Table des matières

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Liste des figures

Liste des tableaux

Report-Gratuit.com

Chapitre 1

Introduction

De nos jours, un grand nombre d'informations est disponible sur internet. Un internaute est, en théorie, capable d'y trouver tout ce qu'il recherche. Cependant, en pratique, cette tâche reste complexe car l'internaute est confronté à un trop grand volume de données.

La recherche d'un internaute peut consister à obtenir un renseignement sur une personne, un évènement, etc., ou peut porter sur un produit satisfaisant un certain besoin. Dans ce cas, il existe des outils automatiques pouvant l'aider à trouver ce produit. C'est le cas des systèmes de recommandation. Le travail présenté dans cette thèse s'inscrit dans le cadre de ces systèmes.

Les systèmes de recommandation [Ricci *et al.* 2011] étudient le comportement des internautes (leurs recherches de produits) et leur font des suggestions de produits susceptibles de les intéresser. Ces suggestions sont personnalisées. Par exemple, on suggérera une liste de films d'action à un internaute ayant récemment acheté un film d'action. Le principal but des systèmes de recommandation est d'aider les utilisateurs à trouver ce qui les intéresse, même en présence de grands volumes d'informations. Pour cela, les systèmes de recommandation sélectionnent les sous-ensembles d'objets qui correspondent le plus aux profils des utilisateurs déterminés à partir de leurs préférences.

Il existe plusieurs types d'approches mis en œuvre au sein des systèmes de recommandation. Nous en distinguerons trois : le filtrage collaboratif (collaborative filtering), les approches basées sur le contenu (content-based) et les approches conversationnelles. Les deux premiers types d'approches sont les plus connus. Il s'agit d'approches qui se basent sur l'expression des opinions des utilisateurs, c'est-à-dire sur leur évaluation préalable (ratings) de produits de même type que celui recherché. Cette évaluation peut prendre différentes formes, par exemple numérique (nombre d'étoiles) ou encore binaire (j'aime/je n'aime pas). Le dernier type d'approche nécessite une interaction entre le système et l'utilisateur.

Le filtrage collaboratif est un processus consistant à évaluer les produits en utilisant les opinions (ratings) des personnes proches d'une personne donnée [Schafer *et al.* 2007].

Ce type d'approche repose sur l'hypothèse que des utilisateurs avec des préférences similaires (ayant donné des évaluations similaires) dans le passé auront des préférences similaires dans le futur. Par exemple, si on sait qu'un certain nombre d'utilisateurs aiment les films A, B, C mais pas le film D et qu'on souhaite conseiller un film à un utilisateur qui aime les films A et B, il vaut mieux lui conseiller le film C plutôt que le film D.

Les approches basées sur le contenu recommandent un produit à un utilisateur en se basant sur le profil de celui-ci et sur la description du produit recommandé [Pazzani & Billsus 2007]. Ce type d'approche repose sur une mise en correspondance entre le profil d'un utilisateur (les évaluations qu'il a faites) et les caractéristiques du produit qui lui sera recommandé. Une description des produits doit donc être disponible. Ainsi, si un utilisateur aime le film Argo : "Ben Affleck, Drame, Thriller, ..." et le film Pulp Fiction : "John Travolta, Drame, Thriller, ...", il semble intéressant de lui proposer d'autres films ayant les genres "Drame" et "Thriller". La plupart de ces approches mettent en œuvre des techniques d'apprentissage supervisé.

Le troisième type d'approche de recommandation, nommée approche conversationnelle ou encore basée sur les connaissances (knowledge-based), est moins connue [Burke 2000]. Ce type d'approche ne dépend pas d'une base d'évaluations (ratings) données par les utilisateurs mais des demandes qu'ils ont explicitement formulées. L'utilisateur interagit avec le système pour préciser ses désirs. En combinant les connaissances sur les désirs de l'utilisateur et celles du domaine, le système est capable de trouver quels produits doivent lui être proposés et dans quel ordre. Cette thèse se situe dans le cadre de ce type de système.

Cette thèse a été motivée par un partenariat entre le Laboratoire de Recherche en Informatique (LRI) et la startup Wepingo. Wepingo développe un moteur de recommandation dit affinitaire, c'est-à-dire capable de calculer une "affinité" entre ce que recherche un utilisateur et des produits répertoriés dans des catalogues fournisseurs. Cela permet d'associer automatiquement des produits à une recherche d'un utilisateur, afin de les lui proposer. La **recherche d'un utilisateur** est composée d'un ensemble de **besoins utilisateur**. Wepingo propose aux internautes de remplir des questionnaires de façon à identifier leurs besoins. Il a, par ailleurs, été établi des mises en correspondance entre des produits issus de catalogues fournisseurs et chaque besoin pouvant être formulé via les questionnaires. Notons que ce que nous appelons "**besoin utilisateur**" peut correspondre à une réponse à une question du questionnaire, ou bien à une réponse à un ensemble de questions. Par exemple, nous avons travaillé sur un questionnaire pour proposer des destinations de vacances. Le besoin utilisateur "Destination avec sports aquatiques praticables en hiver" provient d'une combinaison des réponses possibles aux questions "Quand voulez-vous partir ?" et "Quels types d'activités voulez-vous pratiquer ?".

Ainsi, le processus de recommandation auquel nous nous intéressons repose sur l'existence de mises en correspondance entre des produits issus de catalogues

fournisseurs et des besoins utilisateur. Jusqu'à présent, les produits traités par Wepingo (téléphones, électro-ménager, etc.) sont décrits d'une manière structurée (dans des tables/tableaux) dans les catalogues. De ce fait, les correspondances peuvent être facilement trouvées en exploitant la structure des descriptions des produits. Par exemple, le besoin "très bonne qualité image et vidéo" exprimable via le questionnaire pour suggérer un téléphone, est associé à des critères (résolution, nombre de pixels, etc.) disponibles dans les descriptions des téléphones des catalogues. Cependant, il existe des types de produits (destinations, films, etc.) pour lesquels les descriptions dans les catalogues ne sont pas structurées : la description de chaque produit est un texte. Dans ce cas, un processus capable d'identifier les mises en correspondance malgré l'absence de structure dans les descriptions de produit doit être élaboré. Par ailleurs, les catalogues de produits évoluent et de nouveaux catalogues apparaissent au fil du temps. Un processus automatique d'identification de ces mises en correspondance serait alors d'une grande aide. La collaboration établie avec la société Wepingo porte sur la réalisation d'un tel processus. En d'autres termes, il s'agit de proposer une solution d'étiquetage de documents décrivant des produits avec des annotations traduisant les besoins utilisateur. Deux aspects importants sont à prendre en compte dans la solution attendue : (1) l'automatisation du processus et (2) l'adaptabilité par rapport au domaine, car la solution doit convenir pour différents types de produits (destinations de vacances, films, musique, etc.).

Réaliser cet étiquetage d'une manière automatique implique de comprendre le sens des descriptions textuelles des produits des catalogues. Les besoins utilisateur sont en général des concepts très précis correspondant à leurs propres points de vue (par exemple "destination avec sports aquatiques praticables en hiver", "ordinateur portable avec une bonne portabilité") qui ne sont jamais mentionnés en tant que tel dans les descriptions de produits. Comprendre le sens du contenu de documents du Web et être capable de l'exploiter constitue aujourd'hui un des challenges du Web sémantique. Les ontologies [Charlet *et al.* 2004], qui sont des représentations formelles de conceptualisations [Gruber 1993], décrivent les concepts d'un domaine particulier et peuvent aider à comprendre le sens de contenus. Nous proposons dans cette thèse une approche à base d'ontologie, pour comprendre le contenu des documents du corpus traité, définir explicitement les concepts correspondant aux besoins utilisateur et raisonner sur l'ensemble de ces informations. L'ontologie est donc un outil tout à fait adapté à l'approche proposée, de par sa capacité d'intégration et de raisonnement.

Nos contributions sont les suivantes :

- une approche automatisée d'étiquetage de documents décrivant chacun une entité d'un domaine donné, applicable quel que soit le domaine.

    1. Cette approche extrait des informations des documents, les complète avec des données issues d'autres ressources (ressources du Linked Open Data) et intègre le tout dans une ontologie du domaine concerné.

2. Des définitions de concepts (correspondant à des besoins), compréhensibles par des humains et interprétables par des machines, sont apprises automatiquement et intégrées à l'ontologie.
3. Un processus de raisonnement applique les définitions apprises. Ainsi, si un document se conforme à la définition d'un concept, il sera étiqueté avec ce concept.

- un modèle d'acquisition de données issues du LOD.

  Les données extraites des documents décrivant les produits dans les catalogues fournisseur sont incomplètes. Elles doivent être complétées en utilisant des ressources externes. Le modèle d'acquisition proposé permet de spécifier comment peupler une ontologie cible avec les données d'une ressource externe, ici le LOD, à laquelle correspond une autre ontologie dite source. Le modèle se décompose en deux sous-modèles : (1) un modèle de correspondances qui permet de définir des correspondances, potentiellement complexes, entre les propriétés de deux ontologies différentes. En effet, dans certains cas, pour une propriété ontologique donnée, il peut y avoir une multitude de propriétés équivalentes dans l'autre ontologie, et qui peuvent être exprimées dans des unités de mesure différentes, voire des cas d'union plus complexe où une propriété équivalente peut être calculable sans être explicitement représentée. Il faut la calculer à partir d'autres valeurs de propriétés qui, elles, sont représentées ; (2) un modèle de spécification de chemins d'accès à des propriétés qui permet de définir des chemins alternatifs pour pallier le problème d'incomplétude des données dans le LOD.

- un processus de génération automatique de requêtes et d'intégration des données obtenues dans l'ontologie.

  Les données externes sont intégrées dans l'ontologie grâce à des requêtes SPARQL générées automatiquement. Le processus de génération de ces requêtes est basé sur le modèle d'acquisition des données du LOD. Les requêtes générées intègrent directement les données du Web des données à l'ontologie utilisée dans notre approche.

- des expérimentations et une validation de l'approche proposée.

  L'approche a été implémentée et des expérimentations ont été réalisées. Ces expérimentations évaluent et valident l'approche que nous proposons. Elles mettent en exergue l'importance de l'ontologie et des données du Web des données.

Plan du manuscrit

Nous décrivons ci-dessous les différents chapitres de la thèse.

Le **chapitre 2** décrit le problème que nous avons cherché à résoudre dans la thèse : l'établissement d'une approche aussi automatique que possible pour étiqueter des documents d'un corpus, avec des concepts associés à des besoins utilisateur préalablement répertoriés.

Une première partie est consacrée à la résolution du problème d'étiquetage des documents. Elle se compose de trois chapitres.

Le **chapitre 3** traite des travaux liés à la problématique d'étiquetage des documents. Deux catégories de travaux sont considérées. La première concerne les travaux d'annotation sémantique de documents. La seconde concerne les travaux de peuplement et d'enrichissement d'ontologie.

Le **chapitre 4** décrit l'approche SAUPODOC que nous proposons pour répondre à la problématique d'étiquetage des documents. Cette approche est automatique moyennant un certain nombre d'entrées que le concepteur du système doit fournir. Elle consiste d'abord à peupler l'ontologie avec des assertions de propriété venant des connaissances exprimées dans les documents du corpus et dans le LOD. Par la suite, l'ontologie est enrichie par les concepts utiles pour l'étiquetage, et leur définition apprise grâce à des exemples de documents annotés. Grâce à ces définitions, l'étiquetage de descriptions de produits du même domaine mais issus de nouveaux catalogues fournisseur est possible. Cette approche est dirigée par une ontologie qui joue un rôle de pivot et permet à tout un ensemble de tâches de coopérer.

Le **chapitre 5** détaille les expérimentations effectuées pour valider l'approche SAUPODOC. Nous nous comparons à deux outils standard d'apprentissage supervisé sur deux domaines ayant des caractéristiques différentes (corpus petit/grand, ontologie riche/pauvre). Pour ce faire, nous utilisons des mesures classiques. Nous discutons de l'apport de certains éléments de notre approche, telles que la structure de l'ontologie, la tâche de complétion des données textuelles avec celles du LOD et la présence de définitions explicites des concepts considérés.

Une seconde partie est consacrée à la résolution du problème d'acquisition des données du Web des données. Elle se compose de trois chapitres.

Le **chapitre 6** décrit les travaux liés à l'acquisition de données du Web des données. Il définit le Web des données et dresse un état de l'art des travaux traitant des problèmes d'incomplétude, d'hétérogénéité sémantique et des difficultés d'accès à des données.

Le travail présenté dans le **chapitre 7** traite de l'acquisition de données du LOD dans le but de peupler une ontologie avec des assertions de propriété. Pour cela, il nous faut tenir compte de l'existence, dans le LOD, de propriétés multiples, équivalentes, multi-valuées, de propriétés absentes mais pouvant toutefois être calculées et de

propriétés sans valeurs. Les correspondances à établir étant complexes, nous proposons un modèle pour les spécifier. Nous définissons également un modèle pour spécifier des chemins d'accès alternatifs à des propriétés en cas de valeurs manquantes. Enfin, les valeurs de propriétés auxquelles le LOD donne accès après application des modèles de correspondance et d'accès, peuvent nécessiter des traitements complémentaires de façon à satisfaire les contraintes de l'ontologie à peupler. Nous proposons des mécanismes adaptés.

Le **chapitre 8** montre comment les différents modèles proposés dans le chapitre 7 sont utilisés pour générer automatiquement des requêtes SPARQL et ainsi faciliter l'interrogation.

Le **chapitre 9** conclut et énonce des perspectives futures de travail.

Chapitre 2

Cadre de travail et problématique

Sommaire

Ce travail de thèse a fait l'objet d'une collaboration entre le LRI et la startup Wepingo débutée en avril 2013 et officialisée par une convention de collaboration signée en août 2014 d'une durée de deux ans. L'objet de cette collaboration est l'étiquetage automatisé de documents. Ce chapitre présente le cadre de ce travail et la problématique.

2.1 L'application de recommandation affinitaire proposée par Wepingo

Wepingo est une jeune entreprise dont l'objectif est de recommander des produits aux internautes. Elle souhaite étendre les types de produits actuellement traités. La première section décrit la recommandation affinitaire effectuée par Wepingo. La seconde section relate un premier travail de peuplement automatisé d'ontologie. La dernière section est une description fonctionnelle du problème à résoudre dans le cadre de cette thèse.

2.1.1 La recommandation affinitaire

Wepingo développe un moteur de recommandation affinitaire, c'est-à-dire capable de calculer une affinité entre l'ensemble des critères de recherche d'un utilisateur et des produits provenant de catalogues fournisseurs. Son but est de proposer, à un utilisateur, les produits qui correspondent le plus à ses besoins.

Pour cela, Wepingo a mis en place un site web[1] où un utilisateur peut obtenir une pré-sélection d'un ensemble de produits satisfaisant sa recherche. Pour ce faire, l'utilisateur doit répondre sur le site à un questionnaire portant sur le type de produit qui l'intéresse (appareil photo, lave-linge, téléphone, ou autre). Ce questionnaire permet à l'utilisateur d'exprimer des préférences. Ainsi, Wepingo peut cerner ses besoins. Par exemple, l'entreprise a répertorié les besoins utilisateurs relatifs au domaine des téléphones. La Figure 2.1 montre un extrait du questionnaire destiné aux utilisateurs. Chaque réponse à une question permet de cerner une préférence. Un besoin peut être exprimé au travers d'une ou de plusieurs préférences. Par exemple, la préférence "je regarde **souvent** des vidéos ou des photos sur téléphone" (dernière réponse de la figure 2.1) correspond au besoin "téléphone avec une très bonne qualité image et vidéo". En revanche, dans le domaine des destinations de vacances, l'ensemble des préférences "je pars en hiver" et "je veux faire des sports aquatiques" correspond au besoin "destination avec sports aquatiques praticables en hiver". L'identification d'un besoin permet ensuite de proposer des produits qui peuvent le satisfaire. Les propositions sont rangées par ordre décroissant de satisfaction. Sur la Figure 2.2, les téléphones Samsung Galaxy sont supposés satisfaire à 92% la recherche exprimée par l'utilisateur.

Fig. 2.1 Quiz proposé par Wepingo permettant aux utilisateurs d'exprimer leurs préférences en matière de téléphone

Pour pouvoir faire ces propositions, Wepingo a dû étiqueter les produits des catalogues avec les besoins utilisateurs répertoriés. Suivant le domaine et le type de

[1] www.wepingo.com

Fig. 2.2 Exemples de téléphones proposés en fonction de la recherche de l'utilisateur

description dont dispose Wepingo, cet étiquetage peut être automatique, basé sur la structure de la description d'un produit. Par exemple, si les descriptions des produits sont des tableaux d'attributs, il est facile d'extraire les caractéristiques des produits. Il faut cependant parfois y inclure des règles du domaine, pour savoir par exemple, quelles plages de valeurs et quels attributs font qu'un téléphone a une "très bonne qualité image et vidéo".

Wepingo souhaite aujourd'hui pouvoir traiter des catégories de produits de plus en plus variées. L'entreprise souhaite s'ouvrir à de nouveaux domaines, tels que la recommandation de destinations de vacances, de films, de musique, etc. Les descriptions de ces types de produits ne sont, en général, pas du même type que les descriptions des types de produits déjà traités. Dans de nombreux cas, il s'agit de produits avec des descriptions textuelles non structurées. Il est donc beaucoup plus difficile d'établir l'adéquation entre un produit et un besoin d'une manière automatique. Cela nécessite une compréhension du texte correspondant aux descriptions.

2.1.2 Un travail automatisé de peuplement d'ontologie

Un premier travail a été réalisé [Alec *et al.* 2014a, Alec *et al.* 2014b, Alec *et al.* 2016f] entre avril et septembre 2013. Il avait pour objectif d'associer automatiquement des descriptions textuelles de jouets à leurs catégories et caractéristiques correspondant à des concepts d'une ontologie. Les concepts (catégories et caractéristiques de jouets) faisaient par ailleurs l'objet d'un alignement manuel avec des besoins utilisateurs. Ainsi, tout besoin exprimé via un questionnaire pouvait être lié à une catégorie ou caractéristique de jouet et les jouets de cette catégorie ou caractéristique pouvaient être proposés. Le problème sur lequel nous avons travaillé portait sur l'identification automatisée des catégories et des caractéristiques des jouets des catalogues fournisseur, ce qui correspondait à un problème du peuplement de l'ontologie. Cette tâche n'est pas évidente, même pour un expert du domaine, car les catégories

et caractéristiques ne sont pas citées explicitement dans les textes. Par exemple, les caractéristiques "dextérité", "concentration" ou encore "perception sonore" sont des aptitudes développées par certains jouets mais leur description ne les mentionne pas.

Nous avons conçu un outil permettant d'aider un expert du domaine à étiqueter les descriptions de jouet. À partir de la description d'un jouet donné, l'outil propose à l'expert les catégories et caractéristiques les plus probables, et en suggère d'autres un peu moins probables. L'expert n'a plus qu'à valider ou invalider les propositions par le biais d'une interface. Pour proposer et suggérer ces catégories et caractéristiques, l'outil se base sur des règles du domaine introduites manuellement dans l'ontologie. Ces règles représentent le fait que certaines catégories ou caractéristiques de jouet peuvent, soit systématiquement, soit généralement, impliquer d'autres catégories ou caractéristiques. Par exemple, un jeu dit "de mise en scène" (playmobil, Barbie, etc.) développe systématiquement certaines aptitudes chez un enfant (créativité expressive, reproduction de rôles, etc.). De ce fait, si l'expert du domaine valide un étiquetage en tant que "jeu de mise en scène", les caractéristiques concernant ces aptitudes lui sont aussi automatiquement affectées. Grâce à cet outil, l'expert est aidé pour étiqueter un sous-ensemble de documents. Ce sous-ensemble de documents étiqueté est utilisé par la suite comme ensemble d'exemples par une technique d'apprentissage automatique supervisée, pour étiqueter de nouvelles descriptions de jouet.

Ce premier travail présente certains atouts, notamment le fait de diminuer le travail de l'expert. Cependant, l'approche proposée n'est pas suffisamment automatique au sens où il s'agit d'un processus interactif (coopératif) entre l'outil et l'expert du domaine. De plus, l'expert doit expliciter des règles du domaine, pas toujours évidentes à extraire, à partir des définitions des concepts de l'ontologie telles qu'elles ont été données dans le livre de références [Garon *et al.* 2002], à partir duquel l'ontologie a été construite. Enfin, dans ce travail Wepingo a dû aligner manuellement les concepts de l'ontologie avec les besoins utilisateurs préalablement répertoriés.

2.1.3 Description fonctionnelle du problème à résoudre

Dans le cadre de cette thèse, l'objectif est de proposer un étiquetage automatisé de descriptions textuelles de produits (non structurées) d'un domaine donné avec des concepts qui correspondent à des besoins utilisateurs déjà répertoriés. L'approche proposée doit être aussi automatique que possible pour facilement prendre en compte de nouveaux produits de catalogues traités mais également de nouveaux catalogues du même domaine. Un processus totalement automatique permettrait d'effectuer l'étiquetage souhaité dès qu'un nouveau catalogue fournisseur apparaît. En plus de cette indépendance au corpus, l'approche doit être générique, c'est-à-dire qu'elle doit fonctionner pour divers domaines. Arriver à prendre en compte des catalogues de produits de nouveaux domaines permettrait à Wepingo d'ajouter très facilement de nouveaux types de produits (destinations de vacances, films, livres, etc.) sur son site Web. Il serait aussi intéressant que l'approche ne fonctionne pas uniquement sur des

produits mais sur des entités en général (produit, objet, personne, etc.). De plus, le processus proposé ne doit pas être opaque. Si aucun produit ne correspond à un besoin utilisateur, Wepingo doit être en mesure de proposer des produits satisfaisant partiellement ce besoin. Cela signifie que la solution proposée doit être élaborée en raisonnant sur les connaissances disponibles.

Les paragraphes suivants décrivent les entrées qui nous sont fournies par Wepingo et la sortie attendue.

