilisant, par exemple, une classe spéciale d'unités liée à un nom d'unité et à une valeur d'unité) peut être envisagé dans une version future.

3.2 Modélisation de l'approche

L'approche proposée se doit d'être applicable à différents domaines. Pour un domaine donné, il est nécessaire d'avoir en entrée une ontologie du domaine, ainsi qu'un corpus de documents, où chaque document est un texte qui décrit une instance de la *classe principale*. Ainsi, l'algorithme proposé ne doit pas dépendre de règles ou de modèles linguistiques basés sur le domaine. Nous avons donc choisi d'utiliser la terminologie du domaine (issue de l'ontologie), des indicateurs syntaxiques (comme les phrases ou l'ordre des expressions dans le texte) et des indicateurs de connaissances (comme les domaines et co-domaines de propriétés).
La Figure 1 montre les grandes lignes de l'approche. L'ontologie (O), ainsi que chaque document du corpus, sont utilisés (haut de la Figure) par un comparateur de terminologie. Cela conduit à des correspondances entre les mentions du texte et les entités de O (classes, propriétés et individus). Ensuite, un algorithme de peuplement est appliqué, pour obtenir l'ontologie peuplée. Cet algorithme de peuplement est une succession de plusieurs traitements (milieu de la Figure) : une initialisation d'un objet appelé « Traitements des correspondances » (TC) ; une instanciation de la *classe principale* ; une analyse textuelle, composée de divers traitements (bas de la Figure) ; ainsi qu'une analyse basée sur les connaissances de l'ontologie. Cela est appli-

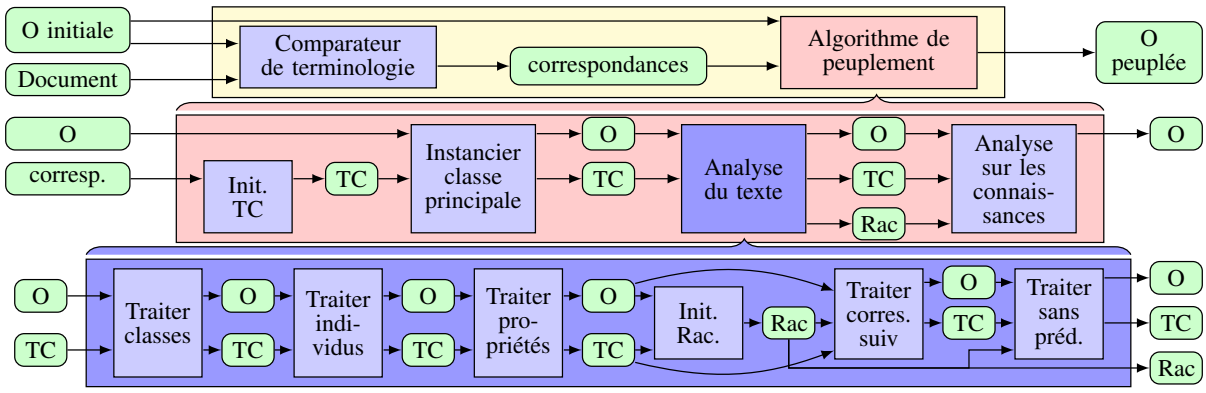

FIGURE 1 – Vision schématique de l'approche KOnPoTe

qué sur chaque document du corpus, permettant d'obtenir à la fin une ontologie représentant toutes les descriptions du corpus. La suite de cette section détaille chaque traitement en déroulant un exemple d'annonce de vente d'une maison.

3.3 Le comparateur de terminologie

Le comparateur de terminologie est la première étape de l'approche. Il prend en entrée l'ontologie initiale et un document, et produit des correspondances entre les mentions textuelles du document et les entités de l'ontologie (cf. Figure 1). Le texte est découpé en phrases et lemmatisé. Les mots-clés de l'ontologie (fragments d'URI, labels, unités) sont aussi lemmatisés. Des correspondances sont établies entre le texte et les mots-clés de l'ontologie. Celles-ci concernent également les expressions d'unité, par exemple, la mention « *honoraires : 4%* » correspond à l'expression d'unité « *honoraires :* xxx % ». Les inclusions sont ignorées. Par exemple, s'il existe une correspondance sur la mention « centre ville » et sur sa sous-mention « ville », seule celle sur « centre ville » est considérée.

Cormelles-le-Royal : Magnifique maison à 15 min du centre-ville de Caen et à 3 min à pied des commerces et des écoles, 110 m^2 sur un terrain de 400 m^2. Le rez-de-chaussée est composé d'une cuisine équipée, d'un salon avec cheminée, exposé sud-ouest, ainsi que d'une chambre de 15 m^2. Premier étage : 2 chambres et 1 sdb. Proche des transports en commun. Terrain arboré et clos. Honoraires : 4%.

FIGURE 3 – Exemple d'un texte et ses correspondances

La Figure 3 montre un exemple fictif d'une annonce immobilière. Les correspondances obtenues concernent des individus (ex : « rez-de-chaussée » $\leftrightarrow$ *RDC*), des classes (« maison » $\leftrightarrow$ *Maison*), des propriétés objet (« Proche des » $\leftrightarrow$ *estProcheDe*) ou typées (« m^2 » $\leftrightarrow$ *surfaceEnM2*).

3.4 L'algorithme de peuplement

Les correspondances sont fournies en entrée à l'algorithme de peuplement. Ce dernier comporte quatre tâches principales (cf. Figure 1), dont les deux premières sont du prétraitement. La troisième, appelée analyse du texte, est une tâche de peuplement utilisant les correspondances et des indicateurs textuels. La dernière tâche, appelée analyse sur les connaissances, vise à ajouter des assertions de propriétés en se basant sur les connaissances de l'ontologie. Ces tâches sont décrites dans la suite de cette section.

3.4.1 Les deux tâches de pré-traitement

Initialisation des traitements de correspondances La première tâche consiste à initialiser les traitements de correspondances $TC = \{tc_1, tc_2, \ldots, tc_n\}$, dont on se servira par la suite. Chaque tc correspond à une correspondance et contient trois attributs (les individus, les assertions, les individus précédents liés) initialement vides.