Les entrées

- Les documents à étiqueter sont donnés sous la forme d'un corpus. Ces documents sont des documents XML très peu structurés, cf. Figure 2.3. Ils contiennent une balise évoquant le nom de l'entité décrite, et une description textuelle décrivant les principales caractéristiques de l'entité décrite.

```
<name>nom de l'entité</name>
<description>
description textuelle décrivant
les principales caractéristiques
de l'entité
</description>
```

Fig. 2.3 Structure des documents XML

- La liste des besoins utilisateurs est donnée. D'une façon générale, nous appelons ces besoins les **concepts cibles**. Il s'agit d'expressions qui ne sont pas explicitement mentionnées dans les descriptions textuelles. Ce sont en général des expressions très précises correspondant à des vues utilisateur (par exemple "Destination où l'on peut pratiquer des sports aquatiques pendant l'hiver" dans le domaine des destinations). Ces concepts sont nommés mais ne sont pas définis.

Les sorties

La Figure 2.4 est une illustration des sorties du système à concevoir. Chaque entité décrite dans un document du corpus doit être étiquetée avec tous les concepts cibles. Si une entité correspond à un concept cible donné, on parlera d'**annotation positive** (flèche verte avec ✔), sinon, on parlera d'**annotation négative** (flèche rouge avec ✖).

Le processus à concevoir est illustré par la Figure 2.5.

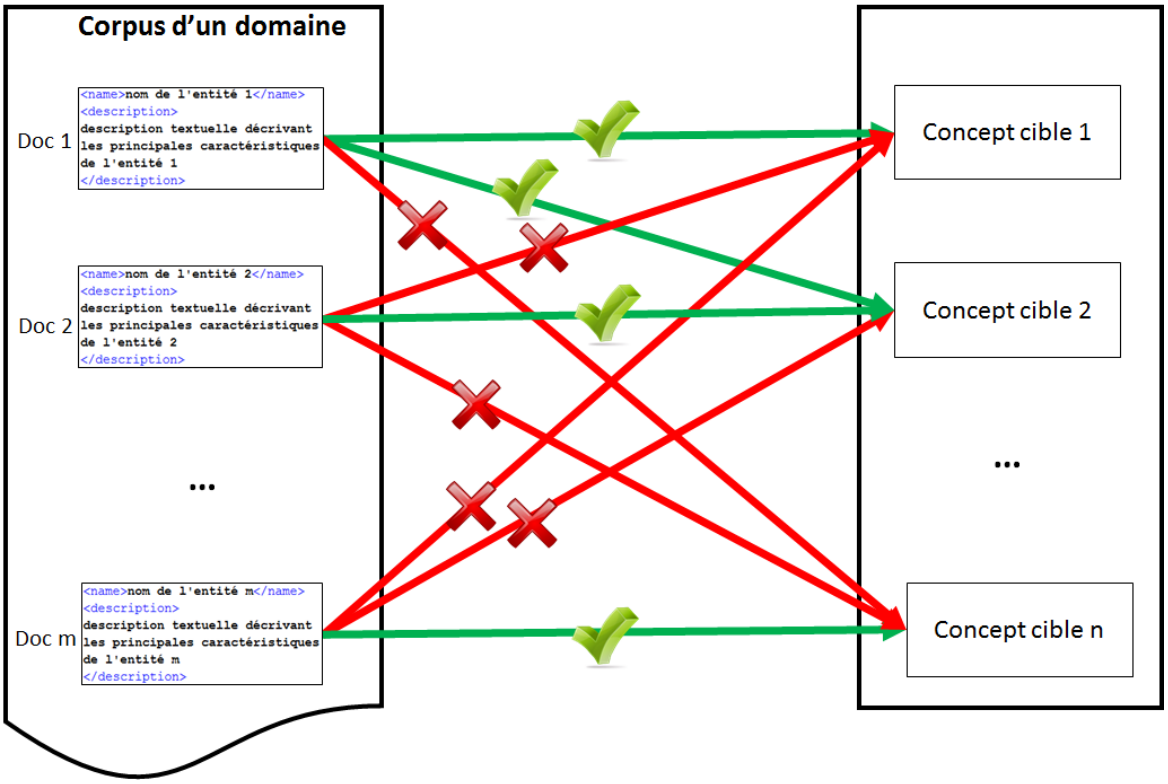


Fig. 2.4 Illustration des sorties du système d'étiquetage à concevoir

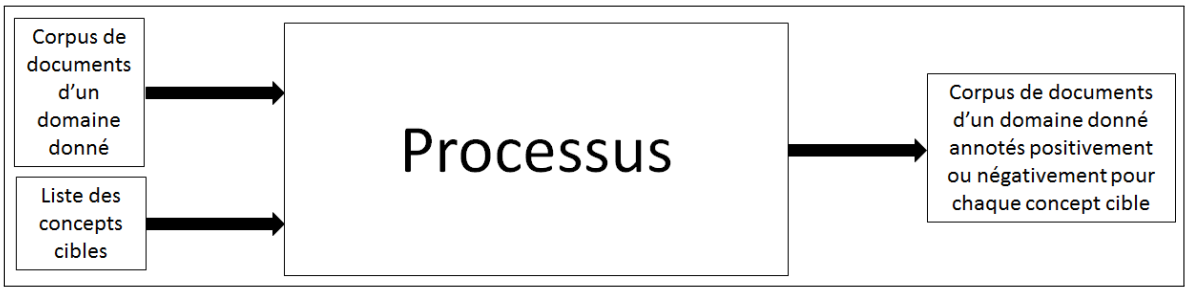


Fig. 2.5 Processus à concevoir

2.2 Problématique

Cette section décrit le problème à résoudre d'une façon plus approfondie et justifie l'intérêt d'une approche à base d'ontologie.

2.2.1 Analyse approfondie du problème

Les descriptions textuelles des entités ne mentionnent pas les concepts cibles qui correspondent à des besoins utilisateurs très spécifiques. Par exemple, dans le domaine des destinations de vacances, le concept cible "Destination où l'on peut pratiquer des

sports aquatiques pendant l'hiver" est un concept spécifique, au sens où il est propre à un point de vue particulier, celui d'un besoin utilisateur. Ce concept est trop précis pour apparaître directement dans les textes. En revanche, les descriptions contiennent des termes dénotant des caractéristiques importantes des produits décrits qui sont aussi des caractéristiques de besoins utilisateurs. Par exemple, on peut trouver des termes mentionnant des sports aquatiques dans les textes ("diving", "snorkeling", "watersports", etc.) car ceux-ci décrivent les principales caractéristiques des destinations et incluent donc les activités qu'on peut y faire. Ces termes ne seront exploitables que si on connaît par ailleurs leurs liens avec les concepts cibles. Il nous faut donc expliciter les définitions des concepts cibles en fonction des caractéristiques des produits.

De plus, éviter l'opacité de la solution suppose qu'un expert du domaine soit capable de comprendre les raisons des étiquetages obtenus. De ce fait, une définition précise des concepts cibles permettrait de connaître quels éléments ont permis un étiquetage positif (respectivement négatif). Ainsi, si une définition ne permet pas d'obtenir la moindre annotation positive, alors l'expert peut étudier comment la modifier (relâcher des contraintes) pour en obtenir. Cela éviterait que certains besoins utilisateurs ne soient associés à aucune description de produits. Cependant, une définition de concept cible peut être assez difficile à formuler par un expert du domaine. Par exemple, intuitivement une "Destination où l'on peut pratiquer des sports aquatiques pendant l'hiver" est une destination avec des sports aquatiques, telle que ceux-ci y sont praticables en hiver. Cela signifie qu'il doit faire suffisamment beau en hiver pour les pratiquer. Toutefois, arriver à définir quantitativement "suffisamment beau en hiver pour les pratiquer" n'est pas une chose aisée : Quelle température ? Quel taux de pluie ?

Par ailleurs, les données des documents sont insuffisantes pour décider d'étiqueter ou non une description avec des concepts cibles. Par exemple, pour "Destination où l'on peut pratiquer des sports aquatiques pendant l'hiver", on aura besoin de connaître la présence ou l'absence de sports aquatiques et également les données météorologiques. Or, les documents dont on dispose mentionnent des sports aquatiques mais ne contiennent aucune information météorologique. En somme, les documents sont complets vis-à-vis de certaines propriétés (par exemple, les sports aquatiques praticables) mais n'ont aucune information vis-à-vis d'autres propriétés (par exemple, la météo). Cela pose problème pour une approche totalement automatique. En effet, si un expert du domaine pouvait interagir au cours de l'approche, il pourrait éventuellement fournir des connaissances du domaine, telles que la météo dans le cas des destinations. Mais comme l'approche doit être automatique, nous devons trouver une autre solution.

De ce fait, nous en déduisons que nous aurons besoin à la fois d'informations supplémentaires pour compléter les documents, et de définitions des concepts cibles. Les informations supplémentaires pourront être recherchées dans des ressources externes. Une fois les définitions connues, un raisonnement pourra être réalisé pour savoir quelles

entités sont instances de quels concepts cibles. Pour raisonner, les connaissances devront être représentées de façon formelle. De ce fait, l'utilisation d'une ontologie semble particulièrement adaptée. En effet, une ontologie est une représentation formelle des concepts d'un domaine [Gruber 1993]. On peut y représenter l'ensemble des connaissances extraites et celles acquises à partir de ressources externes pour raisonner ensuite dessus.

2.2.2 Une approche basée sur une ontologie

Cette section introduit des pré-requis sur les ontologies et reformule le problème en un problème d'annotation sémantique de documents, de peuplement d'ontologie et d'enrichissement d'ontologie.

Pré-requis sur les ontologies

Une ontologie décrit les concepts d'un domaine d'application. Selon [Gruber 1993], l'ontologie est une spécification explicite d'une conceptualisation.

Une ontologie est constituée de différentes entités :

- des concepts,
- des propriétés,
- des axiomes,
- éventuellement des individus.

Une ontologie contenant des individus est une ontologie peuplée. On parle aussi dans ce cas de base de connaissances [Baader *et al.* 2003, Corman *et al.* 2015].

Les concepts sont des abstractions d'ensembles d'objets composant un domaine. Par exemple, les concepts "Cours" et "Enseignant" font partie d'une ontologie qui décrit le monde de l'enseignement.

Les propriétés permettent de caractériser les objets des concepts. On distingue deux catégories de propriétés :

- les propriétés correspondant à des relations entre objets (propriétés objets),
- les propriétés correspondant à des caractéristiques propres des objets ou des concepts (propriétés datatype et propriétés d'annotation).

Les propriétés décrivant des relations entre objets relient un concept à un autre concept tandis que les propriétés précisant des caractéristiques relient un concept à un littéral. Par exemple, "estEnseignéPar" est une propriété objet décrivant des relations

entre objets, qui relie ici le concept "Cours" au concept "Enseignant". En revanche, "APourDescription" est une propriété datatype caractérisant un cours qui relie le concept "Cours" à un littéral (une chaîne de caractères ici). Les propriétés d'annotation sont des propriétés bien spécifiques d'une ontologie dénotant par exemple le label (rdfs:label) ou bien des commentaires (rdfs:comment) sur des éléments ontologiques.

Les axiomes décrivent des connaissances sur les entités de l'ontologie. Il y a deux types d'axiomes : les axiomes terminologiques, qui portent sur les caractéristiques des concepts et des propriétés ; et les assertions (cf. paragraphe sur les individus), qui sont des faits qui portent sur des individus spécifiques et qui expriment qu'un individu appartient à un concept ou qu'une propriété relie deux individus donnés.

Les axiomes terminologiques permettent d'exprimer la subsomption de concepts (concept "Enseignant" sous-concept de "Personne"), l'équivalence entre des entités ontologiques (concept "Enseignant" équivaut à "il existe au moins un cours qu'il enseigne"), le domaine/co-domaine d'une propriété, les caractéristiques des propriétés (fonctionnelle, transitive, etc.), la disjonction entre deux entités.

L'enrichissement d'ontologie est le processus d'extension d'une ontologie avec de nouveaux concepts, propriétés et/ou axiomes terminologiques.

Les individus sont les instances du domaine. Par exemple, "Java" est un individu. Les assertions expriment le typage des individus (l'individu "Java" est une instance du concept "Cours") et des propriétés (le triplet <"Java", "estEnseignéPar", "Monsieur Duchemol"> est une instance de la propriété "estEnseignéPar"). Une instance de propriété est appelée assertion de propriété.

Le peuplement d'ontologie est le processus d'ajout de nouvelles instances dans une ontologie [Petasis *et al.* 2011]. Il s'agit d'ajouter des individus instances de concepts, mais aussi des assertions de propriétés relatives à des instances.

Le World Wide Web Consortium (W3C) est un organisme de standardisation à but non lucratif. Celui-ci a mis en place un groupe de travail consacré au développement de langages standards pour modéliser des ontologies utilisables et échangeables sur le Web. Le W3C a publié en 2004 une recommandation définissant le langage OWL (Web Ontology Language) [McGuinness & van Harmelen 2004], fondé sur le standard RDF, qui a rapidement pris une place prépondérante dans le paysage des ontologies. OWL est désormais le standard le plus utilisé. Il contient plusieurs familles (OWL 1, OWL 2). Les concepts y sont représentés sous forme de classes.

Caractérisation de l'approche à base d'ontologie proposée

Dans notre contexte, étiqueter les documents d'un corpus revient à attacher des informations complémentaires, i.e., des métadonnées, à ces documents. Cela correspond à un processus d'annotation. De plus, lorsque les métadonnées considérées sont des informations sémantiques (en relation avec l'identité sémantique des données annotées) concernant le contenu textuel, on parle d'annotation sémantique. La problématique énoncée par Wepingo est donc une **problématique d'annotation sémantique**.

L'annotation sémantique dans notre contexte est fortement liée au peuplement d'ontologie. En effet, annoter une entité avec un concept revient, dans une ontologie, à instancier une classe représentant ce concept avec cette entité. Ainsi, nous considérons notre problème d'annotation sémantique comme un problème de peuplement d'ontologie. Les concepts cibles devront alors être ajoutés dans une ontologie du domaine (enrichissement) puis instanciés (peuplement). Notre problématique est donc une **problématique de peuplement et d'enrichissement d'ontologie**.

Conclusion

Ce chapitre a introduit notre problématique. Nous traitons d'un problème d'**annotation sémantique** de documents décrivant des entités d'un même domaine. Ce problème peut être ramené à un problème **d'enrichissement et de peuplement d'ontologie**.

La partie I de cette thèse traite de ce problème. La partie II traite de la complétion des informations extraites à partir de ressources externes. Le processus de complétion proposé est adapté à l'exploitation du Web des données.

Partie I

Annoter des documents via un peuplement et un enrichissement d'ontologie

Chapitre 3

État de l'art : Annotation sémantique de documents, peuplement et enrichissement d'ontologie

Sommaire

Dans ce chapitre, nous présentons un état de l'art des différents travaux liés à la première partie de notre problématique. Nous nous focalisons tout d'abord sur

l'annotation sémantique de documents et décrivons des travaux de ce domaine. Par la suite, nous nous intéressons aux travaux de peuplement et d'enrichissement d'ontologie.

3.1 L'annotation sémantique de documents

Cette section définit l'annotation et plus précisément l'annotation sémantique [Oren *et al.* 2006]. Elle explicite les procédés existants pour réaliser une annotation.

L'annotation est un processus qui vise à attacher des données complémentaires à d'autres données. En général, celle-ci concerne des documents textuels, mais on peut aussi annoter d'autres éléments, comme des images, procédé qui consiste à assigner automatiquement une légende ou des mots-clés à une image numérique. Cet état de l'art se focalise sur l'annotation de documents textuels. L'annotation de documents est un processus qui vise à attacher des informations complémentaires (métadonnées) au contenu textuel d'un document. L'annotation sémantique de documents est un type d'annotation bien précis. Les contenus textuels sont reliés à des informations sémantiques les concernant, c'est-à-dire en relation avec l'identité sémantique des données annotées.

Exemple 1. Prenons la phrase "Paris est la capitale de la France". Supposons qu'on souhaite attacher des informations sémantiques à cette phrase, en utilisant la ressource Wikipédia, une encyclopédie universelle et multilingue. Un résultat d'annotation sémantique possible serait
"(Paris, https://fr.wikipedia.org/wiki/Paris) est la capitale de la (France, https://fr.wikipedia.org/wiki/France)".
Ainsi, des méta-données (pages Wikipédia) sont attachées à certains fragments de la phrase.

Nous décrivons par la suite deux types de méthodes d'annotation sémantique : d'une part l'attachement d'informations complémentaires à des parties de documents et d'autre part l'attachement d'informations à des documents, globalement, sans identifier de parties précises concernées.

3.1.1 Les méthodes d'annotation sémantique : attachement d'informations complémentaires à des fragments textuels au sein d'un document

Les méthodes d'annotation sémantique de documents peuvent être classées en deux catégories [Reeve 2005] : (1) basées sur l'utilisation de patrons et (2) basées sur l'application de techniques d'apprentissage automatique [Manning & Schütze 1999]. Les annotations consistent en général à typer une entité.

Les méthodes d'annotation sémantique basées sur l'utilisation de patrons utilisent des patrons, définis manuellement ou découverts. Un patron est une forme syntaxique

qui, si elle est reconnue dans un texte, permet d'associer par règle, une annotation à l'un de ses composants. Par exemple, "X was born in Y" est un patron permettant d'annoter X comme une personne. Pour découvrir des patrons automatiquement, la plupart des approches suivent la méthode décrite dans [Brin 1998] : on utilise un ensemble de documents et un ensemble initial d'entités. Les documents sont analysés afin d'identifier les formes syntaxiques faisant intervenir ces entités. Par exemple, si on dispose d'une liste de livres avec leurs auteurs respectifs, nous pouvons trouver dans des documents des références à ces livres et examiner leur contexte. "X by Y" pourra être un patron trouvé reliant un livre X écrit par un auteur Y. Il permettra par la suite d'annoter les entités correspondantes à ce patron en tant que livre et auteur. De nouveaux patrons et de nouvelles annotations peuvent être découverts pendant ce processus, qui s'arrête quand plus rien de nouveau ne peut être découvert (ni patrons, ni annotations).

D'autres méthodes d'annotation sémantique appliquent des techniques d'apprentissage automatique. Les méthodes probabilistes se basent sur des modèles statistiques pour prédire l'annotation d'un élément en fonction des autres éléments d'un document. Elles sont par exemple utilisées pour annoter les mots d'une phrase en fonction de leur catégorie grammaticale (part-of-speech tagging) : nom, verbe, adjectif, etc. D'autres méthodes [Vargas-Vera *et al.* 2002, Handschuh *et al.* 2002] mettent en œuvre de l'induction et ont pour but de générer des règles d'extraction d'informations, plus précisément des règles permettant de découvrir des instances de concepts donc de faire des annotations. Pour cela, elles utilisent un ensemble d'entraînement annoté et généralisent les exemples issus de cet ensemble pour construire des règles linguistiques ou bien structurelles.

Dans ces deux types d'approche, l'idée est de rechercher les fragments textuels qui mentionnent une entité dans des documents.

Les méthodes d'extraction d'informations se sont beaucoup focalisées sur un processus d'annotation particulier : l'annotation (ou la reconnaissance) des entités nommées [Nadeau & Sekine 2007] dans des textes.

Une entité nommée (EN) est une unité textuelle particulière qui réfère à un élément unique existant : un pays, un acteur, un livre, etc.

La reconnaissance d'entités nommées est une sous-tâche de l'extraction d'information de documents textuels. Elle consiste à rechercher des objets textuels (c'est-à-dire un mot ou un groupe de mots) catégorisables dans des classes : noms de personnes, noms d'organisations ou d'entreprises, noms de lieux, quantités, distances, valeurs, dates, etc.

Exemple 2. Prenons la phrase : "William Bradley Pitt, dit Brad Pitt, est un acteur et producteur de cinéma américain né le 18 décembre 1963 à Shawnee, dans l'Oklahoma." Cette phrase fait référence à de multiples entités nommées (noms de personnes, lieux, dates). Elle peut être annotée comme suit :

"(William Bradley Pitt, PERSONNE), dit (Brad Pitt, PERSONNE), est un acteur et producteur de cinéma américain né le (18 décembre 1963, DATE) à (Shawnee, LIEU) dans l'(Oklahoma, LIEU)."

Diverses approches d'annotation et d'extraction d'informations ont été mises en œuvre au sein d'outils. Beaucoup de ces systèmes, comme KIM [Popov *et al.* 2004], SOFIE [Suchanek *et al.* 2009] ou C-PANKOW [Cimiano *et al.* 2005b] extraient des groupes nominaux correspondant à des entités nommées. Celles-ci sont repérables grâce à des grammaires formelles souvent manuellement définies ou bien apprises. Certaines approches se basent sur des frameworks d'extraction d'informations existants. Par exemple, KIM se base sur le framework GATE [Cunningham *et al.* 2011] qui contient beaucoup de composants utiles pour l'extraction d'informations (tokenizers, part-of-speech taggers, gazetteers, etc.).

Fig. 3.1 Annotations obtenues avec DBpedia Spotlight

Plusieurs outils d'annotation exploitent des bases de connaissances existantes (Wikipedia, DBpedia) afin d'annoter les entités nommées d'un texte. Ainsi, DBpedia Spotlight [Mendes *et al.* 2011, Daiber *et al.* 2013], Wikifier [Cheng & Roth 2013, Ratinov *et al.* 2011] ou encore AIDA [Yosef *et al.* 2011] sont des outils permettant d'associer à des fragments textuels les pages de DBpedia (dans le cas de DBpedia Spotlight) ou de Wikipedia (pour Wikifier et AIDA) qui leur correspondent. Par exemple, DBpedia Spotlight est capable d'associer les IRI des

instances de concepts de DBpedia à un texte donné en entrée. La Figure 3.1 montre un texte annoté par DBpedia Spotlight. La première annotation trouvée (Berlin) correspond à la page DBpedia : http://dbpedia.org/resource/Berlin. Tous les mots ou groupes de mots soulignés et bleus correspondent à des entités reconnues et renvoient à la page DBpedia correspondante.

L'identification d'entités qui ne sont pas des entités nommées est beaucoup plus délicate car aucune base ne répertorie a priori l'ensemble de ces entités et encore moins les expressions linguistiques qui leur sont associées.

Ces ensembles d'entités et la terminologie propre au domaine doivent donc être recueillies pour construire la "gazeteer" (liste) adaptée à un domaine particulier. Par exemple, [Amardeilh & Damljanovic 2009] prétraitent l'ensemble des termes présents dans les différentes ressources d'une ontologie (classes, instances, propriétés, valeurs de propriétés) pour en extraire un ensemble de lemmes à partir desquels est constituée la "gazeteer" associée à cette ontologie.

D'autres approches exploitent la structure du document à annoter. Par exemple, dans [Amardeilh *et al.* 2005], la structure d'un document est représentée sous la forme d'un arbre conceptuel dont chaque nœud est mis en correspondance avec un concept de l'ontologie via des règles définies manuellement. De même, [Aussenac-Gilles *et al.* 2013] définissent des règles d'extraction en exploitant la structure hiérarchique exprimée par les marqueurs typo-dispositionnels (police en gras, en italique, symbole de ponctuation ':') au sein d'un ensemble de fiches de même format.

Les approches d'annotation décrites jusqu'ici font référence à l'attachement d'informations complémentaires (métadonnées) à des fragments textuels au sein d'un document. Les approches décrites par la suite cherchent à attacher des informations complémentaires non pas à des éléments mentionnés dans un document mais à un document tout entier. Ces approches évaluent la proximité entre la description d'une entité (texte plus ou moins long) et des éléments d'annotation (d'autres documents, des instances de concepts, des concepts).

3.1.2 Les méthodes d'annotation sémantique : évaluation de la proximité entre la description d'une entité et les éléments utilisables pour l'annoter

Cette section relate divers travaux visant à effectuer un lien entre une description d'entité et des éléments particuliers (documents, instances de concepts, concepts) utilisés pour annoter cette description considérée comme un tout.

Tout d'abord, certains travaux se basent sur des mesures de similarité. Ainsi, l'objectif de [Kessler *et al.* 2012] est de vérifier l'adéquation entre des offres d'emploi et des candidatures à ces offres (CV et lettres de motivation), c'est-à-dire évaluer

la proximité entre la description d'un élément général (une offre d'emploi) et celles d'éléments plus particuliers (des candidatures). Après avoir été soumis à différents traitements, tous les documents manipulés sont représentés par des vecteurs qui sont ensuite comparés en utilisant des combinaisons de diverses mesures de similarité (cosinus, Minkowski, ...). Ceci permet de classer les candidatures. De plus, pour être sûr qu'une candidature ne soit pas trop vite écartée, sa similarité avec le vecteur représentant l'offre d'emploi, enrichie des candidatures jugées pertinentes par un recruteur, est aussi évaluée.

Dans [Béchet *et al.* 2011], l'objectif est d'annoter des mots ou expressions désignant des services d'hôtels décrits par des hôteliers. Par exemple, un même service peut être décrit comme "climatisation" par un hôtelier et par "air conditionné" par un autre. Il en va de même pour beaucoup de services ("wifi"/"internet sans fil", "coffre-fort"/"coffre"/"coffre-fort électronique", etc). L'objectif de ce travail est d'associer une expression de service hôtelier à un concept la décrivant. La liste des services de chaque hôtel est définie par l'hôtelier avec son propre vocabulaire, sous la forme d'une liste. Les auteurs proposent une approche en deux étapes. La première étape consiste à définir une structure hiérarchique des services avec l'aide d'un expert qui va aussi choisir un premier ensemble d'instances pour peupler cette structure. Ces instances sont obtenues à partir d'une liste d'instances de services prédéfinie par un partenaire. Ainsi, le concept "remise en forme" est instancié par les services "salles de gym", "centre fitness", "centre de santé", etc. La seconde étape est le peuplement automatique de la structure hiérarchique avec les services des hôtels. Autrement dit, il s'agit de relier chacun des services définis par les hôteliers à la structure hiérarchique. On suppose que les nouveaux services sont littéralement proches des instances initialement créées. En effet, on ne peut rapprocher des termes comme "wifi" et "internet sans fil" que si le concept qui leur est associé a été instancié avec, différentes instances telles que chacun des termes est littéralement proche d'au moins une de ces instances. Chaque nouveau service est comparé aux instances existantes et sera considéré comme une instance du concept correspondant à l'instance dont il est le plus proche. La proximité entre un nouveau service et une instance pré-existante est basée sur un calcul de similarité utilisant les n-grammes.

Ces deux travaux évaluent une proximité grâce à des mesures de similarité. Ces mesures de similarité sont utilisées sur divers types d'objets : des mots ou expressions textuelles [Béchet *et al.* 2011] ou bien des vecteurs représentant les documents [Kessler *et al.* 2012]. D'autres travaux utilisent des techniques d'apprentissage automatique.

Dans [Alec *et al.* 2014b, Alec *et al.* 2014a, Alec *et al.* 2016f], on cherche à annoter des descriptions de produits avec des éléments qui ne sont pas des entités nommées. Par exemple, on aimerait associer une description d'un jouet à ses catégories (jeu de construction, jeu de hasard, etc.) et ses caractéristiques (concentration,

dextérité, coopération, etc.). Les descriptions sont non structurées, sans aucune homogénéité, et ne mentionnent pas ces éléments. Comme des mesures de similarité ne sont pas possibles dans ce contexte, une première annotation manuelle est effectuée par un expert, aidé par un outil d'aide à l'annotation qui va orienter son choix en lui proposant et suggérant des annotations via une interface graphique. Par la suite, cet ensemble annoté est utilisé comme base d'exemples par une technique d'apprentissage automatique (SVM). Les documents sont représentés d'une manière vectorielle. Plusieurs représentations sont testées, de type sac-de-mots [Salton & McGill 1986]. Ainsi, l'approche peut prédire pour un document donné si celui-ci doit être annoté par un concept (une catégorie ou une caractéristique) ou non.

Cependant, les modèles classiques de sacs-de-mots ne tiennent pas toujours bien compte de la sémantique. Par exemple, l'expression "cloud computing" est découpée en deux mots pouvant perturber le sens de la phrase originale puisque le mot "cloud" peut potentiellement faussement activer des concepts liés à la météo. Ce genre d'expression, nommée expression polylexicale (multiword expression), peut être défini comme "une unité syntaxique et sémantique dont le sens exact ou la connotation ne peuvent pas être dérivés directement et sans ambiguïté du sens ou de la connotation de ses composantes" [Choueka 1988]. Pour gérer ce genre d'expressions, certains travaux utilisent des dictionnaires linguistiques qui analysent la phrase : ponctuation spéciale, mots en majuscule, onomatopées, adverbes de degré, émoticônes. C'est le cas d'AffectiveSpace 2 [Cambria *et al.* 2015], un framework d'analyse de sentiments au niveau concept (concept-level sentiment analysis), qui prédit l'intuition affective d'un texte (joie, tristesse, etc.) grâce à un dictionnaire linguistique et des analyseurs affectifs (sentic parsers) qui permettent une analyse avancée de la polarité de mots en prenant en compte leurs nuances. D'une manière générale, l'analyse au niveau concept [Cambria & White 2014] (concept-level analysis) se focalise sur une analyse des textes reposant sur des caractéristiques considérées comme implicites parce que non représentées dans les textes. Pour ce faire, elle repose entre autres sur une modélisation des textes sous la forme de sacs-de-concepts permettant de mieux représenter la sémantique associée au langage naturel que le classique sac-de-mots. Pour cela, elle utilise des ontologies ou des réseaux sémantiques.

Enfin, dans le cas de documents structurés, la structure peut être exploitée pour déduire des annotations. C'est le cas dans le travail de [Toussaint & Tenier 2007] qui présente un système d'annotation sémantique visant à classifier des pages web. Par exemple, on cherche à identifier si une page web décrit une équipe de recherche. La présence de certains éléments est révélatrice d'une équipe de recherche : des personnes, des publications, des thèmes de recherche. Grâce aux connaissances représentées dans une ontologie, des déductions peuvent être effectuées. Ainsi, si l'ontologie définit un chercheur comme une personne ayant publié des papiers, alors on peut déduire qu'une page web décrivant des personnes avec des publications est une page décrivant des chercheurs. De plus, si une équipe de recherche est définie comme étant composée de

chercheurs, cette page peut être annotée comme une page décrivant une équipe de recherche. La structuration des pages web permet de définir des modèles de structure, c'est-à-dire des représentations arborescentes dont les nœuds sont des balises HTML, et de les associer à un contenu. Par exemple, un modèle peut être défini pour représenter un chercheur. Ainsi, l'exploitation conjointe de la structure des pages web et de l'ontologie permet d'obtenir l'annotation souhaitée.

En conséquence, les travaux évaluant la proximité entre une description et un élément d'annotation utilisent des mesures de similarités, des techniques d'apprentissage automatique ou encore la structure des documents. L'annotation sémantique de documents est un vaste champ de recherche qui regroupe des techniques d'attachement d'informations complémentaires à des fragments textuels au sein d'un document ou bien au document tout entier.

La section suivante fait état des travaux de peuplement et d'enrichissement d'ontologie.

3.2 Peuplement et enrichissement d'ontologie

Le peuplement et l'enrichissement d'ontologie sont des sous-tâches de l'apprentissage d'ontologie (ontology learning) [Petasis *et al.* 2011]. L'apprentissage d'ontologie est le processus d'acquisition (construction ou intégration) d'une ontologie (semi-) automatiquement. Ici, nous nous focalisons principalement sur l'extension d'une ontologie existante. Cette extension peut être de deux types : le peuplement d'une ontologie et l'enrichissement d'une ontologie. Elle se base en général sur des informations extraites à partir d'un contenu spécifique à un domaine. Le peuplement d'ontologie est le processus d'insertion d'instances de concepts et/ou d'instances de propriétés dans une ontologie existante. Le peuplement ne change pas la structure de l'ontologie (hiérarchie et propriétés). L'enrichissement d'ontologie est le processus d'extension d'une ontologie, à travers l'ajout de nouveaux concepts, propriétés et axiomes. Ainsi, la structure de l'ontologie évolue au cours de ce processus. Nous distinguerons trois catégories de travaux, portant principalement sur l'extraction de connaissances sémantiques à partir de textes plus ou moins structurés : (1) le repérage d'éléments ontologiques dans des textes et leur extraction, (2) le raisonnement pour dériver des concepts complexes non présents dans les textes à partir de concepts primitifs extraits et (3) l'extraction et la formalisation de définitions de concepts.

3.2.1 Repérage d'éléments ontologiques dans des textes et leur extraction

La première catégorie de travaux concerne la génération d'ontologies légères à l'expressivité limitée, souvent réduites à des taxonomies. Les travaux considérés étudient comment extraire différents éléments ontologiques à partir de ressources

textuelles [Cimiano 2006]. Plusieurs tâches existent : extraction de termes, extraction de synonymes, extraction de concepts, extraction de relations et extraction d'axiomes. Nous nous intéressons dans cette section à l'extraction de concepts.

L'étape primordiale est l'extraction de la terminologie pertinente du domaine et la découverte de synonymes [Cimiano *et al.* 2006]. Les termes extraits dans les textes sont supposés dénoter soit un concept soit une instance. Différentes mesures de pondération des termes telles que la fréquence des termes, le TF-IDF ou l'entropie sont utilisées pour identifier les termes qui sont les plus pertinents pour le domaine considéré. Ces termes sont par la suite filtrés par un expert du domaine. La distinction entre les concepts et les instances dépend de la catégorie grammaticale (part-of-speech) associée avec sa représentation lexicale, i.e., les noms propres et communs [Völker 2009]. Des techniques d'apprentissage non supervisées sont ensuite appliquées pour détecter les synonymes. Une classe ontologique peut être dérivée pour chaque groupe de termes similaires.

D'autres travaux s'intéressent à l'apprentissage de hiérarchies de concepts à partir de textes. Ils utilisent principalement des techniques de classification hiérarchiques non supervisées afin d'apprendre en même temps les concepts et leurs relations de subsomption [Caraballo 1999, Cimiano *et al.* 2005a, Faure & Nedellec 1999]. De plus, certaines approches peuvent introduire une étape supervisée en demandant à l'utilisateur de valider ou rejeter certaines classes (clusters) comme dans le système ASIUM [Faure & Nedellec 1999]. D'autres approches [Cimiano *et al.* 2004] identifient des patrons dans les textes mais celles-ci découvrent des relations lexicales entre des termes, pas entre des concepts.

Enfin, lorsque l'ontologie n'a pas à être intégralement construite et qu'une hiérarchie de concepts existe déjà et doit être étendue avec de nouveaux concepts, les méthodes supervisées deviennent possibles. Des classifieurs doivent être entraînés pour chaque concept de l'ontologie existante. Ces concepts ne doivent pas être très nombreux. Les approches non supervisées, quant à elles, utilisent des mesures de similarité appropriées pour comparer un nouveau concept avec ceux déjà existants [Cimiano & Völker 2005]. Dans le cas supervisé comme non supervisé, le ou les concepts les plus proches sont trouvés, et le nouveau concept devient le parent de ceux-ci [Maedche & Staab 2004, Pekar & Staab 2002].

Tous ces travaux visent à reconnaître les termes désignant des concepts (ou des instances) dans les textes, puis à les extraire. Cependant, parfois, les textes n'évoquent que les propriétés des instances sans nommer le concept concerné. Dans ce cas, les approches doivent utiliser du raisonnement pour dériver des connaissances non explicitées dans les textes. C'est le cas des approches décrites ci-après.

3.2.2 Raisonnement pour dériver des concepts complexes non présents dans les textes à partir de concepts primitifs extraits

La deuxième catégorie inclut les travaux qui utilisent le raisonnement pour remplacer partiellement les techniques traditionnelles d'extraction. Quand les concepts utilisés pour annoter ne sont pas explicitement mentionnés dans les descriptions à analyser, un processus en (au moins) deux temps doit être effectué. La première étape est une tâche classique d'extraction tandis que la seconde étape est une tâche de raisonnement sur les résultats de la première étape. À nos connaissances, deux travaux suivent cette idée. Les deux utilisent des ontologies.

Le système BOEMIE [Petasis *et al.* 2013] distingue deux types de concepts : primitifs et composites. Leur différence tient dans le fait que des instances de concepts composites ne sont pas explicitement représentées dans les textes donnés en entrée (ou images ou vidéos) alors que les instances des concepts primitifs et de leurs propriétés le sont. BOEMIE fonctionne en deux étapes : la première consiste à rassembler des instances de concepts et de propriétés de l'ontologie, la seconde raisonne sur ces instances extraites. Les concepts composites sont définis à partir des concepts primitifs. La première étape peuple les concepts primitifs et leurs propriétés avec des outils d'extraction classiques à partir de textes (ou autres supports). La seconde étape applique des règles dites d'interprétation données en entrée dans l'ontologie. Ces règles permettent de définir un concept composite en fonction de concepts primitifs et de leurs propriétés. Elles permettent aussi de peupler une propriété entre deux concepts composites en fonction des caractéristiques de ces deux concepts composites (type, propriétés). Par exemple, la règle définissant "personToRanking(X,Z)" est la suivante :

$$personToRanking(X,Z) \leftarrow Person(X), PersonName(Y), hasPersonName(X,Y), personNameToRanking(Y,Z)$$

Cette étape permet de raisonner sur les instances préalablement extraites (concepts primitifs et leurs propriétés) et d'en déduire des instances de concepts composites et de leurs propriétés.

[Yelagina & Panteleyev 2014] a pour but d'extraire de nouveaux faits à partir de faits décrits dans un texte et de connaissances décrites dans une ontologie. L'approche vise à traiter les cas où des faits pertinents ne sont pas mentionnés explicitement dans les textes analysés. Ces faits, dans certains cas, peuvent être inférés à partir de faits contenus dans les textes et de connaissances basiques. L'approche utilise une ontologie pour extraire les faits des textes et pour dériver de nouveaux faits. Plus précisément, la première étape extrait des faits de textes grâce à des outils de traitement automatique de la langue et à une ontologie. La seconde étape complète les faits extraits. Les nouveaux faits sont appris à partir des faits préalablement extraits et de connaissances

ontologiques. Cet apprentissage est fait via des règles d'inférence, écrites manuellement, reposant sur des connaissances du domaine. Par exemple,

hasSubsystem(?Product, ?Subsystem) $\cap \ldots \cap$isUsedFor(?Technology, ?Material) $\cap$ planToProduce(?Entreprise, ?Product) $\rightarrow$ isPotentialConsumer(?Technology, ?Enterprise)

est une règle déduisant le fait "isPotentialConsumer" à partir d'autres faits.

Ces deux approches arrivent à extraire, à partir de textes, des informations non explicitement mentionnées dans ceux-ci. Pour cela, chaque approche opte pour un processus en deux étapes. Leur première étape repose sur l'extraction des informations explicitement décrites dans les textes. Ces informations sont entrées dans l'ontologie sous la forme d'instances via un peuplement de concepts et/ou de propriétés. La seconde étape se base sur des règles du domaine prédéfinies pour déduire de nouveaux faits, i.e. de nouvelles instances de concepts et/ou propriétés.

Les approches décrites raisonnent sur des définitions de concepts qui sont données en entrée, ce qui n'est pas toujours le cas. Les travaux décrits dans la suite se focalisent sur l'extraction et la formalisation de définitions de concepts.

3.2.3 Extraction et formalisation de définitions de concepts

Cette troisième catégorie de travaux concerne la génération de définitions de concepts. Certaines approches travaillent sur des textes décrivant des concepts.

LExO [Völker *et al.* 2007], par exemple, applique des règles de transformation syntaxiques sur des définitions données en langue naturelle pour générer des axiomes en Logique de Description (LD). Par exemple, la phrase "Data: Facts that result from measurements or observations" est automatiquement traduite en :
Data $\equiv$ Fact $\cap \exists$ result_From.(Measurement $\cup$ Observation).

[Ma & Distel 2013b] ont, quant à eux, une approche basée sur l'extraction de relations et s'appuient sur des contraintes formelles pour assurer la qualité des définitions apprises [Ma & Distel 2013a]. Des informations textuelles (phrases) issues du domaine médical sont automatiquement mises en relation avec les informations médicales répertoriées dans l'ontologie SNOMED CT. En effet, l'outil Metamap (US National Library of Medicine) est capable d'annoter les mots d'une phrase avec les concepts de SNOMED CT. À partir des relations de SNOMED CT (explicites et implicites), une mise en correspondance est possible entre une relation et une phrase analysée. De ce fait, on en déduit des sortes de patrons exprimant une relation bien particulière, par exemple l'expression "is pneumoconiosis caused by" identifiée entre une entité de type "disorder" et une entité de type "substance" fait référence à la relation "Causative_agent". Grâce à cela, de nouvelles instances de relations, non répertoriées dans SNOMED CT, peuvent être découvertes dans les textes.

Ce type d'approche est capable de traduire des définitions exprimées en langage naturel en définitions exprimées en logique de description. Les définitions sont traduites. Il ne s'agit pas de les découvrir. Elles existent déjà dans les informations textuelles données en entrée des systèmes.

D'autres approches n'ont pas de définitions de concepts données en entrée. Ainsi, [Chitsaz 2013] et [Lehmann & Hitzler 2010] ne disposent que de descriptions d'instances. Ces travaux s'appuient sur la programmation logique inductive, plus particulièrement sur des opérateurs de raffinement, utilisés pour traverser l'espace de recherche en généralisant (upward refinement) ou spécialisant (downward refinement) les hypothèses. Par exemple, un opérateur de raffinement appliqué sur "Male" peut donner "Male $\cap \exists$hasChild.$\top$". En partant des connaissances d'une ontologie et d'exemples positifs et négatifs, ces méthodes sont capables d'apprendre la définition d'un concept. Par exemple, si on a :

- Male(MARC), hasChild(MARC, ANNA), Female(ANNA), ...
- positifs: {MARC, ...}, négatifs: {ANNA, ...}

alors le concept peut être appris comme étant : Male $\cap \exists$hasChild.$\top$.

L'approche proposée par [Lehmann & Hitzler 2010] est appliquée sur des ontologies expressives représentées en logique de description, celle de [Chitsaz 2013] sur des ontologies légères. Les deux approches nécessitent un grand nombre d'assertions relatives aux instances.

Ces travaux permettent l'extraction et la formalisation de définitions de concepts. Soit la définition est exprimée dans les documents à traiter : dans ce cas, celle-ci est traduite en Logique de Description. Soit la définition d'un concept est inconnue mais elle peut être apprise à partir d'exemples et d'assertions de propriété.

La section suivante positionne notre travail par rapport aux travaux présentés dans ce chapitre.

3.3 Positionnement de notre travail par rapport à l'état de l'art

Comme les travaux d'annotation présentés, nous voulons faire correspondre une description avec des concepts d'un domaine, c'est-à-dire annoter ces descriptions avec ces concepts. Cependant, nous faisons face à un enjeu supplémentaire : nous voulons des annotations compréhensibles. Cela signifie que le processus d'annotation ne peut pas être une boîte noire. En effet, dans certains cas, un concept (besoin utilisateur) exprimé par un utilisateur de Wepingo peut n'être

associé à aucune description. Le cas échéant, Wepingo se doit tout de même de proposer des produits à l'utilisateur. Des descriptions annotées positivement doivent donc être trouvées pour tous les concepts cibles, correspondant à une satisfaction totale ou partielle du besoin. Cette objectif rend notre travail tout à fait particulier.

Aucune des approches citées, prises isolément, n'est une solution à notre problème. Les travaux générant une ontologie extraient seulement des éléments ontologiques, reconnus par l'intermédiaire des termes qui les dénotent, dans les textes. Ces techniques sont inapplicables car les textes sur lesquels nous travaillons n'incluent pas les noms des concepts cibles, qui sont trop spécifiques pour être mentionnés.

Les textes que nous traitons sont des descriptions d'entités. Ils évoquent donc des caractéristiques d'objets. De ce fait, les informations intéressantes à extraire des textes ne sont pas des entités nommées. Elles dépendent du domaine considéré. De plus, la structure de la description n'est pas exploitable. Les textes ne sont pas structurés.

Les deux approches appliquant des techniques de raisonnement [Petasis *et al.* 2013, Yelagina & Panteleyev 2014] sont les travaux les plus proches de notre problématique. En effet, ils cherchent à extraire de textes des informations qui n'y sont pas explicitées. Pour cela, ils peuplent tout d'abord l'ontologie avec les informations exprimées dans les textes. Par la suite, l'application de définitions connues au préalable permettent d'inférer les informations recherchées mais non explicitées dans les textes. Dans notre cas, cette idée de processus en deux étapes pourrait être appropriée. Il nous manque cependant les définitions des concepts cibles. Il nous faudra les apprendre.