Instanciation de la classe principale La tâche suivante consiste à instancier la classe principale, pour représenter le document en cours de traitement. Ainsi, un nouvel individu, appelé ci-après l'*instance principale*, instance de la classe principale, est ajouté à O. Dans l'exemple de document de la Figure 3, l'individu $bien1$ est créé avec l'assertion $<bien1, isA, Bien>$ [4] ($Bien$ étant la classe principale, représentant un bien immobilier).

3.4.2 Analyse du texte

Cette tâche est détaillée au bas de la Figure 1. Elle analyse les correspondances (via les TC) et vise à ajouter des individus ainsi que des assertions de classe et de propriété en considérant des indicateurs textuels.

Traitement des correspondances de classe Chaque correspondance concernant une classe (sauf la classe principale) est analysée, dans le but de créer un nouvel individu, instance de cette classe. Pour l'exemple en Figure 3, la mise à jour des TC peut être observée (individus et assertions ajoutés) sur les lignes dont le type est « classe » dans le Tableau 1. Ces individus et assertions sont ajoutés à O.
Dans le document, le(s) mot(s) précédent(s) une correspondance sont comparés à une liste de mots-clés exprimant la négation (ex : « pas de »), et la correspondance est ignorée si une trace de négation est trouvée. Par exemple, dans « pas de garage », la correspondance avec « garage » est ignorée, aucune instance de garage n'est créée. Si le mot précédent correspond à un nombre, autant d'individus que le nombre trouvé sont créés. Par exemple, « 2 chambres » engendre la création de deux instances de la classe $Chambre$.

Traitement des correspondances d'individu Ensuite, les TC sont mis à jour pour les correspondances avec des individus de O, en ajoutant les individus concernés. On peut l'observer, pour l'exemple étudié, sur les lignes correspondant au type « individu » dans le Tableau 1.

Traitement des correspondances de propriété Une correspondance avec une propriété doit permettre l'instanciation de cette dernière. Pour ce faire, il faut établir le sujet et l'objet de l'assertion à créer. Ce traitement est décrit dans la suite et est explicité sur des exemples en fin de paragraphe.
Une liste de sujets possibles est considérée. Pour l'instancier, les correspondances candidates sont parcourues, en commençant par celle qui précède la correspondance de propriété en cours de traitement jusqu'au début de la phrase considérée. Dès qu'un tc candidat possède un/des individus qui appartiennent au domaine de la propriété, ces individus constituent la liste des sujets possibles. Pour les objets, tout dépend du type de la propriété à instancier. Dans le cas d'une propriété objet, on considère la correspondance qui suit directement la mention de la propriété. Si une telle correspondance existe, tous les individus de son tc qui sont dans le co-domaine de la propriété sont considérés comme des objets possibles. Dans le cas d'une propriété typée, il existe plusieurs possibilités :
- Si la correspondance vient d'une unité, alors le mot précédent est pris. Dans l'exemple, « chambre de 15 m^2 » a une correspondance entre « m^2 » et *surfaceEnM2* (qui a comme unité m^2). La propriété est instanciée avec la valeur 15.
- Si la correspondance vient d'une expression d'unité, alors la partie qui correspond à la variable est considérée. Par exemple, « Honoraires : 4% » a une correspondance avec la

4. Si un document présente des correspondances avec la classe principale, alors leurs tc sont mis à jour avec l'individu et l'assertion créés. Ce n'est pas le cas dans l'exemple car « bien » ne fait pas partie du texte.

TABLEAU 1 – Les TC de l'exemple après traitements des correspondances de classe, d'individu et de propriété

Correspondance	**Type**	**Individus**	**Assertions**	**Ind. préc.**
Cormelles-le-Royal	individu	Cormelles-le-Royal		
maison	classe	maison1	<maison1, isA, Maison>	
à	propriété objet			
min	propriété typée	distance1	<distance1, isA, Distance><distance1, minVoiture, 15>	
centre-ville	individu	centre-ville		
Caen	individu	Caen		
à	propriété objet			
min à pied	propriété typée	~~distance1~~ distance2	~~<distance1, minPied, 3>~~ <distance2, isA, Distance><distance2, minPied, 3>	
commerces	individu	commerces		
écoles	individu	écoles		
m^2	propriété typée	maison1	<maison1, surfaceEnM2, 110>	
terrain	classe	terrain1	<terrain1, isA, Terrain>	
m^2	propriété typée	terrain1	<terrain1, surface, 400>	
rez-de-chaussée	individu	RDC		
cuisine	classe	cuisine1	<cuisine1, isA, Cuisine>	
équipée	propriété typée	cuisine1	<cuisine1, équipé, true>	
salon	classe	salon1	<salon1, isA, Salon>	
cheminée	classe	cheminée1	<cheminée1, isA, Cheminée>	
exposé	propriété typée	salon1	<salon1, exposition, sud-ouest>	
chambre	classe	chambre1	<chambre1, isA, Bedroom>	
m^2	propriété typée	chambre1	<chambre1, surfaceEnM2, 15>	
Premier étage	individu	premierEtage		
chambres	classe	chambre2 chambre3	<chambre2, isA, Chambre> <chambre3, isA, Chambre>	
sdb	classe	salleDeBain1	<salleDeBain1, isA, SalleDeBain>	
Proche des	propriété objet	bien1	<bien1, estProcheDe, transports_en_commun>	
transports en commun	individu	transports_en_commun		bien1
Terrain	classe	terrain2	<terrain2, isA, Terrain>	
Honoraires : 4%	propriété typée	bien1	<bien1, pourcentageHonoraires, 4>	

propriété *pourcentageHonoraires* (via l'expression d'unité *Honoraires : xxx%*), qui est instanciée avec la valeur 4.