Apprendre des définitions des concepts cibles semble par ailleurs être une bonne idée pour éviter l'opacité des annotations. En effet, cela permettrait de comprendre les raisons des annotations positives et négatives et de les raffiner si besoin. Les approches qui génèrent des définitions formelles de concepts à partir de textes ne sont toutefois pas directement applicables dans notre contexte. En effet, certaines d'entre elles [Völker *et al.* 2007, Ma & Distel 2013b] se basent sur un contenu textuel exprimant des définitions. Or ce n'est pas le cas de nos descriptions de produits. En revanche, les autres approches [Lehmann & Hitzler 2010, Chitsaz 2013] se basent sur (1) des faits décrits dans une ontologie et sur (2) des exemples positifs et négatifs d'un concept pour apprendre la définition de celui-ci. Dans notre cas, les faits ontologiques (1) peuvent être extraits des documents. Quand des informations manquent dans les documents, ces faits peuvent être complétés à partir de ressources externes. Pour les exemples (2), il nous est possible de demander des annotations manuelles au concepteur du système, expert du domaine.

Nous allons donc utiliser une ontologie de domaine qui servira à la fois de support pour intégrer tout un ensemble de données d'un même domaine au sens général du terme et pour raisonner dessus. Les données intégrées proviendront des documents

étudiés et seront complétées par des informations externes provenant du Web des données. Ces informations caractériseront les entités décrites sous la forme d'assertions de propriété (**peuplement d'ontologie**). De plus, pour pouvoir raisonner sur les concepts cibles, nous les introduirons dans l'ontologie sous la forme de classes, que nous appellerons **classes cibles** (**enrichissement d'ontologie**). Annoter un document décrivant une entité avec un concept cible revient alors à instancier la classe cible correspondante. Il nous faut donc trouver si l'entité décrite dans le document est une instance d'une certain classe cible ou non. Si c'est le cas, la classe cible fera l'objet d'une annotation positive et si ce n'est pas le cas, négative. Pour résoudre ce **problème de peuplement d'ontologie**, nous devons comprendre précisément ce à quoi les classes cibles correspondent, c'est-à-dire connaître leurs définitions. Celles-ci sont nécessaires pour être capable de formuler des réponses aux utilisateurs même quand leurs besoins ne peuvent être totalement satisfaits.

Nous faisons l'hypothèse qu'un expert du domaine n'est pas en mesure de fournir des définitions précises des concepts cibles mais qu'en revanche, il est capable d'étiqueter des documents, qui seront ensuite utilisés pour **apprendre automatiquement la définition de chaque concept cible**.

En résumé, le positionnement précédent signifie que nous devons combiner plusieurs processus pour résoudre notre problématique :

- un processus d'extraction d'assertions (instances) de propriété caractérisant les entités à traiter et l'ajout de ces entités et des assertions de propriété associées dans l'ontologie,
- un processus d'apprentissage automatique de définitions de concepts, qui seront apprises grâce à des exemples annotés (positifs et négatifs) pour chaque concept cible et grâce à l'ontologie peuplée avec les caractéristiques (assertions) des entités,
- un processus de raisonnement pour appliquer les définitions de concepts. Grâce aux définitions des concepts cibles apprises, il sera possible d'inférer quelles entités sont instances de quels concepts cibles, ce qui permettra d'annoter ces entités avec les concepts dont elles sont instances.

Conclusion

Ce chapitre a dressé un état de l'art des travaux d'annotation sémantique de documents ainsi que d'enrichissement et de peuplement d'ontologie. Nous avons positionné notre travail par rapport à ces travaux. Nous en avons ainsi déduit un ensemble de processus à exécuter pour répondre au problème. Le chapitre suivant décrit l'approche proposée mettant en œuvre ces processus pour répondre à la problématique.

Chapitre 4

Une approche de peuplement et d'enrichissement d'ontologie à partir de textes et de données ouvertes afin d'annoter des documents

Sommaire

Ce chapitre présente l'approche que nous avons développée pour répondre à la problématique d'étiquetage de documents d'un même domaine décrivant chacun une entité. Le but de l'approche est d'annoter chaque document avec une liste de concepts donnée en entrée. Si un document correspond à un concept de la liste, dit concept cible, alors nous annoterons le document avec ce concept cible et parlerons d'annotation positive. Si ce n'est pas le cas, nous annoterons le document avec la négation de ce concept cible et parlerons d'annotation négative. Nous annoterons ainsi chacun des documents avec chacun des concepts cibles. Chaque document aura donc autant d'annotations qu'il y a de concepts cibles. Cette

approche, nommée SAUPODOC, acronyme de "Semantic Annotation Using Population of Ontology and Definitions Of Concepts" a fait l'objet de plusieurs publications scientifiques [Alec *et al.* 2016c, Alec *et al.* 2016b, Alec *et al.* 2016e].

L'approche repose sur l'utilisation d'une ontologie de domaine progressivement peuplée avec des informations extraites à partir à la fois d'un corpus de documents à annoter et de ressources externes. Ensuite, les définitions des concepts cibles sont apprises en se basant sur l'ontologie peuplée et sur une partie des documents du corpus, manuellement annotée. Enfin, la dernière étape est une étape de raisonnement, où les définitions sont appliquées pour générer les annotations des documents du corpus non annotés manuellement.

La suite de ce chapitre est organisée comme suit. La Section 4.1 décrit l'approche. La Section 4.2 présente les diverses tâches mises en jeu dans celle-ci.

4.1 Description de l'approche

L'approche SAUPODOC est automatique et applicable sur divers domaines. Cela signifie que, moyennant un certain nombre d'entrées concernant un domaine donné, l'approche pourra être exécutée automatiquement, sans nécessiter d'autres adaptations particulières au domaine considéré. L'approche a ainsi été utilisée pour annoter indépendamment des descriptions de destinations de vacances et des descriptions de films.

Néanmoins, chaque domaine nécessite ses propres entrées. Celles-ci sont détaillées dans la sous-section suivante.

4.1.1 Les entrées de l'approche

L'approche SAUPODOC se base sur trois entrées :

- la liste des concepts utilisés pour annoter, nommés **concepts cibles**,
- un corpus initial de documents, chacun décrivant une entité du domaine. Une partie de ces documents doit être annotée manuellement comme étant des exemples positifs ou négatifs de chaque concept cible,
- une ontologie de domaine.

Ces entrées sont données par le concepteur de l'application. Nous nommons en toute généralité "concepteur", un expert du domaine considéré, ayant aussi des connaissances en Web sémantique. Ainsi, le concepteur doit être capable de fournir l'ontologie du domaine en y incluant les propriétés utiles dans l'élaboration de futures définitions des concepts cibles, d'annoter manuellement les documents du corpus et aussi d'expliciter des correspondances entre l'ontologie du domaine et des ressources externes. Dans la

réalité, ce concepteur peut être une seule personne ou bien un groupe de personnes. Dans le cas d'un groupe de personnes, les compétences cumulées de celles-ci doivent permettre de fournir les entrées demandées par l'approche.

Au cours du temps, de nouveaux documents peuvent venir élargir le corpus initial. De même, la liste des concepts utilisés pour annoter peut être élargie. Dans de tels cas, l'approche n'a pas à être relancée entièrement. Les différentes étapes à réaliser sont présentées dans la Section 4.1.2.

Les concepts cibles

Les concepts cibles sont les concepts avec lesquels on souhaite faire les annotations. Ce ne sont au départ que de simples noms de concept qui caractérisent a priori les entités décrites dans les documents, tels que "Destination où l'on peut pratiquer des sports aquatiques en hiver", donnés par le concepteur et dont les définitions doivent être apprises.

Le corpus de documents

Il s'agit de documents XML décrivant chacun une entité du domaine. Ils sont très peu structurés. Leur structure met en avant le nom de l'entité et sa description textuelle. Les descriptions textuelles ne contiennent pas les noms des concepts cibles, mais mentionnent les principales caractéristiques des entités décrites. De ce fait, elles contiennent des termes de la terminologie du domaine, représentée dans l'ontologie.

Dans notre contexte, les documents sont en général extraits de catalogues publicitaires, vantant les mérites des entités décrites. Dans tous les cas, ils décrivent les caractéristiques principales des entités et ne contiennent pas ou peu d'expressions négatives.

L'approche que nous proposons nécessite qu'une partie des documents du corpus soit manuellement annotée par le concepteur. En d'autres termes, ces documents doivent avoir été désignés pour chaque concept cible comme étant un exemple positif ou bien négatif de ce concept cible. Contrairement à ce qu'on pourrait penser, ce processus ne prend pas forcément beaucoup de temps puisque le concepteur annote un document dans son ensemble, plus précisément, il caractérise l'entité décrite dans le document. Il n'annote pas des parties de documents et n'est donc obligé ni de l'analyser, ni même forcément de le lire entièrement. En effet, étant expert du domaine, le concepteur peut éventuellement fournir les annotations en se basant sur ses propres connaissances. Il n'a bien sûr pas non plus à rechercher d'informations complémentaires dans des ressources externes pour faire ses annotations. Une perspective intéressante à ce travail pourrait être d'obtenir ces annotations via un processus de crowdsourcing.

L'ontologie de domaine

L'ontologie définit le domaine. Elle contient tous les éléments définissant les concepts du domaine d'application. C'est un guide pour analyser les documents, pour rechercher dans des ressources externes les informations manquantes dans les documents et pour raisonner avec des définitions. Elle est propre à un seul domaine mais est indépendante de l'approche. En effet, les seules contraintes imposées par l'approche sont décrites dans cette section. Ainsi, l'ontologie peut être en grande partie réutilisée, ou bien (semi-)automatiquement construite. Sa construction n'est pas une préoccupation dans cette thèse.

Plus formellement, l'ontologie $\mathcal{O}$ est une ontologie OWL définie comme un tuple ($\mathcal{C}$, $\mathcal{P}$, $\mathcal{I}$, $\mathcal{A}$, $\mathcal{F}$) où

- $\mathcal{C}$ est un ensemble de classes,
- $\mathcal{P}$ est un ensemble de propriétés caractérisant les classes (propriétés datatype, objet et d'annotation),
- $\mathcal{I}$ est un ensemble d'individus,
- $\mathcal{A}$ est un ensemble d'axiomes incluant des contraintes sur les classes et les propriétés.
- $\mathcal{F}$ est un ensemble de faits (assertions) incluant les typages des individus et des assertions de propriété. Une assertion de propriété est un triplet $<s, p, o>$ qui relie un individu s à un autre individu ou un littéral o via une propriété p de $\mathcal{O}$. Par exemple, si <Destination, hasActivity, Activity> est un axiome dans $\mathcal{A}$, et si d et a sont respectivement des instances de Destination et d'Activity, alors $<d$, hasActivity, $a>$ peut être une assertion de propriété.

La Figure 4.1 montre un extrait de l'ontologie donnée en entrée dans le domaine des destinations de vacances. Les classes Activity, Environment et FamilyType sont respectivement les racines d'une hiérarchie. Par exemple, Environment représente l'environnement naturel (aquatique, désertique, etc.) ou ses qualités (beauté, vue). Certaines propriétés objets représentées sur la figure ont des sous-propriétés, non représentées ici. Les propriétés datatype sont représentées sous la classe qu'elles ont pour domaine. Les individus ne sont pas représentés sur la figure.

$\mathcal{C}$ regroupe deux types de classes. ***La classe principale*** correspond au type général des entités décrites dans le corpus, e.g., Destination. ***Les classes descriptives*** sont toutes les autres classes, utiles pour définir la classe principale, par exemple, Activity. Les concepts cibles seront ultérieurement introduits comme des spécialisations de la classe principale.

$\mathcal{P}$ est l'ensemble des propriétés. Les propriétés objet et datatype caractérisent les classes. Cet ensemble contient aussi des propriétés d'annotation, comme "rdfs:label" ou encore "rdfs:isDefinedBy".

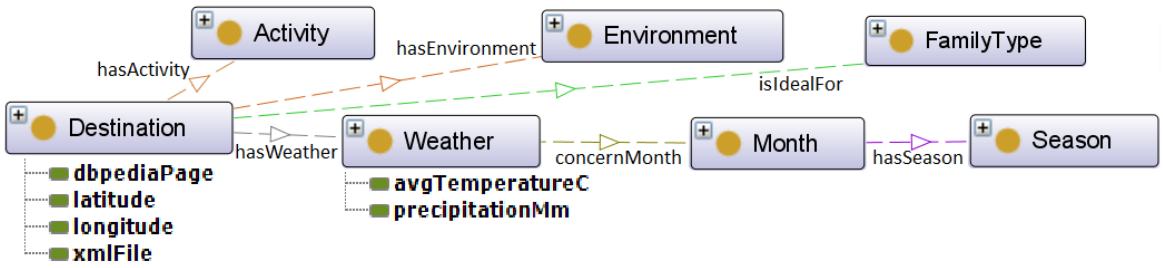


Fig. 4.1 La structure de l'ontologie des destinations

$\mathcal{I}$ contient initialement uniquement des individus qui sont instances de classes descriptives, par exemple _rainForest est un individu.

$\mathcal{A}$ contient initialement des axiomes propres au domaine qui vont permettre d'exprimer la subsomption entre les classes ou les propriétés, le domaine et le co-domaine des propriétés, les caractéristiques des propriétés (fonctionnelle, transitive, etc.), et éventuellement des disjonctions.

$\mathcal{F}$ contient initialement les typages des individus (typés comme étant des instances de classes descriptives). Par exemple, _rainForest est définie comme une instance de Forest (descendant de Environment). Il peut y avoir aussi initialement certaines assertions de propriétés objet pour définir le domaine. Par exemple, l'individu January (instance de Month) est lié à la saison MidWinter (instance de Season) par la propriété hasSeason. $\mathcal{F}$ contient aussi des assertions de propriétés d'annotation permettant de lier un élément ontologique à sa terminologie (labels, mots ou expressions clés). Nous appelons **terminologie de l'ontologie** l'ensemble des labels et expressions reliés par les propriétés "label" et "isDefinedBy" à un individu de l'ontologie. Par exemple, la terminologie de l'individu _rainForest est composée de trois labels : dense forest, rain forest et tropical forest.

Nous distinguons dans l'ontologie deux types de propriétés, que nous nommons respectivement ***propriétés des documents*** et ***propriétés externes***. Les *propriétés des documents* sont les propriétés qui sont explicitées dans les documents tandis que les *propriétés externes* ne le sont pas.

Les documents d'un corpus sont considérés comme complets par rapport aux *propriétés des documents*. Par exemple, les documents décrivant des destinations de vacances mentionnent les activités qui peuvent être pratiquées dans ces destinations. Si une activité n'est pas mentionnée dans un document relatif à une destination, nous pouvons alors considérer qu'elle ne peut être pratiquée dans cette destination.

En revanche, les documents sont incomplets par rapport aux *propriétés externes*. Ces propriétés ne sont (en général) pas mentionnées du tout dans les documents, par exemple dans le corpus de destinations, les données concernant les conditions météorologiques. Ces propriétés sont utilisées de deux façons différentes :

- pour compléter les informations des documents avec des informations non incluses dans les documents, par exemple, des données numériques précises comme les données météorologiques.

- pour éviter les possibles contre-sens lors de l'analyse automatique des documents. Par exemple, on peut choisir d'ignorer les informations contenues dans les documents supposées difficiles à interpréter et faire davantage confiance aux informations issues de ressources externes. Ainsi, dans un document décrivant un film, le terme "French" peut correspondre à la valeur de différentes propriétés. Il peut s'agir d'un film français, ou bien d'un film en langue française ou encore d'un film racontant l'histoire d'un Français. Si on a besoin de connaître ces propriétés, on préférera alors rechercher l'information dans une ressource externe.

Les *propriétés des documents* peuvent être instanciées à partir des documents. En revanche, les *propriétés externes* ont besoin d'être instanciées à partir de données explicitées dans des ressources externes. Dans cette thèse, nous avons choisi de travailler avec des jeux de données issues du LOD (Linked Open Data) aussi appelé Web des données. Le concepteur, indique dans l'ontologie, les *propriétés externes* à peupler et sélectionne les jeux de données du LOD les plus pertinents pour réaliser cette tâche. Cependant, l'ontologie et les jeux de données du LOD n'ont ni le même vocabulaire, ni la même structure. Des correspondances doivent être établies. La forme de ces correspondances, ainsi que l'explicitation de mécanismes permettant de gérer les problèmes liés à l'acquisition des données du LOD sont détaillées dans le Chapitre 7. Comme le nombre de *propriétés externes* n'est pas censé être trop grand, cette tâche peut être faite manuellement.

4.1.2 Description fonctionnelle

La Figure 4.2 décrit le principe de l'approche. Le processus d'annotation est exécuté hors-ligne. Cela signifie que le temps d'exécution de l'approche n'est pas une donnée critique. Pour annoter le corpus avec les concepts cibles, SAUPODOC exploite l'ontologie du domaine qui est progressivement peuplée et enrichie.

L'annotation des documents se ramène à un problème de peuplement et d'enrichissement de l'ontologie. Si une classe de l'ontologie, correspondant à un concept cible *cc*, est peuplée avec une instance, alors le document *d* décrivant cette instance sera annoté par ce concept cible *cc*, sinon il sera annoté par la négation de ce concept cible (*non cc*). Nous appelons *cc* une annotation positive et *non cc* une annotation négative par rapport à un concept cible *cc*.

Le processus de peuplement et d'enrichissement de l'ontologie est décrit dans la Figure 4.3. Il est basé sur quatre tâches guidées par l'ontologie. Les deux premières tâches ont pour but de peupler l'ontologie (étape 1) avec des assertions de propriété. Les deux tâches suivantes (étape 2) sont des tâches de raisonnement. L'étape 2a

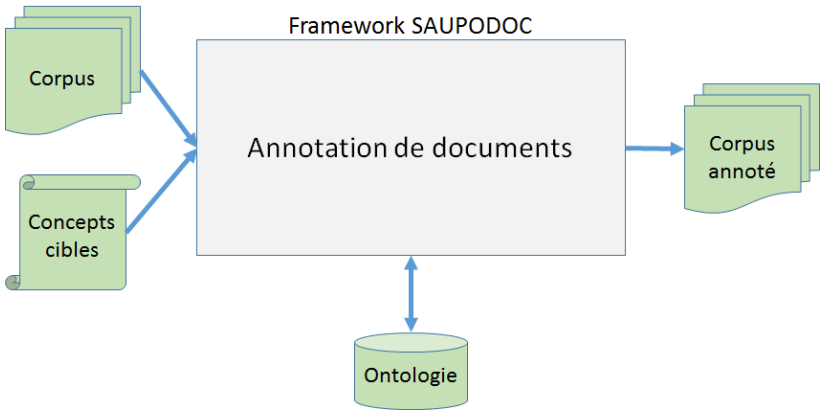


Fig. 4.2 Le framework SAUPODOC

découvre les définitions formelles des concepts cibles tandis que l'étape 2b peuple les classes de l'ontologie qui correspondent à ces définitions. L'ontologie initiale O est donc progressivement peuplée et enrichie tout au long du processus d'annotation. Tout d'abord, l'entité décrite dans chaque document est introduite dans l'ontologie comme une instance de la classe principale. Puis, chaque description textuelle est analysée pour peupler l'ontologie avec les assertions de propriété exprimant les caractéristiques de l'entité. Ces assertions sont ensuite complétées grâce aux informations recherchées dans des ressources externes, plus précisément dans le LOD (O^{+}). Les concepts cibles sont insérés dans l'ontologie comme des classes, appelées classes cibles, qui sont des spécialisations de la classe principale. Leur définition est apprise en se basant sur les documents manuellement annotés (O^{++}). Enfin, les définitions sont appliquées pour peupler les classes cibles (O^{+++}).

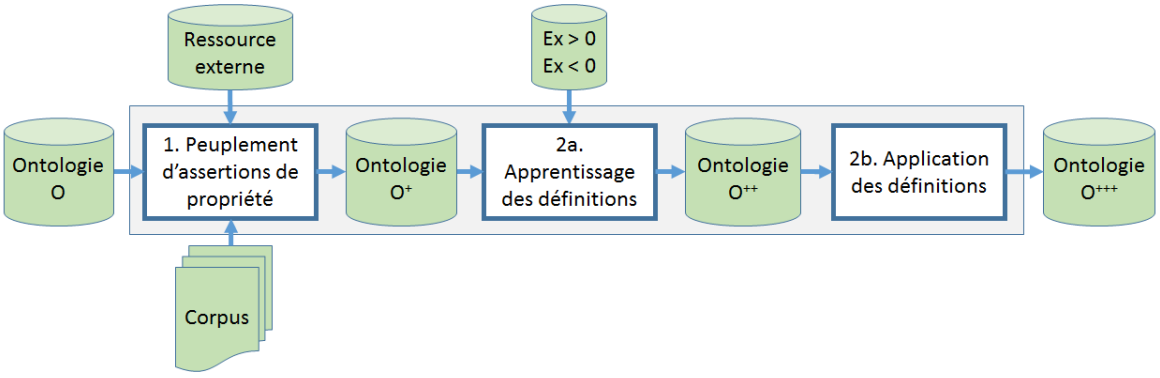


Fig. 4.3 Le workflow de SAUPODOC

Le workflow décrit permet d'annoter les documents d'un corpus. Par ailleurs, il permet d'annoter de nouveaux documents, notamment de nouvelles descriptions de produits issues de nouveaux catalogues fournisseurs du même domaine. Dans ce cas, l'approche n'a pas à être entièrement ré-exécutée. En effet, les définitions des concepts cibles sont déjà connues donc la tâche d'apprentissage de celles-ci n'a

pas à être ré-effectuée. La Figure 4.4 montre les tâches à effectuer dans ce cas et qui exploitent l'ontologie courante. Celle-ci doit être peuplée avec les assertions de propriété caractérisant les nouveaux documents, puis les définitions des concepts cibles sont directement appliquées. Ainsi, l'ontologie obtenue en sortie permet d'annoter les nouveaux documents.