- Si la propriété est booléenne, alors on regarde s'il y a une trace de négation avant. La même liste de mots-clés de négation que dans le traitement des correspondances de classe est utilisée. Par exemple, la propriété booléenne *aménageable* est instanciée avec true pour « le grenier est aménageable » et false pour « le grenier n'est pas aménageable ».
- Si le co-domaine de la propriété est une liste de choix, c'est-à-dire une expression utilisant *owl*:*oneOf*, le(s) mot(s) suivant(s) dans le texte sont comparés aux choix possibles. Par exemple, si la propriété *exposition* a pour valeurs possibles {nord, sud, est, ouest}, alors « exposition sud », conduit à une instanciation avec la valeur « sud ».
- Dans les autres cas, le mot suivant est pris. Par exemple, le texte « construit en 2005 » instancie la propriété *anneeDeConstruction* avec la valeur 2005.

Pour chaque sujet et objet possible, une assertion de la propriété considérée est ajoutée, à condition que celle-ci ne crée pas d'incohérence dans O. Le tc est mis à jour : la ou les nouvelles assertions et leur(s) individu(s) sujet(s) sont ajoutés respectivement aux attributs *assertions* et *individus*. Le tc représentant l'objet est également mis à jour : le sujet de l'assertion est ajouté dans son attribut *individus précédents liés* (attribut exploité dans la suite). Si aucune assertion ne peut être ajoutée, alors le processus est répété avec une nouvelle instance du domaine de la propriété comme sujet. Une fois que toutes les correspondances de propriétés ont été traitées, tous les individus de O qui sont reconnus comme équivalents sont fusionnés, et TC est mis à jour en fonction de cette fusion.

Le Tableau 1 (cf. les lignes dont le type est une propriété) montre le traitement des correspondances de propriété sur l'exemple. Tout d'abord, la mention « à » fait référence à la propriété objet `seSitueA` dont le co-domaine est une commune. Elle est suivie d'une valeur numérique et non d'une correspondance avec une commune. Par conséquent, l'ensemble des objets possibles est vide et la propriété ne peut pas être instanciée. La correspondance sur « min » fait référence à la propriété `distanceMinVoiture` associant une distance à une valeur numérique en minutes. Elle porte sur une unité (« min » est une unité associée à cette propriété), on considère donc le mot précédent comme objet : 15. Il n'y a pas d'instance de la classe `Distance`, l'ensemble des sujets possibles est donc initialement vide, et on considère une nouvelle instance de `Distance` ($distance1$[5]). Pour la correspondance sur « min à pied », le processus est le même, sauf que l'ensemble des sujets

5. Les nouveaux individus sont nommés en fonction de la classe dont ils sont des instances (par exemple, $distance1$ et $distance2$ pour la classe $Distance$). Si le domaine à instancier est une expression de classe, alors les nouveaux individus sont nommés $indiv1$, $indiv2$, etc.

possibles considère $distance1$ puisqu'il s'agit d'un individu résultant d'un tc avant celui que l'on traite, issu de la même phrase, et dans le domaine de la propriété. On tente donc d'ajouter une assertion <$distance1, minAPied, 3$> mais celle-ci est incohérente (car cette distance représente déjà 15 min en voiture). Un nouvel individu $distance2$ est créé pour le sujet. On peut observer dans le tableau toutes les assertions créées. Notons que, comme la correspondance portant sur « Proche des » conduit à l'assertion <$bien1, estProcheDe, transports\_en\_commun$>, l'instance $bien1$ est ajoutée comme individu précédent lié dans le tc de « transports en commun ».

Initialisation des raccrochabilités Dans un contexte descriptif, les verbes ne sont pas très significatifs (cf. Section 2). Il est donc très probable d'avoir manqué, à ce stade, des assertions de propriétés. Le reste de l'algorithme de peuplement vise à ajouter celles-ci. Le défi ici est d'ajouter les assertions manquantes (ce qui augmenterait le rappel) sans en ajouter trop d'erronées (ce qui diminuerait la précision). Dans ce contexte, nous introduisons un ensemble appelé *les raccrochabilités* (Rac). Son initialisation consiste à créer toutes les raccrochabilités $Rac(i)$, pour chaque individu i des TC. Un élément de $Rac(i)$ est un tuple *(propriété, expression de co-domaine)*, tel que i est raccrochable (via la propriété) à un individu appartenant à l'expression de co-domaine. En d'autres termes, pour un individu i, on cherche chaque propriété $prop$ pour laquelle i peut être un sujet. Si i peut être sujet de $prop$, on cherche l'expression de co-domaine à laquelle doit nécessairement appartenir un objet obj d'une éventuelle assertion <i,$prop$,obj>. Pour chaque individu i, l'ensemble des raccrochabilités $Rac(i)$ est automatiquement construit.

Dans l'exemple, prenons l'instance principale $bien1$, qui appartient au domaine de la propriété $contient$: une raccrochabilité est établie. Or, $bien1$ est une instance de la classe $Bien$, sous-classe de l'expression `contient only PartieDeBien`. Ainsi, l'expression de co-domaine associée à $bien1$ et $contient$ est l'intersection du co-domaine de $contient$ (c'est-à-dire `PartieDeBien or PièceDeMaison`) et de la classe $PartieDeBien$ (issue de la définition avec `only`). Le tuple ($contient$, $PartieDeBien$) est donc une raccrochabilité pour $bien1$. Cela signifie que $bien1$ ne pourra être le sujet d'une assertion de $contient$ qu'avec des instances de $PartieDeBien$.

Les raccrochabilités $Rac(i)$ sont exploitées dans les étapes suivantes pour trouver la propriété la plus adaptée pour relier un sujet i avec un objet j. L'ensemble $Rac(i)$ sera parcouru jusqu'à trouver une raccrochabilité telle que j appartienne à l'expression de co-domaine de cette dernière.

$Rac(i)$ est un ensemble trié. Dans le cas où plusieurs propriétés seraient candidates, nous voulons trouver la meilleure, ce qui n'est pas une tâche évidente. Nous avons choisi de donner la priorité à la spécificité. Les premiers éléments sont les raccrochabilités dont le co-domaine est le plus spécifique, puis celles dont le domaine de la propriété est le plus spécifique. Sinon, le tri est arbitraire, en fonction de l'URI de la propriété. Par exemple, si une instance de $Bien$ a pour raccrochabilité l_1 = ($seSitueA$, $Commune$) et l_2 = ($estProcheDe$, $Lieu$) tels que $Commune$ est une sous-classe de $Lieu$, alors ils seront triés dans l'ordre $l_1 < l_2$. Ainsi, si une assertion doit être ajoutée entre $bien1$ et une instance de commune, la propriété choisie sera $seSitueA$ et non $estProcheDe$, puisque le co-domaine de l_1 est plus spécifique que celui de l_2. Dans l'exemple, $Rac(bien1) = \{(seSitueA, Commune), (estProcheDe, Lieu), (seSitueADistance, Distance), (contient, PartieDeBien)\}$.

Traitement des correspondances suivantes Cette étape consiste à vérifier s'il est possible de lier le ou les individus résultant d'une correspondance avec celui ou ceux de la correspondance suivante dans le texte, provenant de la même phrase. Des assertions de propriété sont ajoutées lorsque cela est possible. Les propriétés n-aires sont prises en compte. Une succession de correspondances impliquant respectivement des individus a, b et c peut conduire à des assertions de propriétés du type <a,...,b> et <a,...,c>. Ici, le but est de lier a et b, qui sont issus de correspondances consécutives dans le texte, mais aussi a et c, qui ne le sont pas. Pour pouvoir le faire, on exploite les individus précédents liés des TC.

L'idée générale est de regarder si un individu ($sujet$), issu de la correspondance examinée, peut être le sujet d'une assertion ayant pour objet un individu de la correspondance suivante ($objet$), si celle-ci est dans la même phrase. Si la correspondance suivante n'a pas d'individu associé, alors celle d'après est considérée et ainsi de suite. Si $sujet$ et $objet$ ne sont pas déjà reliés entre eux, il peut manquer une assertion. Pour choisir la meilleure propriété possible entre ces deux individus, on considère l'ensemble trié $Rac(sujet)$ et prend la première propriété d'une raccrochabilité telle que $objet$ est dans l'expression de co-domaine de cette dernière et telle qu'elle n'ajoute aucune incohérence à O. L'assertion est ajoutée au tc de $sujet$, et $sujet$ est également ajouté comme un individu précédent lié du tc de l'objet. Si aucune assertion n'est possible entre les individus de

TABLEAU 2 – Les TC de l'exemple étudié après le traitement des correspondances suivantes

Mention	**Individus**	**Assertions**	**Ind. préc.**
Cormelles	Cormelles		
maison	maison1	<maison1, isA, Maison>	
à			
min	distance1	<dist1, isA, Distance> <dist1, minVoiture, 15> <dist1, distDuPI, centre-ville> (1)	
centre-ville	centre-ville	<dist1, distDeLaVillle, Caen> (2)	dist1 (1)
Caen	Caen		dist1 (2)
à			
min à pied	distance2	<dist2, isA, Distance> <dist2, minPied, 3> <dist2, distDuPI, commerces> (3)	
commerces	commerces	<dist2, distDuPI, écoles> (4)	dist2 (3)
écoles	écoles		dist2 (4)
...	...	...	...
salon	salon1	<salon1, isA, Salon> <salon1, aPrElmt, cheminée1> (5)	
cheminée	cheminée1	<cheminée1, isA, Cheminée>	salon1 (5)
exposé	salon1	<salon1, exposition, sud-ouest>	
...	...	...	...

deux correspondances consécutives, alors nous essayons de faire une assertion avec les individus précédents liés comme sujet, afin de faciliter le peuplement des propriétés n-aires. Une fois que tout le texte est examiné, les individus de O reconnus comme équivalents par un moteur d'inférence sont fusionnés, et TC est mis à jour en fonction de cette fusion. Le Tableau 2 montre les TC de l'exemple après cette étape. D'abord, on essaie de lier *Cormelles* à *maison1* (impossible), puis *maison1* à *distance1* (impossible) et ainsi de suite. On peut relier *distance1* à *centre-ville* via la propriété *distanceDuPointDIntérêt* (1). L'assertion est ajoutée dans le tc du sujet (*distance1* portant sur la mention « min »). Le tc suivant (sur la mention « centre-ville ») est mis à jour avec l'individu précédent lié *distance1*. Ensuite, lorsqu'on essaie de lier le tc sur « centre-ville » avec le suivant (sur « Caen »), nous ne pouvons pas lier les individus associés (*centre-ville* et *Caen*). Néanmoins, nous pouvons lier l'individu précédemment lié (*distance1*) avec *Caen*. C'est ce qui est fait dans (2). Et ainsi de suite, on obtient (3), (4) et (5).

Traitement des individus sans prédécesseurs La dernière étape de l'analyse textuelle consiste à traiter les individus sans prédécesseurs. En effet, chaque document décrit une instance de la classe principale et on s'attend à ce que cette instance soit le point de départ des assertions de propriétés. Par conséquent, il semble assez intuitif de penser que chaque individu considéré, sauf l'instance principale, doit être l'objet d'au moins une assertion. L'objectif est donc de trouver des sujets et des propriétés possibles pour les individus (sauf l'instance principale) n'ayant aucun prédécesseur, c'est-à-dire, n'étant pas l'objet d'une assertion de propriété. Pour minimiser le risque de relier des individus qui n'ont rien à voir entre eux, on se concentre uniquement sur les individus résultant de correspondances d'une même phrase. Cela signifie que, pour chaque individu sans prédécesseur ($objet$) d'une phrase, on essaie de lui relier un autre individu de cette phrase ($sujet$), afin d'obtenir l'assertion <$sujet$, $prop$, $objet$>, où la propriété $prop$ choisie est la « meilleure » au sens des raccrochabilités. Enfin, les individus de O qui sont reconnus comme équivalents sont fusionnés, et TC est mis à jour en fonction de cette fusion. La Figure 4 montre les individus (nœuds) et assertions (arêtes) ajoutés avant cette étape pour l'exemple étudié. Les individus sans prédécesseurs (sauf l'instance principale $bien1$) sont grisés. Le Tableau 3 détaille le traitement effectué à cette étape. Chaque ligne correspond à une phrase. Les individus sans prédécesseurs sont en italique. Pour chacun d'eux, on cherche s'il est possible d'ajouter une assertion ayant pour sujet un individu de la même phrase. Par exemple, pour $Cormelles$, on cherche si on peut ajouter des assertions <$maison1$,...,$Cormelles$>, <$dist1$,...,$Cormelles$>, etc. Les seules possibilités pour la première phrase sont données dans la dernière colonne mais elles ne sont pas ajoutées car elles sont incohérentes par rapport à O. En effet, $dist1$ et $dist2$ concernent déjà d'autres lieux et ne peuvent pas concerner $Cormelles$. On répète ce principe pour chaque phrase, et obtient les assertions mentionnées dans le Tableau 3, qui sont ajoutées dans O et TC.