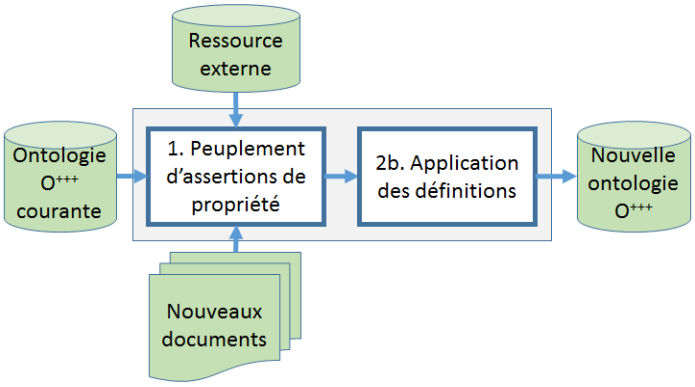


Fig. 4.4 Traitement de nouveaux documents

De même, si de nouveaux concepts cibles viennent s'ajouter, notamment quand le questionnaire de Wepingo évolue, le processus n'a pas à être entièrement relancé, comme le montre la Figure 4.5. Des exemples positifs et négatifs de ces nouveaux concepts cibles doivent être fournis parmi les documents du corpus. Seule l'étape 2 doit être effectuée afin d'apprendre les définitions des nouveaux concepts cibles (et uniquement celles-ci) et de les appliquer. Ainsi, l'ontologie obtenue en sortie permet d'annoter tous les documents avec tous les concepts cibles (les anciens et les nouveaux).

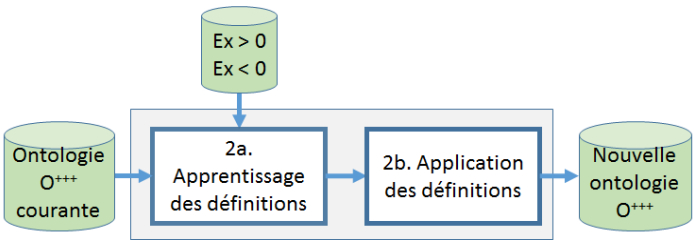


Fig. 4.5 Traitement de nouveaux concepts cibles

En cas de d'annotation de documents de corpus de nouveaux domaines, le worklow décrit en Figure 4.3 s'applique. Les entrées (corpus, concepts cibles, ontologie) doivent être ajustées au nouveau domaine considéré.

4.1.3 Une problématique sous l'hypothèse du monde clos

Une originalité de notre approche ontologique est le fonctionnement de celle-ci sous l'hypothèse du monde clos.

L'hypothèse du monde clos signifie qu'un fait ne peut être que vrai ou faux. Il n'y a pas de faits inconnus. L'absence d'information sur une affirmation est interprétée comme la preuve de sa fausseté. Cette hypothèse s'oppose à celle du monde ouvert dans laquelle on ne peut pas dire qu'un fait n'existe pas tant qu'il n'a pas été explicitement statué qu'il n'existait pas.

Ici, le problème posé suppose l'hypothèse du monde clos. En effet, les documents donnés en entrée décrivent les principales caractéristiques des entités et ces documents sont supposés complets par rapport aux propriétés dites *des documents*. Par exemple, les documents décrivant des destinations mentionnent les activités qui peuvent y être pratiquées. Si une activité n'est pas mentionnée, nous pouvons admettre qu'elle ne peut pas être pratiquée dans cette destination. Ainsi, "Paris Plages" n'est pas mentionné dans la description de Paris. De ce fait, nous considérons qu'il n'y a pas de plages à Paris, ce qui est plutôt raisonnable.

De plus, la reformulation du problème d'annotation en problème de peuplement et d'enrichissement d'ontologie suppose que si nous n'arrivons pas à établir qu'une entité est une instance d'une classe, alors elle n'est pas instance de cette classe, donc nous pouvons annoter l'entité négativement pour cette classe. Nous sommes donc bien sous l'hypothèse du monde clos.

Puisque notre approche se base sur l'hypothèse du monde clos, toutes les étapes devront respecter cette hypothèse. L'extraction d'information à partir des documents respecte cette hypothèse puisque les documents décrivent les principales caractéristiques des entités. Pour la complétion à l'aide de ressources externes, nous utilisons des ressources potentiellement incomplètes. Le Chapitre 7 explique comment nous avons géré l'incomplétude pour être cohérent avec notre hypothèse du monde clos. Le raisonnement sur les définitions est aussi sous l'hypothèse du monde clos (cf. sous-sections 4.2.2 et 4.2.2). Le fonctionnement de la totalité de l'approche sous l'hypothèse du monde clos nous permet de considérer que si une entité n'est pas instance d'une classe cible, alors elle est instance de la négation de cette classe cible.

La section suivante détaille les différentes tâches de l'approche.

4.2 Les tâches de l'approche

Cette section présente les différentes tâches de l'approche, qui exploitent les données à différent niveaux d'abstraction (classes et individus) et qui coopèrent via l'ontologie pour atteindre le but final. Le rôle de pivot de l'ontologie est central dans cette approche. En effet, celle-ci est un support permettant d'intégrer les différentes informations des différentes tâches. Les tâches, décrites dans cette partie, sont :

- une tâche préliminaire de peuplement de l'ontologie, qui représente chacune des entités décrites dans les documents du corpus comme des instances de la classe principale de l'ontologie (Étape 1),
- une tâche d'extraction de données à partir des documents (peuplement de l'ontologie avec des assertions de propriété), nommée tâche 1.a (Étape 1),
- une tâche de complétion de données à partir de ressources externes (peuplement de l'ontologie avec des assertions de propriété), nommée tâche 1.b (Étape 1),
- une tâche d'apprentissage de définitions des concepts cibles (enrichissement de l'ontologie avec les concepts cibles ajoutés comme des sous-classes de la classe principale et avec des d'axiomes d'équivalence exprimant la définition de ceux-ci) nommée tâche 2.a (Étape 2),
- une tâche de raisonnement sur les définitions apprises, permettant ainsi de peupler les classes cibles, nommée tâche 2.b (Étape 2).

4.2.1 Étape 1 : Extraction de données

La figure 4.6 récapitule l'étape 1 de l'approche. Celle-ci est composée de trois tâches : la tâche préliminaire qui crée des instances de la classe principale représentant les entités décrites dans les documents, la tâche 1.a qui extrait des données à partir des documents et la tâche 1.b qui complète ces données avec des ressources externes. Les informations recueillies par les deux dernières tâches s'ajoutent à l'ontologie sous la forme d'assertions de propriété. L'étape 1 s'effectue à chaque fois que de nouveaux documents doivent être annotés.

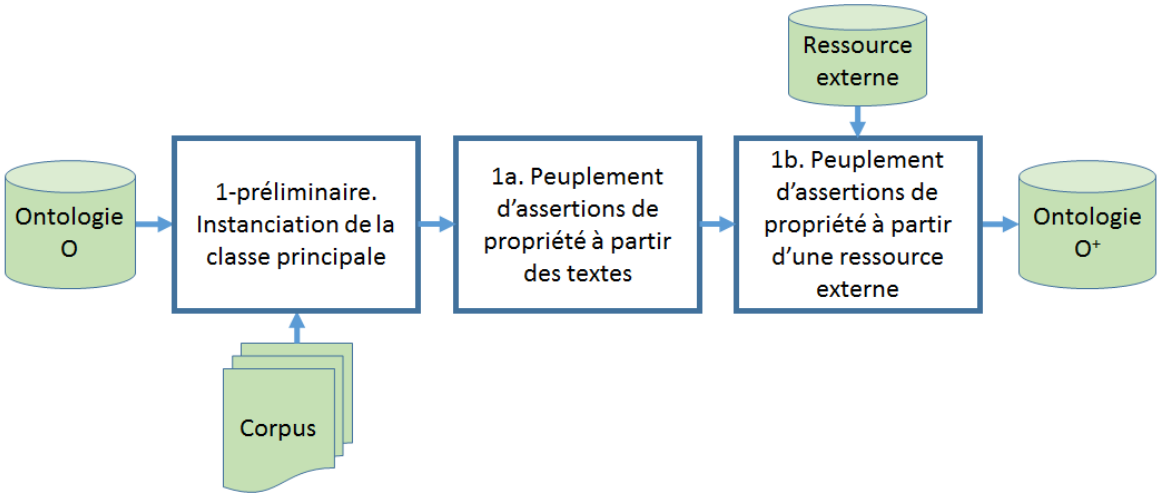


Fig. 4.6 L'étape d'extraction de données (étape 1)

Tâche préliminaire : création d'instances de la classe principale

La tâche préliminaire crée, pour chaque document, une instance de la classe principale représentant l'entité décrite dans le document considéré. Par exemple, dans le domaine des destinations, l'individu Dominican_Republic est créé à partir du document décrivant la République dominicaine. Cet individu est créé en tant qu'instance de la classe principale Destination : <Dominican_Republic rdf:type Destination>. Pour chaque entité et donc chaque individu instance de la classe principale, les deux tâches d'extraction décrites ci-dessous peuplent l'ontologie avec des informations qui seront utilisées dans l'étape 2.

Tâche 1.a : Extraction de données à partir des documents

La première tâche d'extraction de données de l'étape 1 extrait des données des documents. Son but est de peupler l'ontologie avec des assertions relatives aux propriétés dites *propriétés des documents*. Cette extraction est guidée à la fois par la terminologie de l'ontologie et par les contraintes exprimées dans l'ontologie sur les co-domaines des *propriétés des documents*.

Par exemple, pour la propriété hasActivity, la contrainte <Destination, hasActivity, Activity> requiert que la valeur de co-domaine de la propriété hasActivity appartienne à l'extension de la classe Activity, c'est-à-dire qu'une assertion de hasActivity doit avoir pour co-domaine une instance de la classe Activity (ou d'une de ses classes descendantes). À partir de cette contrainte, si le document décrivant une entité e contient des termes liés à une instance a de Activity, alors l'assertion <e, hasActivity, a> est construite. La Figure 4.7 représente un extrait d'un document décrivant la République dominicaine. Les expressions scuba divers and diving sont des termes dénotant l'individu _diving, qui est une instance d'une sous-classe de la classe Activity. Donc l'assertion <Dominican_Republic, hasActivity, _diving> est ajoutée. Notons que dans l'ensemble de nos expérimentations, nous avons spécialisé les co-domaines de propriété pour éviter d'avoir deux propriétés (de type *propriétés des documents*) avec le même co-domaine. L'approche devrait être étendue dans des travaux futurs pour prendre en compte les cas où une telle spécialisation est impossible.

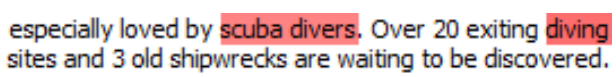


Fig. 4.7 Extrait du document sur la République dominicaine annoté par GATE

Pour effectuer cette tâche, nous utilisons le logiciel d'annotation GATE [Bontcheva *et al.* 2004, Cunningham *et al.* 2011], un logiciel open source capable d'effectuer beaucoup de tâches utiles en analyse de textes. GATE a été choisi pour sa capacité à utiliser une ontologie fournie en entrée, contrairement à d'autres outils comme Open Calais[1] qui reconnaît des entités nommées, des faits ou des évènements dans les documents mais ne peut pas être utilisé avec une ontologie

[1]http://www.opencalais.com/

externe.

La ressource GATE OntoRoot Gazetteer, combinée avec d'autres ressources génériques de GATE, peut produire des appariements, appelés **lookups**, entre des documents textuels et les éléments d'une ontologie donnée en entrée. Ce paragraphe expose brièvement[2] le fonctionnement de GATE pour obtenir des lookups à partir d'une ontologie et d'un corpus de documents. GATE s'appuie sur les ressources ontologiques (classes, instances, propriétés) de l'ontologie fournie en entrée. Les noms des ressources ontologiques (fragments identificateurs des IRI ou bien valeurs des propriétés telles que le label par exemple) sont extraits de l'ontologie et insérés dans une liste. Un pré-traitement a lieu au préalable (remplacement, entre-autres, des caractères "-" et "_" par " "). Chaque élément de la liste créée est analysé par une application Onto Root composée de l'enchaînement de plusieurs ressources GATE (tokeniser, part-of-speech tagger, morphological analyser) et les racines obtenues sont sauvegardées. De même, des ressources GATE sont appliquées sur les documents du corpus pour connaître les racines des mots des textes. Ainsi, des lookups peuvent être trouvés en comparant les termes (racines) de la liste issus des ressources ontologiques et ceux des textes.

La Figure 4.8 montre en rouge les lookups trouvés par GATE sur le document décrivant la République dominicaine. Ces lookups concernent tous les appariements possibles avec n'importe quel élément ontologique : classe, propriété, individu, via leur nom ou leur terminologie associée. Tous ces lookups ne sont pas forcément pertinents, en particulier, ceux concernant les noms de propriété ou de concept ne doivent pas conduire à une assertion de propriété. Par exemple, un lookup entre le mot "activity" dans un document et le concept Activity de l'ontologie ne sera pas considéré comme pertinent puisqu'il ne peut pas conduire à une assertion de propriété.

GATE peut être utilisé avec un traducteur JAPE, qui applique des règles JAPE (Java Annotation Patterns Engine). Dans notre contexte, les règles JAPE transforment les lookups qui nous intéressent, c'est-à-dire ceux portant sur la terminologie des individus instances de classes descriptives, en assertions de propriété.

Les règles JAPE utilisées sont automatiquement construites à partir d'un patron, utilisable pour n'importe quelle ontologie donnée en entrée. Ce patron est décrit dans l'annexe A. L'idée du patron est de parcourir les lookups. La Figure 4.9 montre l'heuristique exprimée dans le patron générique : si un lookup concerne un individu i et que cet individu i est une instance d'une classe c (ou des descendants d'une classe c) qui est un co-domaine d'une propriété *prop* dite *propriété des documents*, alors on crée une assertion entre l'entité e du document concerné, la propriété concernée *prop* et l'individu concerné i, i.e., $<e$, *prop*, $i>$. Le patron JAPE générique est instancié pour chaque propriété dite *propriété des documents*, créant ainsi autant de règles JAPE que

[2]Pour plus de détails, voir https://gate.ac.uk/sale/tao/splitch13.html#x18-34000013.8

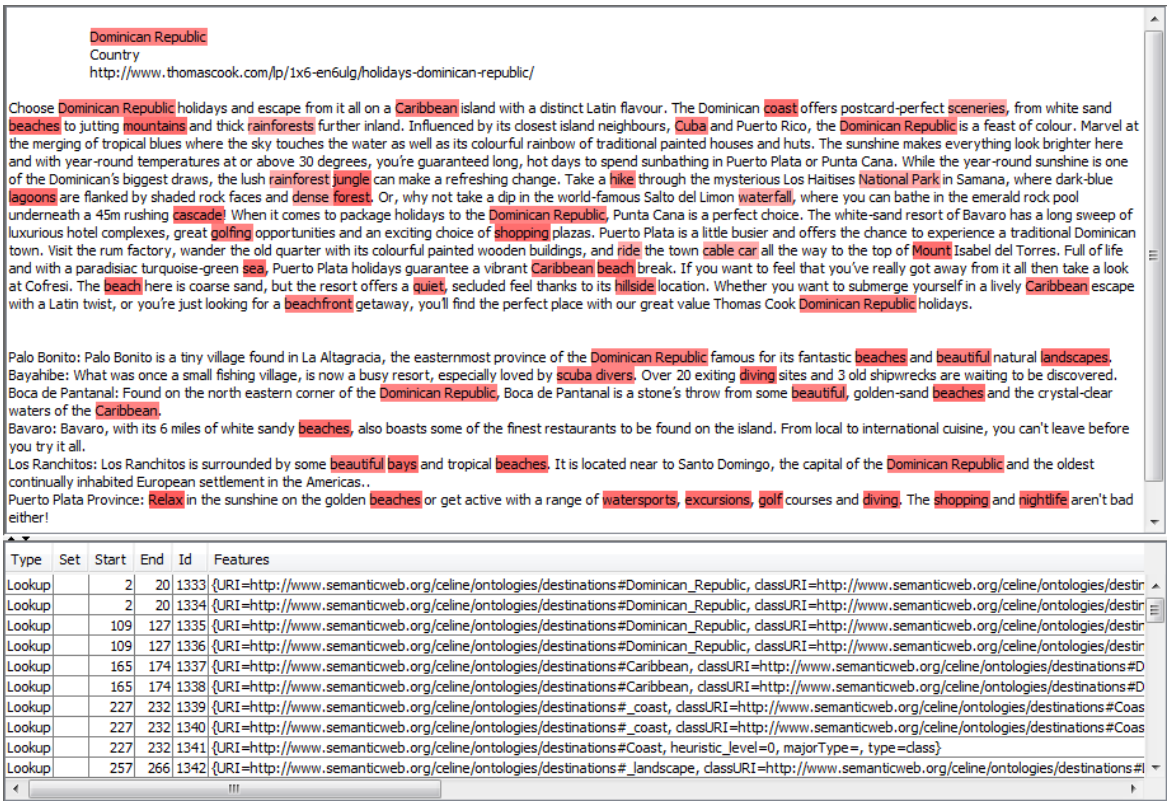


Fig. 4.8 Lookups trouvés par GATE sur le document traitant de la République dominicaine

de propriétés des documents. La Figure 4.10 montre des exemples de ce patron instancié.

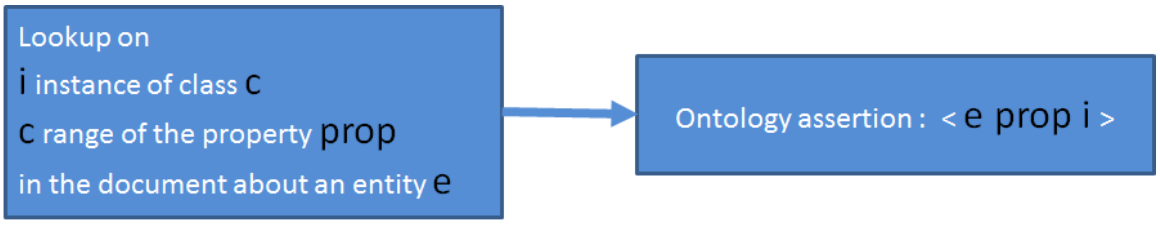


Fig. 4.9 L'heuristique exprimée dans le patron JAPE générique

Un exemple des assertions obtenues pour l'entité "Dominican_Republic" est donné dans la Figure. 4.11. Notons que nous sommes dans un contexte où les documents décrivent les principales caractéristiques des entités et n'incluent pas d'expressions négatives qui pourraient perturber le processus. Ainsi, une tâche d'extraction d'informations telle que celle-ci est appropriée. Si nous devions étendre le contexte à d'autres types de documents, des améliorations seraient à faire sur cette tâche, mais cela ne remettrait pas en cause l'approche globale.

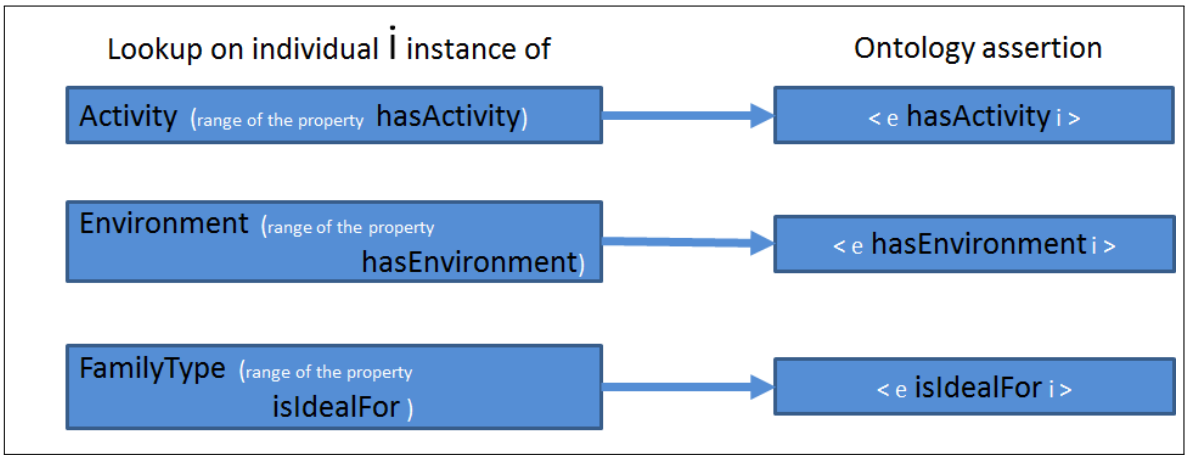


Fig. 4.10 Exemples de règles JAPE obtenues grâce au patron JAPE générique (sur l'ontologie des destinations)

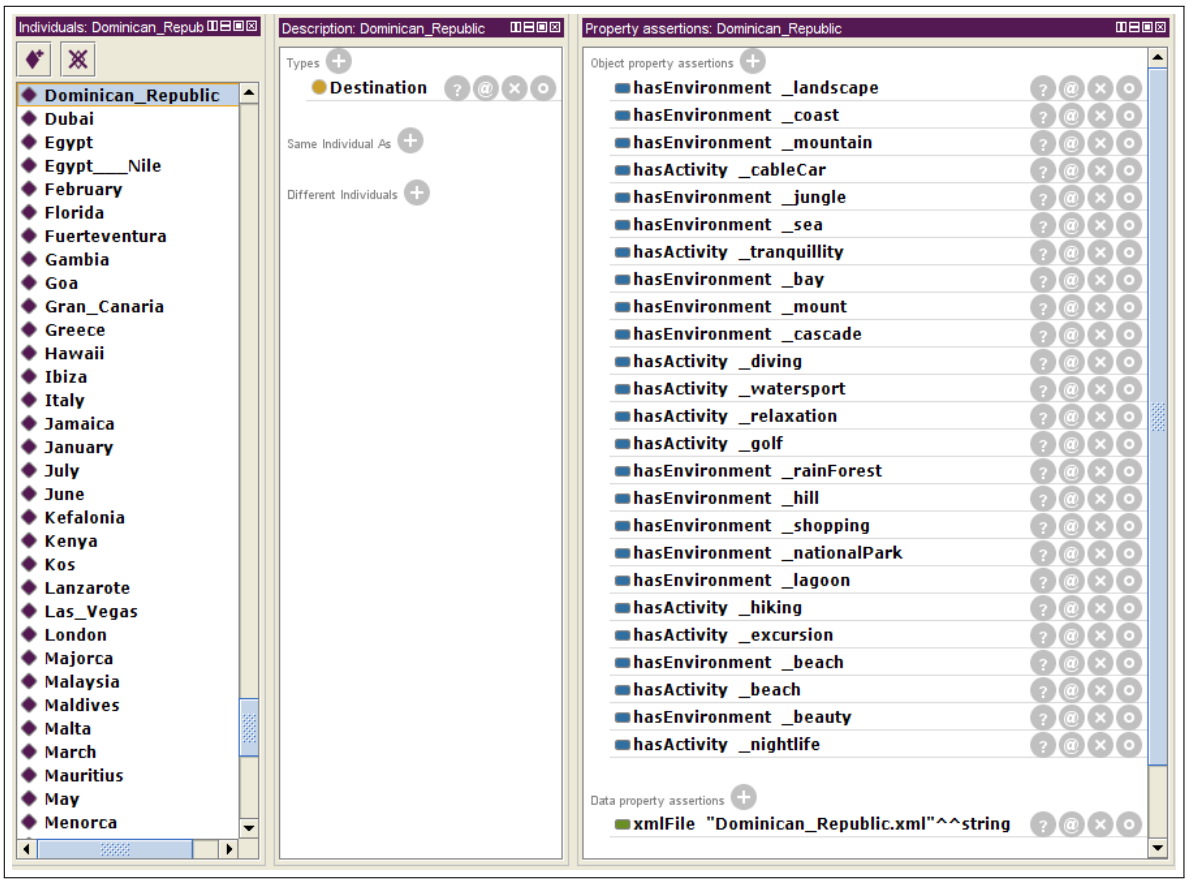


Fig. 4.11 Les assertions obtenues pour la République dominicaine

Tâche 1.b : Complétion des données avec des ressources externes

Les descriptions textuelles sont souvent courtes et ne contiennent pas toutes les informations nécessaires. Par exemple, définir une "destination où l'on peut pratiquer des sports aquatiques en hiver" demande de connaître les températures et les précipitations par saison ou par mois pour chaque destination. Ces données n'apparaissant pas dans les descriptions, la collecte doit être enrichie en exploitant les ressources disponibles sur le Web. Ici encore, cette tâche est guidée par l'ontologie. Elle implique de trouver une ressource RDF relative aux entités du corpus et d'identifier dans cette ressource quelles données correspondent à celles requises par l'ontologie (les *propriétés externes*).

Nous avons choisi de travailler avec DBpedia [Auer *et al.* 2008]. Cette tâche est composée de deux parties. La première permet d'associer à un document la page DBpedia décrivant l'entité du document. Pour cela, nous utilisons DBpedia Spotlight [Mendes *et al.* 2011, Daiber *et al.* 2013], un outil capable d'annoter automatiquement un texte avec des références aux entités de DBpedia. Appliqué sur le nom de l'entité, il donne un accès direct à la ressource DBpedia correspondant à l'entité de chaque document, contrairement à d'autres outils tels que Wikifier [Cheng & Roth 2013, Ratinov *et al.* 2011] ou AIDA [Yosef *et al.* 2011], qui retournent des pages Wikipédia. La Figure 4.12 montre que DBpedia Spotlight est capable d'associer le nom de l'entité "Dominican Republic" à sa page DBpedia.

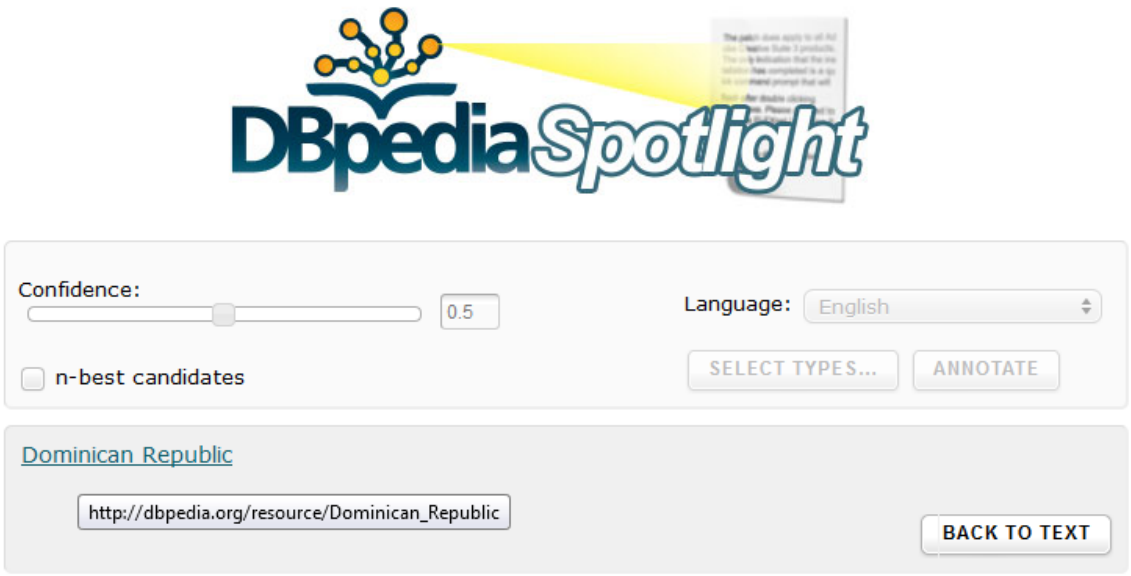


Fig. 4.12 DBpedia Spotlight utilisé sur le document relatif à République dominicaine

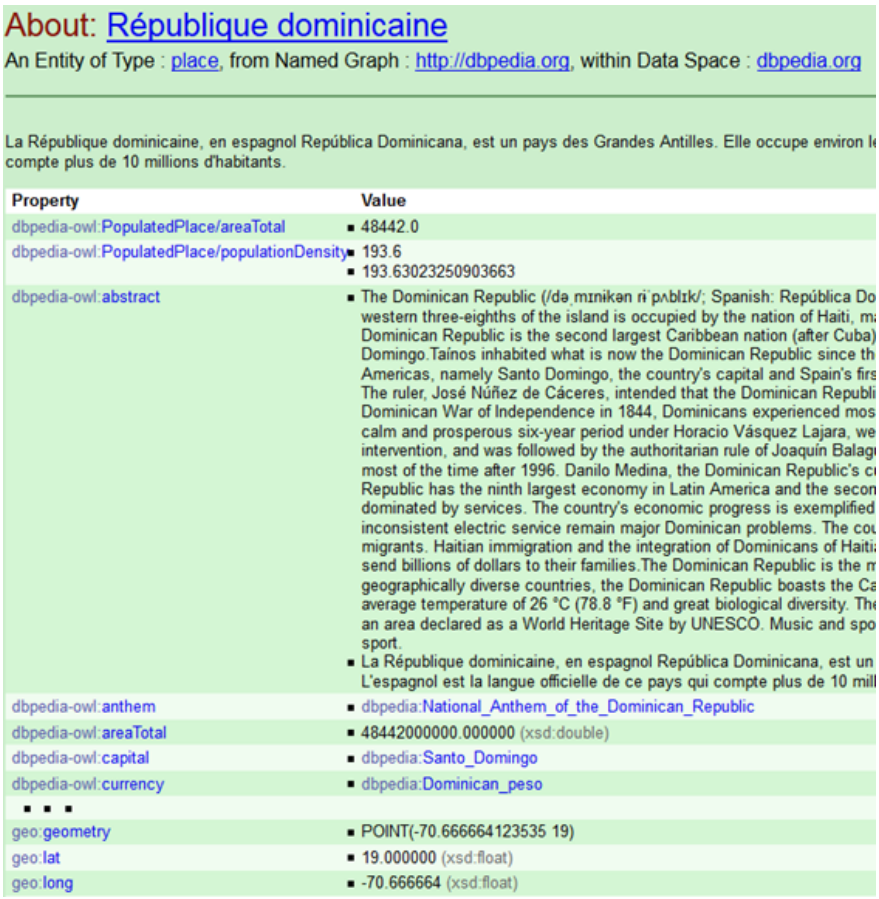
La seconde partie de la tâche consiste à aller chercher dans la page DBpedia trouvée les informations manquantes dans le document et à les mettre dans l'ontologie. Par

exemple, la Figure 4.13 montre un extrait des informations disponibles dans la page DBpedia décrivant la République dominicaine. On peut y trouver des informations géographiques telles que sa latitude et sa longitude mais il n'y a pas d'informations météorologiques. Sur notre corpus de 80 destinations, seules 29 d'entre elles ont des informations météorologiques dans DBpedia. En effet, DBpedia est une ressource incomplète. C'est d'ailleurs le cas du LOD en général. Or, notre approche fonctionne sous l'hypothèse du monde clos. Cette hypothèse ne doit pas s'appliquer ici : le fait que des données météorologiques ne soient pas présentes sur une page ne signifie pas qu'elles n'existent pas. Pour pallier l'incomplétude de DBpedia, nous allons chercher à obtenir une valeur pour ces données manquantes, qui peut être une approximation. Par exemple, la page de la République dominicaine (Figure 4.13) pointe sur la page de Saint-Domingue, sa capitale. Or, la page de Saint-Domingue contient des données météorologiques, cf. Figure 4.14, qui peuvent être de bonnes approximations pour celles de la République dominicaine. C'est sur cette idée qu'est basée cette seconde partie de la tâche. Le concepteur fournit en entrée des correspondances entre les propriétés de l'ontologie et des propriétés de DBpedia. Il fournit aussi une liste ordonnée de chemins à parcourir en cas de valeurs manquantes (tels que de passer par la capitale dans le cas de la météo). Pour fournir ces entrées, le concepteur se base sur le modèle d'acquisition donné dans le Chapitre 7. Ce chapitre explique comment doivent être exprimées les correspondances entre des propriétés ontologiques et des propriétés de jeux de données du LOD, ainsi que des chemins d'accès pour gérer l'incomplétude.

Le modèle d'acquisition est actuellement utilisé pour extraire les informations de DBpedia, et pourra dans un travail futur être utilisé pour d'autres jeux de données. Ce modèle suppose qu'on connaisse les équivalences entre les individus du domaine des propriétés considérées. De ce fait, il suppose dans notre cas que l'entité décrite dans un document a un individu équivalent dans DBpedia. Cette équivalence est actuellement trouvée grâce à DBpedia Spotlight.

À partir de ce modèle d'acquisition, des requêtes SPARQL (de type CONSTRUCT) sont générées automatiquement, grâce à un ensemble de patrons donnés dans le Chapitre 8. Ces requêtes permettent de construire dans notre ontologie les assertions de propriété basées sur les données disponibles dans DBpedia. Pour faire cela, le point d'accès SPARQL de DBpedia[3] est utilisé.

[3]http://dbpedia.org/sparql



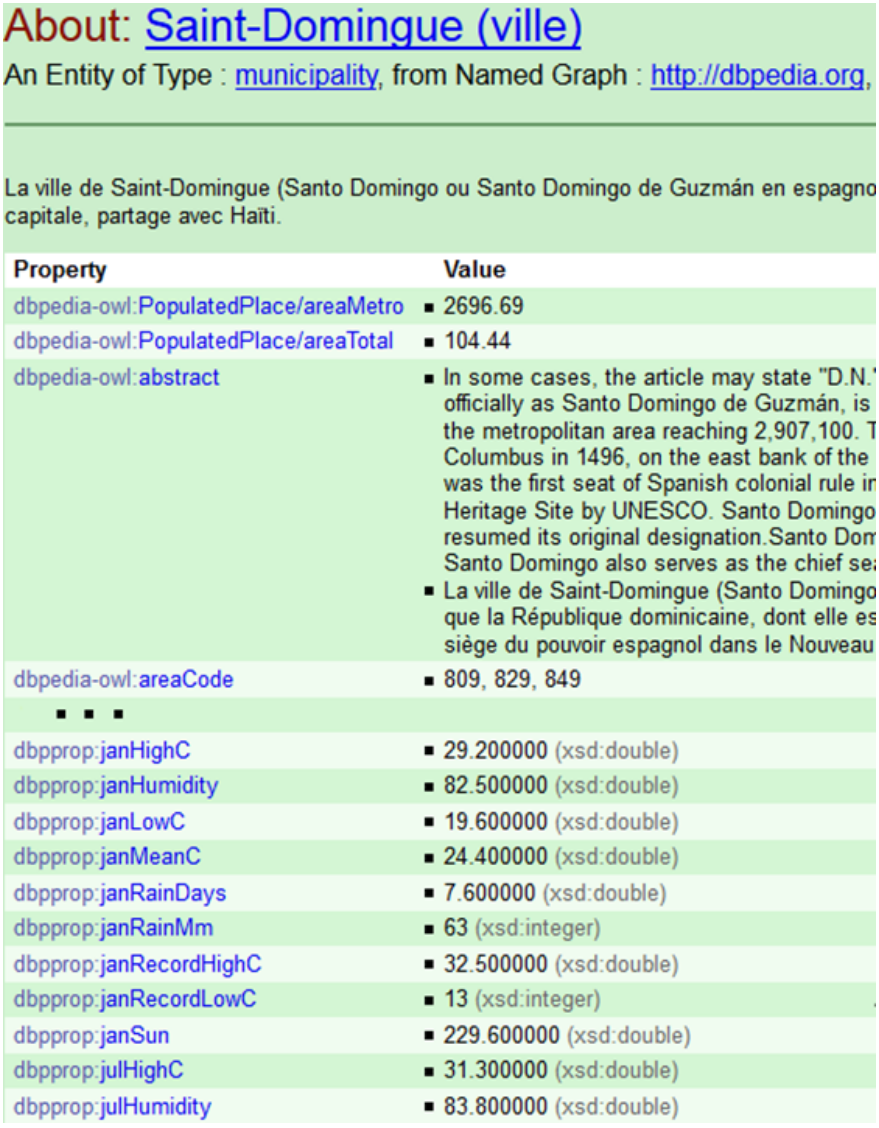
About: République dominicaine

An Entity of Type : place, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

La République dominicaine, en espagnol República Dominicana, est un pays des Grandes Antilles. Elle occupe environ l
compte plus de 10 millions d'habitants.

Property	Value
dbpedia-owl:PopulatedPlace/areaTotal	▪ 48442.0
dbpedia-owl:PopulatedPlace/populationDensity	▪ 193.6 ▪ 193.63023250903663
dbpedia-owl:abstract	▪ The Dominican Republic (/dəˌmɪnikən rɪˈpʌblɪk/; Spanish: República Do western three-eighths of the island is occupied by the nation of Haiti, m Dominican Republic is the second largest Caribbean nation (after Cuba) Domingo.Taínos inhabited what is now the Dominican Republic since th Americas, namely Santo Domingo, the country's capital and Spain's firs The ruler, José Núñez de Cáceres, intended that the Dominican Republi Dominican War of Independence in 1844, Dominicans experienced mos calm and prosperous six-year period under Horacio Vásquez Lajara, we intervention, and was followed by the authoritarian rule of Joaquín Balag most of the time after 1996. Danilo Medina, the Dominican Republic's c Republic has the ninth largest economy in Latin America and the secon dominated by services. The country's economic progress is exemplified inconsistent electric service remain major Dominican problems. The cou migrants. Haitian immigration and the integration of Dominicans of Haiti send billions of dollars to their families.The Dominican Republic is the n geographically diverse countries, the Dominican Republic boasts the Ca average temperature of 26 °C (78.8 °F) and great biological diversity. The an area declared as a World Heritage Site by UNESCO. Music and spo sport. ▪ La République dominicaine, en espagnol República Dominicana, est un L'espagnol est la langue officielle de ce pays qui compte plus de 10 mill
dbpedia-owl:anthem	▪ dbpedia:National_Anthem_of_the_Dominican_Republic
dbpedia-owl:areaTotal	▪ 48442000000.000000 (xsd:double)
dbpedia-owl:capital	▪ dbpedia:Santo_Domingo
dbpedia-owl:currency	▪ dbpedia:Dominican_peso
▪ ▪ ▪	
geo:geometry	▪ POINT(-70.666664123535 19)
geo:lat	▪ 19.000000 (xsd:float)
geo:long	▪ -70.666664 (xsd:float)

Fig. 4.13 Extrait de la page DBpedia représentant la République dominicaine



About: Saint-Domingue (ville)

An Entity of Type : municipality, from Named Graph : http://dbpedia.org,

La ville de Saint-Domingue (Santo Domingo ou Santo Domingo de Guzmán en espagno capitale, partage avec Haïti.

Property	Value
dbpedia-owl:PopulatedPlace/areaMetro	▪ 2696.69
dbpedia-owl:PopulatedPlace/areaTotal	▪ 104.44
dbpedia-owl:abstract	▪ In some cases, the article may state "D.N. officially as Santo Domingo de Guzmán, is the metropolitan area reaching 2,907,100. T Columbus in 1496, on the east bank of the was the first seat of Spanish colonial rule in Heritage Site by UNESCO. Santo Domingo resumed its original designation.Santo Dom Santo Domingo also serves as the chief sea ▪ La ville de Saint-Domingue (Santo Domingo que la République dominicaine, dont elle es siège du pouvoir espagnol dans le Nouveau
dbpedia-owl:areaCode	▪ 809, 829, 849
▪ ▪ ▪	
dbpprop:janHighC	▪ 29.200000 (xsd:double)
dbpprop:janHumidity	▪ 82.500000 (xsd:double)
dbpprop:janLowC	▪ 19.600000 (xsd:double)
dbpprop:janMeanC	▪ 24.400000 (xsd:double)
dbpprop:janRainDays	▪ 7.600000 (xsd:double)
dbpprop:janRainMm	▪ 63 (xsd:integer)
dbpprop:janRecordHighC	▪ 32.500000 (xsd:double)
dbpprop:janRecordLowC	▪ 13 (xsd:integer)
dbpprop:janSun	▪ 229.600000 (xsd:double)
dbpprop:julHighC	▪ 31.300000 (xsd:double)
dbpprop:julHumidity	▪ 83.800000 (xsd:double)

Fig. 4.14 Extrait de la page DBpedia représentant Saint Domingue

4.2.2 Étape 2 : Raisonnement sur l'ontologie peuplée

La figure 4.15 récapitule l'étape 2 de l'approche. Celle-ci est composée de deux tâches : la tâche 2.a apprend des définitions de concepts cibles et les ajoute dans l'ontologie tandis que la tâche 2.b applique ces définitions pour peupler les classes cibles et annoter les documents.

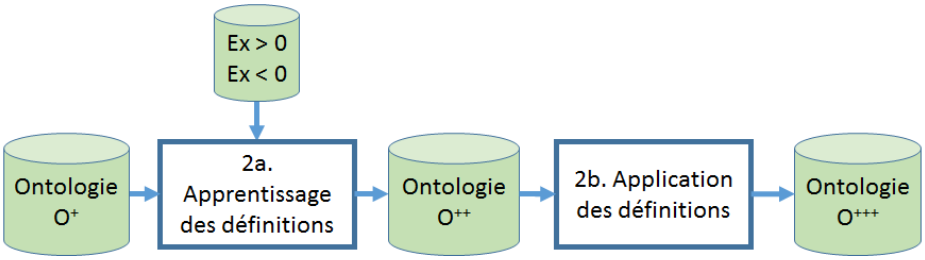


Fig. 4.15 L'étape 2 de l'approche

Tâche 2.a : Apprentissage des définitions de concepts cibles

La première tâche de l'étape 2 est une tâche de raisonnement qui est exécutée lorsque de nouveaux concepts cibles sont à prendre en compte. Elle a pour but d'apprendre les définitions de concepts cibles, qu'on appellera **définitions cibles**, en se basant sur les annotations manuelles des documents fournies par le concepteur et les données collectées dans l'étape 1 (assertions de propriété sur les entités décrites telles que dans la Figure 4.11). Au cours de cette tâche, chaque concept cible est introduit sous la forme d'une classe (nommée classe cible) dans l'ontologie comme une sous-classe de la classe principale. Lorsque les définitions des concepts cibles sont apprises, l'ontologie est enrichie par des axiomes d'équivalence entre chaque classe cible et sa définition. La définition d'une classe cible est exprimable en OWL-DL. Dans notre approche, elle est générée sous la forme d'une expression en syntaxe OWL Manchester [Horridge *et al.* 2006]. Par exemple, supposons qu'on considère le concept cible "Destination culturelle" et que la définition apprise correspond à hasActivity min 4 Culture (c'est-à-dire correspond à tout individu de l'ontologie qui est lié à au moins 4 activités culturelles). Dans ce cas, la classe "Destination culturelle" est ajoutée dans l'ontologie comme une sous-classe de la classe principale, ici Destination. Puis, un axiome d'équivalence est ajouté exprimant l'équivalence entre la classe Destination culturelle et l'expression hasActivity min 4 Culture.

La plupart des outils d'apprentissage automatique ne prennent pas en compte les relations explicitées dans une ontologie (subsomption, propriétés établies entre les classes) dans leurs représentations des exemples. À notre connaissance, il existe trois outils effectuant de l'apprentissage de concepts sur des logiques de description. Ces trois outils, YinYang [Esposito *et al.* 2004], DL-FOIL [Fanizzi *et al.* 2008] et DL-Learner [Lehmann 2009], sont basés sur la programmation logique inductive.

Nous avons choisi d'utiliser DL-Learner [Lehmann 2009], car c'est le seul logiciel en libre accès utilisant une ontologie en entrée pour apprendre des définitions de classes exprimées en logique de description, à partir d'exemples fournis. Il nous permet d'obtenir les définitions explicites de chaque concept cible, ce qui est un atout important dans les applications concrètes.

DL-Learner étend la Programmation Logique Inductive aux Logiques de Description. Il s'agit d'un outil d'apprentissage automatique basé sur une ontologie qui permet de résoudre des tâches d'apprentissage supervisé. La tâche qui nous intéresse est l'apprentissage de définitions de classes à partir d'exemples positifs et négatifs : en se basant sur des connaissances d'un domaine exprimées dans une ontologie et sur des exemples positifs et négatifs d'une classe, DL-Learner peut découvrir une définition de cette classe. Pour découvrir cette définition, plusieurs algorithmes ont été développés dans DL-Learner. Ainsi, l'algorithme de raffinement [Lehmann & Hitzler 2007] utilise un algorithme d'apprentissage basé sur des opérateurs de raffinement. Il s'agit d'un algorithme "top-down" i.e. partant d'un concept générique (Thing) et appliquant au fur et à mesure des opérateurs de raffinement pour obtenir une définition de plus en plus spécialisée. Il a par la suite été étendu (support de certains types de propriétés datatype, support des restrictions hasValue, nouvelles heuristiques, etc.). Cet algorithme étendu, nommé CELOE (Class Expression Learning for Ontology Engineering), est annoncé comme le meilleur algorithme d'apprentissage de classe disponible dans DL-Learner.