TABLEAU 3 – Traitement des individus sans prédécesseurs

#	Individus	Assertions ajoutées
1	*Cormelles*, *maison1*, *dist1*, centre-ville, Caen, *dist2*, commerces, écoles, *terrain1*	~~<dist1, distDeLaVille, *Cormelles*>~~ ~~<dist2, distDeLaVille, *Cormelles*>~~ non ajoutées car incohérentes
2	*RDC*, *cuisine1*, *salon1*, *chambre1*	<cuisine1, seTrouveEtage, *RDC*> <salon1, seTrouveEtage, *RDC*> <chambre1, seTrouveEtage, *RDC*>
3	1^{er}*Etage*, *chambre2*, *chambre3*, *sdb1*	<chambre2, seTrouveEtage, 1^{er}*Et.*> <chambre3, seTrouveEtage, 1^{er}*Et.*> <sdb1, seTrouveEtage, 1^{er}*Etage*>
4	bien1, transports	
5	*terrain2*	
6	bien1	

3.4.3 Analyse basée sur les connaissances

La dernière tâche de l'algorithme de peuplement est basée sur une analyse des connaissances de O. Elle exploite O, les TC et les raccrochabilités. Idéalement, à partir de l'instance principale, tous les individus du document devraient être atteignables. L'objectif est d'ajouter des assertions de propriété objet manquantes pour obtenir un graphe connexe, dont le point de départ serait l'instance principale. La Figure 5 (haut gauche) montre, entre autres, l'ensemble des individus (nœuds) et des assertions de propriété objet (arêtes) de l'exemple étudié. À partir de l'instance principale $bien1$, seul $transports\_en\_commun$ est atteignable. Les individus et assertions sont mis dans des lots, chaque individu étant dans exactement un lot. Chaque lot est créé à partir d'un individu i, et contient tous les individus qui sont atteignables depuis i (directement ou via une séquence d'assertions). Le premier lot créé est appelé le *lot principal*. Il est composé de l'instance principale et de tous les individus accessibles depuis elle. Ensuite, les individus des assertions restantes sont considérés. On prend celui qui a le moins de prédécesseurs ; l'ordre alphabétique des URI est choisi en cas d'égalité. Son lot est créé, et le processus continue jusqu'à ce que chaque assertion ait été traitée. Enfin, chaque individu restant est respectivement placé dans un nouveau lot. Dans l'exemple étudié, la première division en lots est représentée dans la Figure 5 en haut à gauche (lots encadrés). Le *lot principal* est composé de $bien1$ et $transports$. Ensuite, parmi les assertions restantes, celle

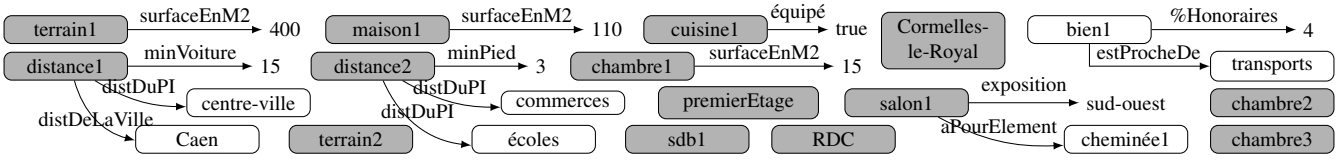

FIGURE 4 – Individus et assertions de propriété de l'exemple étudié, avant le traitement des individus sans prédécesseurs

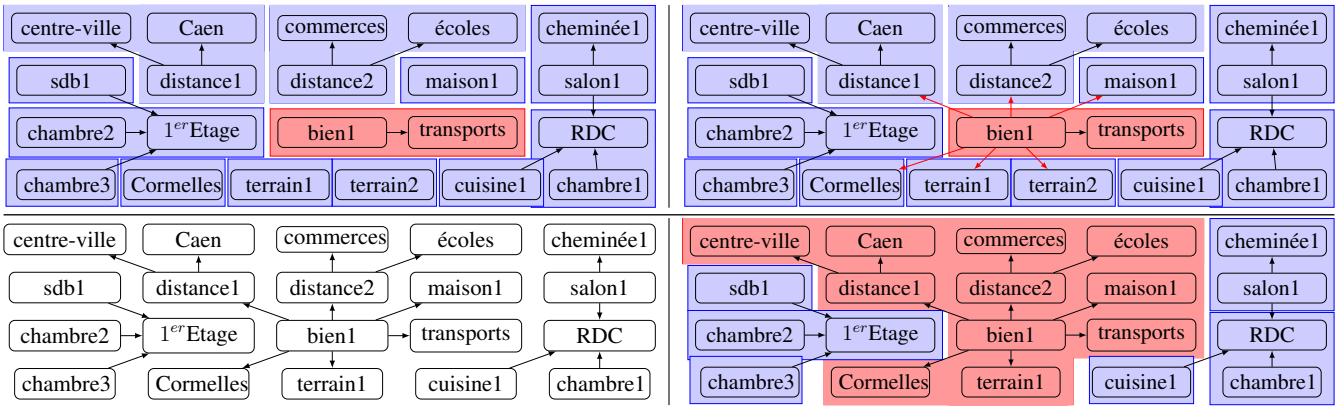

FIGURE 5 – Analyse sur les connaissances : première division en lots (en haut à gauche), premiers ajouts d'assertions (en haut à droite), fusion des individus équivalents (en bas à gauche), seconde division en lots (en bas à droite)

qui a le moins de prédécesseurs (première dans l'ordre alphabétique) est $chambre1$. Le lot suivant est donc constitué de $chambre1$ et RDC. Ainsi de suite, les lots sont formés.