Les définitions construites par DL-Learner sont exprimées en syntaxe OWL Manchester [Horridge *et al.* 2006], une syntaxe compacte et facile à comprendre pour les ontologies OWL. Les constructeurs de cette syntaxe sont explicités dans le Tableau 4.1. Les définitions construites contiennent des conjonctions (and) ou des disjonctions (or) d'éléments. Un élément peut être une classe (Destination) ou une expression concernant une propriété objet au niveau classe (hasActivity some Nightlife), une expression concernant une propriété objet au niveau instance (hasActivity value _diving), une expression concernant une propriété datatype (avgTemperatureC some double[>= 23.0]), ou une contrainte de cardinalité (hasCulture min 3 Culture). Les co-domaines peuvent aussi être des conjonctions ou des disjonctions d'éléments. Ainsi, des définitions assez compliquées peuvent être apprises. Par exemple, on pourrait imaginer la définition de "Destination où l'on peut pratiquer des sports aquatiques pendant l'hiver" comme étant une destination avec des sports aquatiques et où la température des mois d'hiver (janvier et février) est assez élevée et la précipitation assez basse, ce qui donnerait :

```
(Destination and (hasActivity some Watersport)
      and (hasWeather min 2 ((concernMonth some (hasSeason some MidWinter))
                  and (avgTemperatureC some double[>= 23.0])
                  and (precipitationMm some double[<= 70.0])))).
```

OWL	**DL**	**OWL Manchester**	**Exemple**
intersectionOf	C ⊓ D	C **AND** D	Human **AND** Male
unionOf	C ⊔ D	C **OR** D	Man **OR** Woman
complementOf	¬ C	**NOT** C	**NOT** Male
oneOf	{a} ⊔ {b}...	{a, b, ...}	{England, Italy, Spain}
someValuesFrom	∃ R C	R **SOME** C	hasColleague **SOME** Man
allValuesFrom	∀ R C	R **ONLY** C	hasColleague **ONLY** Man
minCardinality	≥ N R C	R **MIN** nb C	hasColleague **MIN** 3 Man
maxCardinality	≤ N R C	R **MAX** nb C	hasColleague **MAX** 3 Man
cardinality	= N R C	R **EXACTLY** nb C	hasColleague **EXACTLY** 3 Man
hasValue	∃ R a	R **VALUE** a	hasColleague **VALUE** Matthew

Tableau 4.1 Les constructeurs de la syntaxe OWL Manchester :
'C' et 'D' désignent des classes simples (nom de concept) ou complexes (expressions), 'R' désigne une propriété, et 'a' un individu. Notons que les propriétés datatype sont aussi considérées : dans ce cas 'C' et 'D' désignent des intervalles de valeurs (ex : double[>= 23.0]) et 'a' désigne une valeur de littéral.

Les paramètres de DL-Learner ont été choisis en se basant sur le manuel utilisateur et à partir de discussions avec les développeurs du logiciel. Nous utilisons l'algorithme CELOE [Lehmann *et al.* 2011], et le raisonneur par défaut, appelé *fast instance checker*, fonctionnant sous l'hypothèse du monde clos (CWA). Cependant, les définitions apprises doivent être incorporées dans une ontologie OWL où le raisonnement est basé sur l'hypothèse du monde ouvert (OWA). De ce fait, un certain nombre d'opérateurs sont désactivés : l'opérateur de négation NOT, l'opérateur de restriction universel ONLY, l'opérateur de cardinalité maximale MAX, l'opérateur de cardinalité exacte EXACTLY.

Ces opérateurs ont été désactivés car inutilisables sous l'hypothèse du monde ouvert. Par exemple, pour l'opérateur NOT, la définition qui contiendrait cet opérateur ne pourrait jamais s'appliquer car un raisonnement en monde ouvert pré-suppose que toutes les informations ne sont pas forcément connues. On ne pourra donc jamais affirmer qu'une certaine propriété n'est pas vérifiée. Il en est de même pour les opérateurs ONLY, MAX et EXACTLY. D'une manière générale, une définition contenant n'importe lequel de ces opérateurs n'engendre jamais d'instances avec un raisonneur fonctionnant sous l'hypothèse du monde ouvert.

De plus, pour être capable d'apprendre et d'exploiter des contraintes de cardinalité minimale, par exemple (hasActivity MIN 3 Activity), les individus de l'ontologie doivent tous être automatiquement exprimés comme étant disjoints (owl:AllDifferent), afin d'interdire au raisonneur OWL de considérer que deux individus peuvent être reliés par un owl:sameAs. En effet, sans cela, un raisonneur OWL n'associerait jamais aucun individu à une définition contenant un MIN. Par exemple, considérons la définition (hasActivity MIN 3 Activity) et un individu i lié à au moins trois activités dont a, b et c. Si leur disjonction n'est pas explicitée, le raisonneur peut supposer que a, b et c sont

potentiellement les mêmes activités (owl:sameAs) et i ne se conformera pas à la définition. En revanche, en explicitant l'Unique Name Assumption (UNA) dans l'ontologie qui assure que deux individus de noms différents sont différents, i se conforme bien à la définition.

En plus de ces paramétrages liés à l'hypothèse du monde clos, nous avons aussi ajusté certains paramètres de DL-Learner. Cet ajustement s'est fait suite à nos discussions avec les développeurs de DL-Learner. Il permet de faire un compromis entre l'expressivité de la définition à trouver et le temps d'exécution de l'outil. Ainsi, nous autorisons les contraintes de cardinalité avec une valeur maximum à 10 au lieu de 5 (valeur par défaut) de sorte que les définitions contenant une expression de la forme `hasObjectProperty min 10 class_name` puissent être apprises. Le temps d'exécution maximum est mis à 200 secondes, une durée suffisamment courte pour faire des expérimentations d'une durée acceptable et suffisamment longue pour trouver des définitions éventuellement complexes. Ces paramètres seront utilisés pour tous les domaines d'application.

Un autre paramètre important est le pourcentage de bruit, c'est-à-dire le pourcentage de bruit approximatif estimé dans les exemples. Nous avons procédé par essais et erreurs pour le fixer. Ainsi, nous avons développé une méthodologie à partir d'expérimentations tests, où nous avons testé différentes valeurs de pourcentage de bruit. Au final, nous avons déduit qu'il était intéressant de tester 5 valeurs différentes pour le pourcentage de bruit (5-15-25-35-45%). En effet, d'autres valeurs ont été testées (comme 10-20-30-40%), mais celles-ci n'ont pas engendré de définitions différentes des premières. De plus, un paramètre d'heuristique de recherche doit être mis en place si on cherche une définition vraiment complexe, c'est-à-dire mettant en jeu beaucoup d'opérateurs. Nous appelons configuration "complexe" l'activation de cette heuristique de recherche. Si cette heuristique est désactivée, nous qualifions la configuration de "basique".

Notre méthodologie consiste donc à tester 10 lancements de DL-Learner pour chaque concept cible : les configurations basiques et complexes, chacune avec les 5 valeurs du pourcentage de bruit. Pour chaque lancement, DL-Learner retourne la meilleure solution, c'est-à-dire celle avec la meilleure exactitude[4]. Pour chaque concept cible, la meilleure définition (celle qui a la meilleure exactitude puis la meilleure F-mesure[5] en cas d'égalité d'exactitude) des 10 lancements est choisie. Un axiome d'équivalence entre la classe cible et cette définition est ajouté dans l'ontologie.

[4]L'exactitude (accuracy) est le pourcentage d'éléments bien classés. Une formule mathématique est donnée, voir équation 5.4 page 68.

[5]La F-mesure est une sorte de moyenne combinant la précision et le rappel. Les formules mathématiques de ces trois mesures sont données, voir équations 5.5, 5.6 et 5.7 page 68.

Par exemple, pour le concept cible "Destination très culturelle", les résultats des 10 lancements sont donnés dans le tableau 4.2. La définition gardée est "hasCulture min 6 Thing" puisqu'elle génère la meilleure exactitude sur l'ensemble de test.

Config.	Bruit	Exactitude	Définition
basique	5	1	hasCulture min 6 Thing
basique	15	1	hasCulture min 6 Thing
basique	25	1	hasCulture min 6 Thing
basique	35	1	hasCulture min 6 Thing
basique	45	1	hasCulture min 6 Thing
complexe	5	1	hasCulture min 6 Thing
complexe	15	0,88	(hasCulture some Sightseeing) and (hasActivity min 5 Culture) and (hasCulture min 6 Thing)
complexe	25	0,96	(isIdealFor value _family) and (hasCulture min 6 Thing)
complexe	35	0,88	(hasCulture some Sightseeing) and (hasCulture min 6 Thing)
complexe	45	0,88	(isIdealFor some WithoutKids) and (hasCulture value _history) and (isIdealFor value _family) and (hasCulture min 6 Thing)

Tableau 4.2 Les 10 lancements de DL-Learner testés pour le concept cible "Destination très culturelle"
L'exactitude est calculée sur l'ensemble de test, i.e., 1/3 de l'échantillon. Pour plus de détails, voir la procédure d'évaluation des expérimentations détaillées dans la Section 5.1 du Chapitre 5 (page 59).

Tâche 2.b : Raisonnement pour peupler des classes cibles et annoter les documents

La seconde tâche de l'étape 2 consiste à appliquer les définitions apprises pour peupler les classes cibles dans l'ontologie. Cette tâche est effectuée à chaque fois que des descriptions doivent être annotées avec des concepts cibles (donc en cas de nouveaux documents ou de nouveaux concepts cibles). Pour appliquer les définitions dans l'ontologie OWL, il faut utiliser un raisonneur OWL. Nous avons choisi d'utiliser FaCT++ [Tsarkov & Horrocks 2006], un raisonneur OWL-DL, qui a l'avantage de fonctionner même en présence d'un grand nombre d'individus, contrairement à HermiT [Shearer *et al.* 2008] et Pellet [Sirin *et al.* 2007] qui ne peuvent pas terminer, d'après nos expérimentations.

FaCT++ se fie aux définitions des concepts cibles dans l'ontologie pour identifier les entités des documents qui doivent être annotées avec ces concepts cibles. Pour chaque concept cible *cc*, si l'entité décrite dans un document *d* se conforme à la

définition de *cc*, alors cette entité devient une instance de la classe représentant *cc*. De cette façon, le document *d* est annoté par *cc*. En revanche, si l'entité ne se conforme pas à la définition de *cc*, alors cette entité ne devient pas une instance de la classe *cc* et le document *d* est annoté par *non cc*.

Comme nous l'avons dit précédemment, toute notre approche fonctionne sous l'hypothèse du monde clos (CWA) tandis que les raisonneurs OWL, tel que FaCT++, fonctionnent sous l'hypothèse du monde ouvert (OWA). Le paramétrage de DL-Learner (la désactivation des opérateurs problématiques) permet d'obtenir des définitions dont l'application sous l'hypothèse du monde ouvert revient au même que l'application sous l'hypothèse du monde clos. De ce fait, même si FaCT++ raisonne sous l'hypothèse du monde ouvert, notre contexte fait qu'on peut voir ce raisonnement comme un raisonnement sous l'hypothèse du monde clos.

Conclusion

Pour conclure, ce chapitre a présenté l'approche Saupodoc, basée sur une ontologie, permettant de faire de l'annotation sémantique sur des documents décrivant des entités d'un même domaine. Les annotations sont faites avec une liste de concepts, qui ne sont pas mentionnés explicitement dans les documents. L'approche combine des étapes permettant à la fois de peupler et d'enrichir l'ontologie du domaine, qui a donc un rôle de pivot entre les tâches. Saupodoc s'adapte à divers domaines : elle peut être utilisée pour annoter divers corpus, moyennant en entrée une ontologie propre au domaine considéré. L'approche est automatique. En effet, elle se base sur un certain nombre d'entrées qui doivent être fournies mais ne nécessite en cours d'exécution aucune interaction avec le concepteur. De plus, elle est modulaire : plusieurs tâches se succèdent. De ce fait, l'approche peut être facilement modifiée : par exemple, s'il n'y a pas besoin de rechercher d'informations dans une ressource externe ou si le concepteur est directement capable de donner une définition, on peut ne pas effectuer la tâche correspondante ; ou encore s'il faut améliorer une tâche, cela ne remet pas en cause les autres tâches. Dans le cadre de notre collaboration avec Wepingo, l'approche est utilisée pour annoter des documents publicitaires avec des concepts cibles qui correspondent aux besoins utilisateurs que Wepingo a répertoriés.

L'approche met en jeu un certain nombre de tâches qui doivent coopérer autour de l'ontologie. Arriver à faire coopérer et à adapter les composants correspondants est un enjeu complexe. En effet, cela implique :

- d'adapter les entrées des composants : gérer les différences de format, par exemple entre les annotations du concepteur (fichier excel) et le format d'entrée de DL-Learner (fichier .conf avec des notations spécifiques) ; gérer les paramètres des composants, par exemple pour manipuler correctement DL-Learner dans un contexte comme le nôtre.

- d'adapter les sorties des composants : munir GATE du traducteur JAPE pour obtenir les assertions souhaitées ; insérer les définitions obtenues par DL-Learner dans l'ontologie via un axiome d'équivalence avec sa classe cible et simuler l'hypothèse de nom unique (UNA) en ajoutant des disjonctions entre les individus de l'ontologie.

- d'assurer une certaine cohérence, en l'occurrence ici rester sous l'hypothèse du monde clos (CWA) durant toutes les tâches du processus.

  À l'étape 1, pour chaque entité, si une assertion de propriété n'est pas créée, alors l'assertion n'existe pas. En effet, l'extraction des assertions de propriété à partir des documents opère sous CWA puisque les documents sont complets par rapport aux *propriétés des documents*. De plus, l'extraction des assertions de propriété à partir des ressources externes opère "à peu près" sous CWA grâce au modèle d'acquisition des données présenté brièvement Section 4.2.1. En effet, les chemins d'accès fournissant des valeurs approximatives sont de bons moyens de pallier l'incomplétude. Si des valeurs, même approximatives, pouvaient être obtenues pour chaque propriété existante, cette tâche serait bien sous CWA. Ici, nous ne pouvons pas l'affirmer complètement puisqu'il sera possible de ne pas trouver de données parfois. Mais un gros effort est fait pour éviter au maximum les données manquantes, donc nous simulons "à peu près" l'hypothèse du monde clos.

  Pour l'étape 2, comme exprimé dans la Section 4.2.2, les définitions trouvées respectent l'hypothèse du monde clos. De plus, elles sont définies de sorte que leur application soit identique quel que soit le mode de raisonnement (CWA ou OWA). L'application des définitions par un raisonneur OWL (OWA) respecte donc l'hypothèse du monde clos.

Un gros avantage de l'approche est qu'elle génère les définitions des concepts cibles sous une forme qui est à la fois compréhensible par un humain et interprétable par une machine. Dans le cadre du travail de Wepingo, le cas où un concept cible, donc un besoin utilisateur, n'a que des annotations négatives est problématique. En effet, Wepingo ne serait pas en mesure de proposer quoi que ce soit à un utilisateur ayant ce besoin. Notre approche permet de raffiner les définitions pour obtenir des annotations positives.

Le Chapitre suivant détaille les expérimentations effectuées sur l'approche Saupodoc.

Chapitre 5

Expérimentations

Sommaire

Ce chapitre présente les différentes expérimentations que nous avons effectuées afin de valider l'approche SAUPODOC.

5.1 Procédure d'évaluation

Pour évaluer notre processus d'annotation, nous avons cherché à nous comparer avec des approches permettant d'effectuer un étiquetage de documents. Les deux approches les plus proches de nos travaux [Petasis *et al.* 2013, Yelagina & Panteleyev 2014], citées dans l'état de l'art, prennent en entrée l'équivalent de nos définitions de concepts

cibles. Comme un des apports de notre approche réside dans l'apprentissage de ces définitions, nous ne pouvons pas nous comparer à celles-ci. Cependant, les approches classiques de classification sont capables d'annoter des documents de la même manière que SAUPODOC, c'est-à-dire avec une annotation positive (*cc*) ou une annotation négative (*non cc*) pour chaque concept cible *cc*. De ce fait, nous avons décidé de nous comparer à celles-ci, plus précisément à deux classifieurs bien connus : machine à vecteurs de support (SVM) [Cortes & Vapnik 1995, Joachims 1998] et arbre de décision [Quinlan 1986].

Les SVM sont une généralisation des classifieurs linéaires. Ils consistent à délimiter des classes dans l'ensemble d'apprentissage en cherchant à séparer les éléments appartenant à deux classes distinctes par un hyperplan. Les SVM reposent sur deux idées clés. La première idée est la notion de marge maximale. En effet, supposons que deux classes soient linéairement séparables par un hyperplan séparateur linéaire. Dans ce cas, il existe une infinité de séparateurs. Parmi ceux-ci, on choisit l'hyperplan séparateur optimal tel que sa distance avec les exemples les plus proches des deux classes soit la plus grande possible (appelée marge maximale). La marge est donc la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés *vecteurs supports*. Afin de pouvoir traiter des cas où les données ne sont pas linéairement séparables, la deuxième idée des SVM est de transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension (possiblement de dimension infinie), dans lequel il est probable qu'il existe un séparateur linéaire. Ceci est réalisé grâce à une fonction, nommée fonction noyau.

Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches (les "feuilles" de l'arbre), et sont atteints en fonction de décisions prises à chaque étape. L'arbre de décision a l'avantage d'être lisible et rapide à exécuter.

Afin de pouvoir évaluer la qualité des annotations obtenues, l'ensemble des documents manuellement annotés par le concepteur est utilisé. Cet ensemble est découpé en deux parties :

- 2/3 des documents sont utilisés comme **ensemble d'apprentissage**. En d'autres termes, ce sous-ensemble est utilisé pour réaliser l'apprentissage des définitions grâce aux exemples positifs et négatifs donnés.
- 1/3 des documents est utilisé comme **ensemble de test**. En d'autres termes, cet ensemble permet d'évaluer les annotations obtenues par chaque approche en comparant celles-ci avec les annotations correctes. Les comparaisons sont basées sur plusieurs mesures : exactitude (accuracy), précision, rappel, F-mesure.

Pour évaluer et comparer de façon équitable SAUPODOC et les approches de classification, nous considérons la même terminologie. En effet, SAUPODOC est basée

sur une ontologie tandis que les classifieurs ne le sont pas. Ceux-ci prennent en entrée les documents représentés sous la forme d'une liste d'attributs et d'une étiquette. L'étiquette est binaire pour un concept cible donné (l'annotation de ce document avec ce concept cible est vrai ou fausse). Pour la liste d'attributs, nous utilisons une représentation vectorielle (Vector Space Model [Salton *et al.* 1975]) de chaque document. Une méthode sac-de-mots [Salton & McGill 1986] est utilisée et emploie la terminologie du domaine comme dictionnaire, c'est-à-dire l'ensemble des labels ou expressions-clés associés à chaque individu initial de l'ontologie (les instances de classes descriptives). Cela signifie que chaque élément du vecteur (chaque attribut) correspond à un mot du dictionnaire, donc à un individu de l'ontologie. Ainsi, des expressions distinctes (les différents labels) faisant référence à la même idée (au même individu ontologique), telles que scuba diver, dive ou encore diving, sont utilisées pour un même mot du dictionnaire (le mot "_diving"), donc un même élément du vecteur. Tous les mots du dictionnaire sont lemmatisés. Si un mot du dictionnaire est trouvé dans un document, préalablement lemmatisé, alors la valeur associée à l'élément du vecteur correspondant à ce mot est la valeur TF-IDF [Salton & Buckley 1988], sinon la valeur est 0.

Le TF-IDF (*Term Frequency - Inverse Document Frequency*), équation (5.1), est une mesure qui permet d'évaluer l'importance d'un mot contenu dans un document issu d'un corpus de documents. Le poids d'un mot varie proportionnellement à la fréquence du mot dans le document (TF Term Frequency donné en équation (5.2)). Il varie également d'une manière inversement proportionnelle en fonction de la fréquence du mot dans le corpus (IDF Inverse Document Frequency donné en (5.3)), l'idée étant que plus un mot est rare dans le corpus, plus sa présence dans un document a de poids.

$$\forall\ mot \in Dictionnaire, \forall\ doc \in Corpus,$$

$$tf\text{-}idf(mot, doc) = tf(mot, doc) \times idf(mot) \tag{5.1}$$

avec

$$tf(m, d) = \frac{\text{nb d'occurrences de } m \text{ dans } d}{\text{nb de mots dans } d} \tag{5.2}$$

$$idf(m) = \log \frac{\text{nb de documents dans } Corpus}{\text{nb de documents contenant le mot } m \text{ dans } Corpus} \tag{5.3}$$

Un classifieur va, pour un corpus donné, considérer une liste de vecteurs comprenant un vecteur pour chaque document qui représente son contenu par rapport à la liste d'attributs établie et un label pour le concept cible en question. Comme la liste d'attributs établie correspond aux individus instances des classes descriptives de l'ontologie, cette représentation en sac-de-mots est en réalité une représentation en sac-de-relations-extraites, beaucoup plus précise qu'un simple sac de mots. Pour chaque concept cible, nous lançons le classifieur avec les mêmes attributs mais nous changeons l'étiquette binaire en fonction de l'annotation manuelle du document (positive ou négative) associé à ce concept cible. Par exemple, pour un concept cible cc_1 étiqueté positivement pour un document d,

le vecteur considéré représente le sac-de-relations-extraites de d et l'étiquette 1 (positif).

Prenons l'exemple de l'ontologie utilisée pour traiter les destinations de vacances. Elle contient initialement 190 individus instances de classes descriptives. Le vecteur créé pour un classifieur aura donc 190 attributs. Ainsi, le vecteur représentant la République dominicaine est composé entre autres des valeurs suivantes : *beach* 0.0221, *diving* 0.03738, *watersport* 0.0108, *wintersport* 0.0, etc. Les différents labels et expressions-clés associés à chaque individu ne font qu'un avec cet individu. Par exemple, la valeur de TF-IDF associée à "diving" pour la République dominicaine provient à la fois de la présence de diving (présent 2 fois) et de scuba divers (présent 1 fois) dans le document (nombre d'occurrences du mot dans le document = 3). Dans l'approche SAUPODOC, la présence de ces deux termes va permettre d'ajouter l'assertion <Dominican_Republic hasActivity _diving>. SAUPODOC et les classifieurs travaillent tous les trois sur la même terminologie mais ne l'utilisent pas de la même façon. L'avantage de SAUPODOC est l'utilisation d'une ontologie et donc d'informations permettant d'identifier la plus ou moins grande similarité entre deux individus. Par exemple, les individus _diving et _snorkeling sont vus comme étant proches car ils sont tous deux instances de la classe UnderWaterActivity (descendant de Watersport). L'avantage du sac-de-mots TF-IDF est la notion de fréquence. Un mot fortement présent dans un texte n'aura pas la même valeur qu'un mot présent une seule fois, et un mot souvent présent dans le corpus n'aura pas la même valeur qu'un mot peu présent. Ces deux différences sont discutées dans la Section 5.5.