L'objectif est d'accéder à tous les individus à partir de l'instance principale. Ainsi, une fois que tous les lots sont construits, on cherche à lier l'instance principale à un individu de chacun des autres lots. Afin de limiter les erreurs, le lien ne se fait qu'avec un seul individu d'un même lot. Les individus d'un lot sont triés par ordre croissant du nombre de prédécesseurs (puis par ordre alphabétique). Pour chaque lot, on tente de créer une assertion entre l'instance principale (comme sujet) et un individu du lot (comme objet) en respectant le tri, et on s'arrête dès qu'une assertion de propriété peut être établie. Dans l'exemple étudié, $distance2$ est placé en premier dans le lot $\{distance2, commerces, écoles\}$ car il n'a pas de prédécesseurs. Pour traiter ce lot, on regarde d'abord si l'instance principale $bien1$ peut être liée à $distance2$, en utilisant les raccrochabilités. C'est le cas via la propriété $seSitueADistance$. On fait ceci pour chaque lot. Cela permet d'ajouter six assertions, visibles sur la Figure 5 en haut à droite.

Ensuite, les individus équivalents sont fusionnés, et TC est mis à jour en fonction de cette fusion. Dans l'exemple d'ontologie, un bien ne peut contenir qu'un seul terrain. Pour un moteur d'inférence, $terrain1$ et $terrain2$ sont équivalents. Ils sont fusionnés dans $terrain1$ (Figure 5 en bas à gauche).

Le même processus (création de lots, ajouts d'assertions, fusion) est répété, mais en considérant comme sujets tous les individus qui sont à une distance 1 de l'instance principale, c'est-à-dire tous les individus pour lesquels il existe une assertion entre la classe principale et eux. En d'autres termes, la distance d'un individu i peut être définie comme le nombre minimal d'arêtes entre l'instance principale et i. Cette distance est incrémentée progressivement. Nous nous arrêtons soit lorsque le lot principal contient tous les individus, soit lorsque nous avons déjà tenté de relier chaque individu du lot principal. Dans l'exemple, la division en lots est ré-appliquée, conduisant aux lots de la Figure 5 en bas à droite. Le lot principal est plus grand que la première fois, et il n'y a que six autres lots. On cherche maintenant des assertions dont le sujet serait les éléments du lot principal qui sont à une distance de 1 de la classe principale ($Cormelles$, $dist1$, $dist2$, $maison1$, $terrain1$ et $transports$). L'algorithme permet d'obtenir six assertions : <$maison1$, $contient$, $sdb1$/$chambre1$/$chambre2$/$chambre3$/$cuisine1$/$salon1$>. Ainsi, la nouvelle division ne donne plus qu'un seul lot ; autrement dit, tous les éléments sont accessibles à partir de l'instance principale. Le processus est donc arrêté.

L'exemple déroulé est une illustration où KOnPoTe fonctionne bien. Cependant, l'algorithme peut générer des assertions erronées ou oublier des assertions. La section suivante détaille quelques résultats d'évaluation.

4 Expérimentations et évaluation

Cette section présente nos expérimentations sur un corpus d'annonces de vente de maisons. Le protocole expérimental et les résultats obtenus sont discutés. L'approche est implémentée en Java et utilise :

— OWL API [7] pour gérer l'ontologie ;
— Stanford NLP [14] pour découper les textes en phrases ;
— deux lemmatiseurs français [6] : celui d'Ahmet Aker [7] (noté $Aker$) et TreeTagger [8] [17] (noté TT) ;
— le moteur d'inférence Openllet reasoner [9] (Pellet).

4.1 Protocole expérimental

L'approche KOnPoTe a été testée sur un corpus extrait d'un site web [10]. Il contient 78 annonces, en français, annotées comme des ventes d'une maison à Caen. Les informations structurées sur les annonces ont été extraites au format XML, mais nous nous concentrons uniquement sur leur description textuelle. L'ontologie, construite manuellement, décrit le domaine des ventes de maisons, et respecte les contraintes mentionnées dans la Section 3.1. Elle contient initialement quelques individus génériques, tels que *double vitrage*, *transports publics*, etc., ainsi que

6. Des post-traitements ont été implémentés sur les résultats des tokens des lemmatiseurs lors de la considération des mots précédents ou suivants dans le texte. Par exemple, *10 000* devient *10000*, *3 . 4* devient *3.4*, etc.
7. http://staffwww.dcs.shef.ac.uk/people/A.Aker/activityNLPProjects.html
8. https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
9. https://github.com/Galigator/openllet
10. https://www.lecoindelimmo.com/

des entités nommées correspondant à des noms de villes ou de villages. À grande échelle, il faudrait importer une liste importante de villes, mais pour cette expérimentation, nous avons décidé de ne représenter que celles mentionnées dans le corpus. Un Gold Standard (GS) a été construit. Il s'agit de l'ontologie initiale alimentée manuellement avec des assertions représentant les descriptions du corpus. Nous avons appliqué l'approche sur l'ontologie initiale et chaque description du corpus, et nous avons cherché à comparer notre ontologie résultante avec le GS, en calculant la précision (*Pré.*), le rappel (*Rap.*) et la F-mesure (*F-m.*).

$$\textit{Pré.} = \frac{VP}{VP + FP} \quad \textit{Rap.} = \frac{VP}{VP + FN} \quad \textit{F-m.} = \frac{2 \times Pré. \times Rap.}{Pré. + Rap.}$$

Afin de calculer ces mesures, nous définissons les assertions de propriété comme des vrais positifs (VP), des faux positifs (FP) ou des faux négatifs (FN). Un VP est une assertion présente à la fois dans le GS et dans l'ontologie obtenue. Un FP est une assertion présente dans l'ontologie obtenue mais absente du GS. Un FN est une assertion présente dans le GS mais absente de l'ontologie obtenue. Pour être le plus juste possible dans nos résultats, nous avons fixé trois règles. Tout d'abord, (1) les assertions de classe ne sont pas prises en compte car elles sont souvent redondantes avec les assertions de propriété. Par exemple, avec l'assertion <$distance1$, $distanceDeLaVille$, $Caen$>, on sait via la définition du domaine de $distanceDeLaVille$ que $distance1$ appartient à la classe $Distance$. De plus, le véritable enjeu de notre problématique se situe au niveau des assertions de propriété, bien plus compliquées à obtenir, les assertions de classe étant le plus souvent issues des correspondances de classe. (2) Les assertions déduites par un moteur d'inférences sont prises en compte, sinon, la comparaison n'aurait pas de sens. Par exemple, si le GS contient l'assertion <a, $prop$, b> et que l'ontologie résultant de notre approche contient <b, $prop^{-1}$, a>, $prop^{-1}$ étant la propriété inverse de $prop$; alors nous devons réaliser que ces deux assertions signifient la même chose. (3) On ignore les propriétés qui nous conduisent à compter plusieurs fois un même élément. Par exemple, s'il existe une propriété et son inverse, un moteur d'inférences traduit une assertion de l'une en une assertion de l'autre. On se retrouve avec deux assertions équivalentes. Dans ce cas, toutes les assertions d'une des deux propriétés sont ignorées. Comme autre exemple, on peut citer le cas où une propriété n'est jamais peuplée par elle-même mais uniquement par le biais de ses sous-propriétés. Dans ce cas, nous ignorons cette propriété, de sorte qu'une assertion (juste ou fausse) ne compte pas deux fois, puisque les assertions inférées sont prises en compte.
Les fichiers expérimentaux sont disponibles [11]. KOnPoTe crée des URIs qui ne sont pas nécessairement les mêmes que ceux du GS. Pour évaluer ces cas-là, des équivalences manuelles sont définies entre les individus du GS et ceux de l'ontologie résultante représentant la même chose. De plus, l'approche peut générer des individus équivalents, sans détecter qu'ils sont équivalents. Dans ce cas, nous supposons qu'un axiome d'équivalence (owl:sameAs) est manquant et le comptons comme une assertion manquante (un FN).

11. Les fichiers expérimentaux (entrées, sorties pour toutes les approches testées, et GS) sont disponibles sur `https://doi.org/10.5281/zenodo.5776752`. Un fichier zip avec un jar exécutable pour KOnPote avec le lemmatiseur $Aker$ est disponible sur `https://alec.users.greyc.fr/research/konpote/`.

4.2 Résultats et discussion

Notre problématique étant éloignée des travaux connexes (cf. Section 2), il n'y a pas d'approches concurrentes existantes avec lesquelles se comparer. Nous évaluons KOnPoTe et analysons l'apport de ses modules principaux. La première référence que nous utilisons, nommée *Baseline*, consiste à traiter uniquement les correspondances de classe, d'individu et de propriété (les 3 premières étapes de l'analyse textuelle). Ensuite, dans *Baseline+suivant*, on ajoute le traitement des correspondances suivantes (et l'initialisation des raccrochabilités). Puis, dans *Analyse text.*, on ajoute le traitement des individus sans prédécesseurs, pour enfin ajouter l'analyse basée sur les connaissances (*KOnPoTe*).

TABLEAU 4 – Résultats de KOnPoTe et de trois baselines

Approche	Précision$_{mac}$	Rappel$_{mac}$	F-mesure$_{mac}$
KOnPoTe$_{Aker}$	**0,9516**	**0,8740**	**0,9079**
KOnPoTe$_{TT}$	0,9496	0,8681	0,9039
Analyse text.$_{Aker}$	0,8989	0,4648	0,5994
Analyse text.$_{TT}$	0,8964	0,4579	0,5929
Baseline+suiv$_{Aker}$	0,8911	0,3138	0,4440
Baseline+suiv$_{TT}$	0,8879	0,3081	0,4377
Baseline$_{Aker}$	0,9234	0,1922	0,3099
Baseline$_{TT}$	0,9230	0,1888	0,3054
Approche	Précision$_{mic}$	Rappel$_{mic}$	F-mesure$_{mic}$
KOnPoTe$_{Aker}$	**0,9465**	**0,8606**	**0,9015**
KOnPoTe$_{TT}$	0,9446	0,8545	0,8973
Analyse text.$_{Aker}$	0,8956	0,4726	0,6188
Analyse text.$_{TT}$	0,8937	0,4662	0,6127
Baseline+suiv$_{Aker}$	0,8741	0,3085	0,4561
Baseline+suiv$_{TT}$	0,8732	0,3036	0,4505
Baseline$_{Aker}$	0,9135	0,1926	0,3182
Baseline$_{TT}$	0,9138	0,1892	0,3135

Le Tableau 4 montre les résultats. Les approches sont testées avec les deux lemmatiseurs ($Aker$ et TT). Chaque métrique est calculée de manière macroscopique (mac) et microscopique (mic). Le macro-calcul est la moyenne des métriques pour chaque annonce (chaque annonce a le même poids), tandis que le micro-calcul considère la somme de tous les VP, FP, FN (chaque assertion a le même poids).
Le lemmatiseur *Aker* est plus performant que *TreeTagger*, mais la différence est relativement faible. Les modules ajoutés ont une bonne contribution, puisque chacun permet un gain relativement élevé de F-mesure. Tout d'abord, la baseline, composée des traitements de base, donne un score relativement élevé en précision ($> 0,9$) mais un faible score en rappel ($< 0,2$). Ainsi, la plupart des assertions sont correctes, mais beaucoup d'assertions sont manquantes. L'ajout des modules permet d'ajouter des assertions, qui ont plutôt tendance à être correctes. En effet, au fur et à mesure que les modules sont ajoutés, le rappel augmente sans trop de perte en précision. Plus précisément, le traitement des correspondances suivantes ajoute un peu de

bruit (petite perte de précision) mais augmente le rappel de moitié (de $\sim 0,2$ à $\sim 0,3$). Le traitement des individus sans prédécesseurs augmente également le rappel de moitié (de $\sim 0,3$ à $\sim 0,45$), sans diminuer la précision (en l'augmentant légèrement). Enfin, l'analyse basée sur les connaissances apporte une contribution considérable. La précision et le rappel augmentent, ce qui entraîne une augmentation de moitié de la F-mesure (de $\sim 0,6$ à $\sim 0,9$). Ces trois modules sont donc essentiels : ils insèrent beaucoup d'assertions manquantes sans ajouter trop d'assertions erronées.

5 Conclusion et perspectives

Cet article présente KOnPoTe, une approche générique, entièrement automatique, pour peupler une ontologie de domaine, à partir de descriptions textuelles d'objets de ce domaine. KOnPoTe est une chaîne de traitements, dont les résultats sont prometteurs sur une première expérimentation. Son algorithme est basé uniquement sur le contexte du problème (descriptions textuelles d'objets) et non sur des règles linguistiques propres à un domaine.

Dans un travail futur, nous expérimenterons l'approche sur de nouveaux domaines : d'autres types d'annonces (comme les ventes de bateaux) et des descriptions diverses (hôtels, restaurants, incidents, etc.). Bien sûr, cela est chronophage : constitution du corpus, de l'ontologie, du gold standard, ainsi que des équivalences potentielles entre le gold standard et l'ontologie en sortie de KOnPoTe, et des liens d'équivalence potentiellement manquants dans la sortie. Une autre idée est de tester KOnPoTe sur le même domaine et corpus mais avec des ontologies différentes (même terminologie mais choix de représentation différents). Une analyse approfondie d'une telle expérience pourrait conduire à un ensemble de conseils à suivre pour représenter l'ontologie d'entrée. Enfin, notre objectif final est d'utiliser KOnPoTe comme une première étape pour traiter le problème des annotations erronées mentionné dans la Section 1.

Remerciements

Nous remercions Q. Leroy et J.-P. Kotowicz pour leur participation dans l'élaboration de l'ontologie initiale, ainsi qu'E.-A. Carré et M. Gueret pour la constitution du corpus.

Références

[1] Harith Alani et al. Automatic ontology-based knowledge extraction and tailored biography generation from the web. *IEEE Intell Syst*, pages 14–21, 2003.

[2] Ali Ayadi, Ahmed Samet, François de Bertrand de Beuvron, and Cecilia Zanni-Merk. Ontology population with deep learning-based NLP : a case study on the Biomolecular Network Ontology. *Procedia Computer Science*, 159 :572–581, 2019.

[3] Silvana Castano et al. Multimedia Interpretation for Dynamic Ontology Evolution. *Journal of Logic and Computation*, 19(5) :859–897, 09 2008.

[4] Yohann Chasseray, Anne-Marie Barthe-Delanoë, Stéphane Négny, and Jean-Marc Le Lann. A Generic Metamodel for Data Extraction and Generic Ontology Population. *J. Inf. Sci.*, 48(6) :838—856, dec 2022.

[5] Carla Faria, Ivo Serra, and Rosario Girardi. A domain-independent process for automatic ontology population from text. *Science of Computer Programming*, 95 :26 – 43, 2014.

[6] Houssem Gasmi, Jannik Laval, and Abdelaziz Bouras. Cold-start cybersecurity ontology population using information extraction with LSTM. In *CSET'2019*, pages 1–6, Doha, Qatar, October 2019.

[7] Matthew Horridge and Sean Bechhofer. The OWL API : A Java API for Working with OWL 2 Ontologies. In *OWLED*, page 49–58, Aachen, DEU, 2009.

[8] Ian Horrocks et al. SWRL : A Semantic Web Rule Language Combining OWL and RuleML. Technical report, World Wide Web Consortium, 2004.

[9] Vindula Jayawardana et al. Semi-supervised instance population of an ontology using word vector embedding. In *ICTer*. IEEE, sep 2017.

[10] Natthawut Kertkeidkachorn and Ryutaro Ichise. An Automatic Knowledge Graph Creation Framework from Natural Language Text. *IEICE Transactions on Information and Systems*, E101.D(1) :90–98, 2018.

[11] Andreas Korger and Joachim Baumeister. Rule-based Semantic Relation Extraction in Regulatory Documents. In *LWDA*, volume 2993 of *CEUR Workshop Proceedings*, pages 26–37, September 2021.

[12] Mohamed Lubani, Shahrul Azman Mohd. Noah, and Rohana Mahmud. Ontology population : Approaches and design aspects. *Journal of Information Science*, 45 :502 – 515, 2019.

[13] Jawad Makki, Anne-Marie Alquier, and Violaine Prince. Ontology Population via NLP Techniques in Risk Management. *International Journal of Humanities and Social Sciences*, pages 212–217, 2009.

[14] Christopher D. Manning et al. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL System Demonstrations*, pages 55–60, 2014.

[15] Sergio Oramas, Mohamed Sordo, and Luis Espinosa-Anke. A Rule-Based Approach to Extracting Relations from Music Tidbits. In *Int. Conf. on World Wide Web*, pages 661–666, Florence, Italy, 2015.

[16] José Alejandro Reyes-Ortiz. Criminal Event Ontology Population and Enrichment using Patterns Recognition from Text. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(11), 2019.

[17] Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees, 1994.

[18] Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer, 2009.

[19] Fabian Suchanek, Georgiana Ifrim, and Gerhard Weikum. LEILA: Learning to Extract Information by Linguistic Analysis. In *Workshop on Ontology Learning and Population*, pages 18–25, Sydney, Aust., 2006.