Pour chacun des deux classifieurs, SVM et arbre de décision, plusieurs paramètres sont testés. Nous ne gardons dans nos résultats de comparaison (cf. Section 5.4) que la configuration ayant mené à la meilleure exactitude. Les paramètres mis en jeu sont les suivants :

- pour SVM : nous testons plusieurs fonctions noyau usuellement employées avec SVM. Tout d'abord, nous testons un noyau polynomial avec plusieurs valeurs pour l'exposant (1 - 2 - 3). Un noyau polynomial avec un exposant de 1 (noyau linéaire) permet d'obtenir un hyperplan séparateur linéaire (par exemple, en deux dimensions on aura une droite séparatrice), tandis qu'avec un exposant plus grand on obtient un hyperplan séparateur polynomial (par exemple, en deux dimensions on aura une courbe séparatrice). La seconde fonction noyau testée est le noyau RBF (Radial Basis Function) qui utilise un paramètre gamma pour lequel nous testons différentes valeurs (0,01 - 0,04 - 0,25). Ce type de fonction permet de définir l'hyperplan séparateur d'une manière plus fine qu'une fonction polynomiale. Pour toutes ces fonctions noyau, nous testons aussi plusieurs valeurs de complexité pour tous ces noyaux (1 - 10). Ce dernier paramètre correspond au coût de violation des contraintes : plus la valeur est grande, plus on impose que les données sont sûres (non bruitées) et donc plus la séparatrice obtenue devra chercher à séparer au maximum les données de classes différentes.

Cela représente donc 12 modèles SVM pour lesquels nous ne gardons à chaque fois que le meilleur, c'est-à-dire celui avec la meilleure exactitude.

- pour les arbres de décision : nous testons plusieurs valeurs pour le facteur de confiance (0,03 - 0,2 - 0,25). Plus cette valeur est petite, plus les données sont supposées contenir du bruit. De ce fait, on préfère obtenir un arbre plus élagué dans ce cas. Un arbre plus élagué est un arbre de plus petite taille, i.e., qui comporte moins de conditions pour atteindre une feuille. Ici aussi nous ne gardons que le meilleur des 3 arbres.

Les exemples positifs et négatifs de chaque concept cible doivent être donnés en entrée pour chaque approche testée. Deux domaines ont été considérés : le domaine des destinations de vacances et celui des films (cf Section 5.3). Les exemples positifs et négatifs sont donnés par le concepteur de l'application dans le cas des destinations et automatiquement générés pour les films.

5.2 Versions des outils utilisés

L'évaluation expérimentale a été réalisée en utilisant les versions suivantes des composants de Saupodoc :

- GATE 8.0,
- DBpedia Spotlight 0.7,
- DBpedia 2014,
- DL-Learner 1.0,
- FaCT++ 1.6.2.

Pour les classifieurs, nous utilisons le logiciel Weka 3.6.6 [Hall *et al.* 2009] qui implémente des algorithmes d'apprentissage automatique. En particulier, l'algorithme pour construire un SVM est implémenté sous le nom de SMO [Platt 1998, Keerthi *et al.* 2001, Hastie & Tibshirani 1998] et celui pour construire un arbre de décision est implémenté sous le nom de J48 [Quinlan 1993]. Pour lemmatiser au préalable les documents et les mots du dictionnaire, nous utilisons Stanford NLP 3.4.1 [Manning *et al.* 2014].

5.3 Les données utilisées

Des expérimentations ont été effectuées dans deux domaines d'application, chacun ayant des caractéristiques différentes. Le premier domaine concerne un petit corpus sur lequel nous pouvons vérifier les assertions obtenues avec GATE et DBpedia. L'autre regroupe un grand nombre de documents et permet de vérifier l'applicabilité de la phase d'apprentissage sur un grand nombres d'instances. De plus la richesse de l'ontologie diffère sur ces deux domaines. Nous étudions comment celle-ci affecte la qualité des résultats.

5.3.1 Le domaine des destinations de vacances

Le corpus

Le corpus des destinations est petit (80 documents), ce qui rend possible une vérification manuelle des assertions trouvées. Chaque document a été automatiquement extrait à partir du catalogue de Thomas Cook[1] et décrit une destination particulière (pays, région, île ou ville), telle que la République dominicaine présentée en Fig. 5.1. Les documents issus de ce catalogue sont promotionnels, i.e. ils mettent en avant les qualités des destinations et évitent de faire mention des caractéristiques absentes. Ils contiennent donc très peu d'expressions négatives. Les coordonnées géographiques et les données météorologiques ne sont pas explicitées dans les documents et seront extraites de DBpedia grâce au modèle d'acquisition et à la génération automatique des requêtes SPARQL (Chapitres 7 et 8).

```
<destination>
    <name>Dominican Republic</name>
    <text>
        <description>
            Choose Dominican Republic holidays and escape from it all on a
            Caribbean island with a distinct Latin flavour. The Dominican
            coast offers postcard-perfect sceneries, from white sand beaches to
            jutting mountains and thick rainforests further inland. Influenced
            by its closest island neighbours, Cuba and Puerto Rico, the
            Dominican Republic is a feast of colour. Marvel at the merging of
            tropical blues where the sky touches the water as well as its
            colourful rainbow of traditional painted houses and huts. [...]
        </description>
        <wheretostay>
            Palo Bonito: Palo Bonito is a tiny village found in La Altagracia,
            the easternmost province of the Dominican Republic famous for its
            fantastic beaches and beautiful natural landscapes.
            Bayahibe: What was once a small fishing village, is now a busy
            resort, especially loved by scuba divers. Over 20 exiting diving
            sites and 3 old shipwrecks are waiting to be discovered.
            [...]
        </wheretostay>
    </text>
</destination>
```

Fig. 5.1 Une partie du fichier décrivant la République dominicaine

L'ontologie

L'ontologie du domaine des destinations comprend une classe principale, Destination, et 161 classes descriptives. Une représentation simplifiée de sa structure a été donnée

[1] http://www.thomascook.com/

Fig. 4.1 page 37.

Les classes descriptives sont utilisées pour caractériser la nature de l'environnement (46 classes), les activités possibles (102 classes), le type de familles concernées, par exemple, avec enfants, couples, etc. (6 classes) et des classes relatives aux températures comme les saisons (7 classes). Les classes descriptives contiennent des instances et leurs formes terminologiques (propriétés d'annotation "label" et "isDefinedBy") pour faciliter leur identification dans les textes. Par exemple, les termes archaeology, archaeological, acropolis, roman villa, excavation site, mosaic sont associés à l'individu _archaeology.

Les concepts cibles

39 concepts cibles sont considérés. Parmi eux, 20 sont des concepts liés aux saisons tels que "Destination avec sports aquatiques en hiver" et 19 sont des concepts un peu moins spécifiques tels que "Destinations côtières" ou encore "Destination avec vie nocturne". Chacune des 80 destinations du corpus est annotée par le concepteur comme un exemple positif ou négatif pour chaque concept cible.

5.3.2 Le domaine des films

Le corpus

Le corpus des films contient 10 000 documents, un nombre suffisamment grand pour vérifier l'applicabilité de l'approche avec beaucoup d'individus. Il a été construit automatiquement à partir de DBpedia. Chaque document correspond à la page DBpedia d'un film. Il contient le nom du film, son résumé et l'URI de la page DBpedia. Ainsi, la page étant déjà connue, DBpedia Spotlight ne sera pas utilisé. Un exemple de fichier XML est donné Fig. 5.2.

```
<movie>
    <name>Adventure_(1945_film)</name>
    <abstract>
        Adventure is a 1945 American romantic drama film directed by Victor
        Fleming and starring Clark Gable and Greer Garson. Based on the 1937 novel
        The Annointed by Clyde Brion Davis, the film is about a sailor who falls
        in love with a librarian. Adventure was Gable's first postwar film and the
        tagline repeated in the movie's famous trailer was "Gable's back and
        Garson's got him!" Victor Fleming was one of Gable's favorite directors.
    </abstract>
    <dbpediaPage>http://dbpedia.org/resource/Adventure_(1945_film)</dbpediaPage>
</movie>
```

Fig. 5.2 Le fichier décrivant le film "Adventure"

La durée du film ainsi que ses langues et pays d'origine seront extraits à partir de DBpedia. En effet, la durée des films n'est pas mentionnée dans les descriptions tandis

que les langues et pays d'origine peuvent l'être mais un contresens peut être possible. Par exemple, la présence du terme "French" peut avoir diverses significations : le film peut être français (pays), ou bien en français (langue), ou il peut raconter l'histoire d'un Français. C'est la raison pour laquelle, quand des contresens sont possibles, nous préférons utiliser les informations de DBpedia si elles existent, comme dans cet exemple, plutôt que les informations des textes.

L'ontologie

L'ontologie des films est très simple. Elle contient la classe principale Film et uniquement les 5 classes descriptives utiles pour définir les concepts cibles de nos expérimentations. En cas de nouveaux concepts cibles, il faudra éventuellement rendre l'ontologie plus exhaustive. Les deux classes descriptives Language et Country n'ont initialement pas d'instances, puisque l'étape de complétion est capable d'ajouter grâce à DBpedia non seulement des assertions de propriété mais aussi d'instancier les classes descriptives. Ainsi, si un film f est lié à la langue French_language[2] dans DBpedia, alors la requête SPARQL automatiquement générée permet d'ajouter cette langue sous la forme d'un individu dans l'ontologie (_French_language), d'associer f avec ce nouvel individu via la propriété hasLanguage (1.), de typer ce nouvel individu avec le co-domaine de la propriété instanciée (2.), et enfin de représenter dans l'ontologie l'ensemble des labels de ce nouvel individu (3.). Ainsi, on obtient :

1. $<$ f ; hasLanguage ; _French_language $>$: assertion de propriété liant le film f a l'individu _French_language nouvellement créé.

2. $<$ _French_language ; rdf:type ; Language $>$: typage de l'individu nouvellement créé avec le concept qui est le co-domaine de la propriété considérée (hasLanguage).

3. $<$ _French_language ; rdf:label ; French language $>$: le littéral French language est ajouté comme label du nouvel individu car c'est un label de la ressource DBpedia French_language. Il en sera fait de même pour tous les autres labels présents dans la page DBpedia French_language.

Dans le cadre de notre approche, la terminologie du domaine présente dans l'ontologie peut donc être partielle, car la tâche de complétion utilisant DBpedia peut étendre cette terminologie. Par exemple, elle ajoute automatiquement les noms de langues et de pays dans le cas du domaine des films.

Pour effectuer une évaluation équitable envers les deux classifieurs avec lesquels nous nous comparons, les termes du domaine manquants (noms de langue et pays) sont ajoutés dans le dictionnaire des classifieurs. De ce fait le dictionnaire contient les individus de l'ontologie des films ainsi que la liste suivante de mots représentant les noms de langues et de pays qui semblent essentiels par rapport aux concepts

[2]http://dbpedia.org/resource/French_language

cibles considérés pour le domaine des films. Le format de ces mots est le suivant : "_nom_du_mot : liste des labels du mot".

- _english : English
- _american : American
- _america : America, US, USA, United States
- _indian : Indian
- _india : India
- _british : British
- _england : England, Great Britain
- _hindi : Hindi
- _italian : Italian
- _italy : Italy
- _french : French
- _france : France
- _spanish : Spanish
- _japanese : Japanese
- _japan : Japan
- _tamil : Tamil

Les concepts cibles

Pour simplifier l'évaluation et donc éviter la phase d'annotation manuelle, nous considérons ici 12 concepts cibles pour lesquels les documents du corpus peuvent être automatiquement annotés. Les 12 concepts cibles choisis correspondent à des catégories de DBpedia explicitées via la propriété dcterms:subject liant une ressource DBpedia à une catégorie. Un film f est considéré comme un exemple positif pour un concept cible correspondant à une catégorie c si $<f$ dcterms:subject $c>$, sinon il s'agit d'un exemple négatif. Nous avons choisi les 12 catégories parmi celles les plus représentées dans nos films. Celles-ci ont en moyenne 785 exemples positifs sur nos 10 000 films.

5.4 Résultats obtenus

Cette section présente et discute les résultats obtenus tout d'abord sur l'ensemble de test puis sur un ensemble de validation. Nous comparons les résultats de SAUPODOC avec ceux des deux classifieurs testés (SVM et arbre de décision) sur plusieurs mesures et sur nos deux domaines.

5.4.1 Expérimentations sur l'ensemble de test

Les résultats de cette section proviennent de l'ensemble de test (1/3 des documents annotés). Nous comparons les annotations obtenues avec l'approche SAUPODOC (toutes les tâches) et celles obtenues avec chacun des classifieurs. Nous nous intéressons à quatre mesures données dans les équations (5.4), (5.5), (5.6) et (5.7). Ces mesures considèrent le nombre de vrais positifs (VP), vrais négatifs (VN), faux positifs (FP) et faux négatifs (FN). Pour un concept cible donné, les vrais positifs regroupent les documents annotés correctement comme positifs, les vrais négatifs regroupent les documents annotés correctement comme négatifs, les faux positifs regroupent les documents annotés comme positifs alors qu'ils auraient dû être annotés comme négatifs et les faux négatifs regroupent les documents annotés comme négatifs alors qu'ils auraient dû être annotés comme positifs.

$$\begin{aligned} \textit{Exactitude} &= \frac{VP+VN}{VP+FP+VN+FN} \\ &= \frac{\text{nombre de documents correctement annotés}}{\text{nombre de documents}} \end{aligned} \tag{5.4}$$

$$\textit{F-mesure} = \frac{2 \times \textit{précision} \times \textit{rappel}}{\textit{précision} + \textit{rappel}} \tag{5.5}$$

$$\begin{aligned} \textit{Précision} &= \frac{VP}{VP+FP} \\ &= \frac{\text{nombre de documents correctement annotés en positifs}}{\text{nombre de documents annotés en positifs}} \end{aligned} \tag{5.6}$$

$$\begin{aligned} \textit{Rappel} &= \frac{VP}{VP+FN} \\ &= \frac{\text{nombre de documents correctement annotés en positifs}}{\text{nombre de documents qui auraient dû être annotés en positifs}} \end{aligned} \tag{5.7}$$

Nous pouvons observer (cf. Tableau 5.1 et Figure 5.3) que les trois approches donnent de bons résultats pour l'exactitude mais que SAUPODOC est meilleure. Cependant, dans notre contexte, chaque concept cible a en général beaucoup d'exemples négatifs

et peu de positifs. De ce fait, un classifieur qui ferait une prédiction systématiquement négative sur toutes les entrées aurait une exactitude élevée (91,76% en moyenne sur les concepts cibles des films). Malgré cette exactitude élevée, un tel classifieur ne permettrait de trouver aucune annotation positive. Dans notre cadre de travail, il est important de trouver des annotations positives pour que Wepingo puisse proposer des produits qui correspondent aux besoins utilisateurs. Ainsi, l'exactitude n'est pas la seule mesure à considérer dans notre problème. D'autres mesures comme la précision, le rappel et la F-mesure sont nécessaires pour évaluer la prédiction des positifs qui est centrale dans notre contexte. Le tableau 5.1 et les Figures 5.4, 5.5 et 5.6 montrent les résultats. Nous pouvons observer que notre approche est la meilleure pour ces trois mesures dans les deux domaines. Ainsi, même si l'écart d'exactitude n'est pas flagrant dans le domaine des films, la précision, le rappel et la f-mesure sont nettement plus élevés pour l'approche SAUPODOC : l'écart minimum est de plus de 6% (écart de précision entre SAUPODOC et SVM) et il peut aller jusqu'à plus de 20% (écart de rappel entre SAUPODOC et SVM). De même, l'écart dans le domaine des destinations varie entre plus de 5% (écart de rappel entre SAUPODOC et arbre de décision) et plus de 18% (écart de F-mesure entre SAUPODOC et SVM).

Mesure (%)	Exactitude			F-mesure			Précision			Rappel		
Corpus	**Nous**	SVM	Arbre	**Nous**	SVM	Arbre	**Nous**	SVM	Arbre	**Nous**	SVM	Arbre
Destination	**95,89**	84,52	86,23	**72,23**	54,14	63,22	**73,95**	58,10	64,23	**71,58**	55,32	65,89
Film	**95,46**	94,41	94,32	**75,65**	61,74	61,40	**76,27**	69,90	67,72	**77,76**	57,59	58,99

Tableau 5.1 Résultats moyens pour les destinations (39 *CC*) et les films (12 *CC*)
Le résultat moyen d'une mesure correspond à la moyenne de cette mesure sur chaque concept cible.

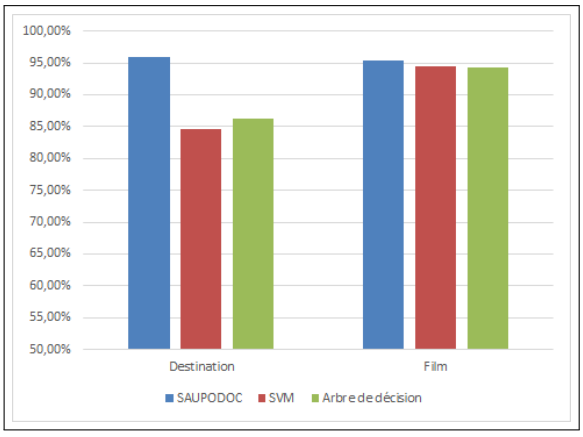


Fig. 5.3 Exactitude moyenne

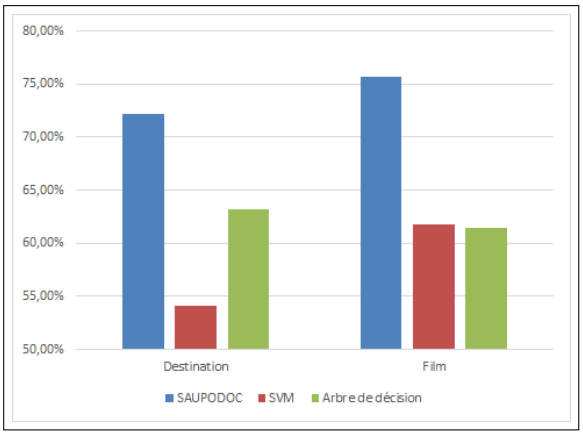


Fig. 5.4 F-mesure moyenne

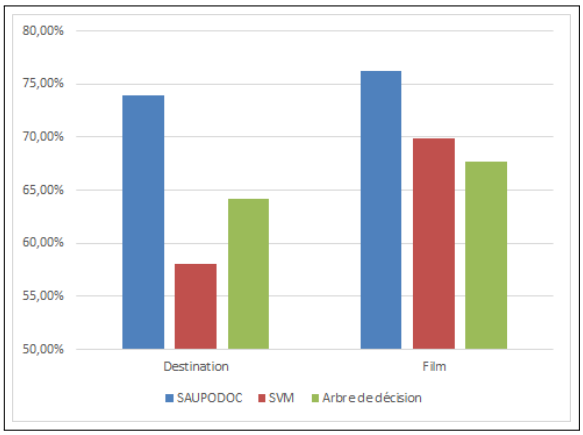


Fig. 5.5 Précision moyenne

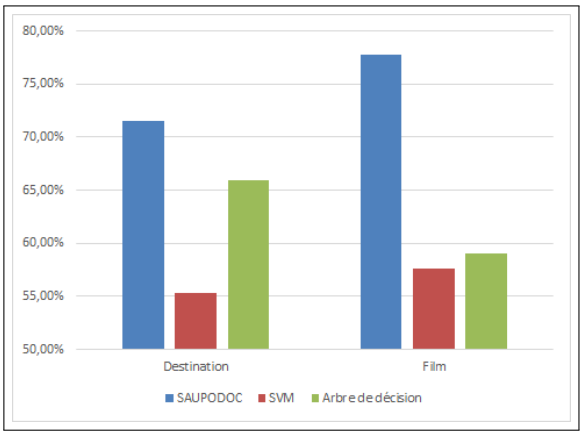


Fig. 5.6 Rappel moyen

L'annexe B présente les valeurs détaillées d'exactitude, de F-mesure, de précision et de rappel pour chaque concept cible des deux domaines. Elle précise aussi la pertinence moyenne des annotations négatives.

5.4.2 Expérimentations sur un autre ensemble de documents

Les échantillons annotés ont été découpés en 2/3 pour l'apprentissage et 1/3 pour le test. Cela signifie que les mesures données dans la section précédente portent sur 1/3 des exemples annotés. Pour notre approche, comme pour les classifieurs, un certain nombre de paramètres sont testés et nous ne gardons que les meilleurs résultats obtenus, c'est-à-dire ceux générant la meilleure exactitude. De ce fait, une sorte de biais est introduit dans les résultats obtenus puisqu'ils représentent les modèles générant la plus grande exactitude.

Nous supposons que les résultats obtenus sur l'échantillon de test sont tout de même révélateurs puisque ce biais est présent dans les trois approches. Pour vérifier cette hypothèse, nous avons utilisé un nouvel échantillon annoté dans le domaine des films. Cet échantillon dit de validation contient 10 000 nouveaux documents annotés avec les 12 concepts cibles. Il a été obtenu de la même manière que pour les ensembles d'apprentissage et de test, c'est-à-dire en utilisant DBpedia.

Sur cet échantillon, nous avons appliqué l'approche SAUPODOC avec les définitions obtenues dans l'expérimentation précédente, présentée section 5.4.1, ainsi que les classifieurs obtenus dans celle-ci (cf. Figure 5.7). Nous pouvons observer les mêmes tendances que celles dégagées pour l'ensemble de test. En effet, pour chacune des quatre mesures testées le classement des approches est le même (SAUPODOC puis SVM puis arbre de décision pour exactitude, précision, F-mesure et SAUPODOC puis arbre de décision puis SVM pour le rappel). Les écarts entre les valeurs obtenues sur l'échantillon de test et celui de validation sont minimes (allant de 0.08% à maximum

4.84%). De ce fait, l'expérimentation effectuée est cohérente.

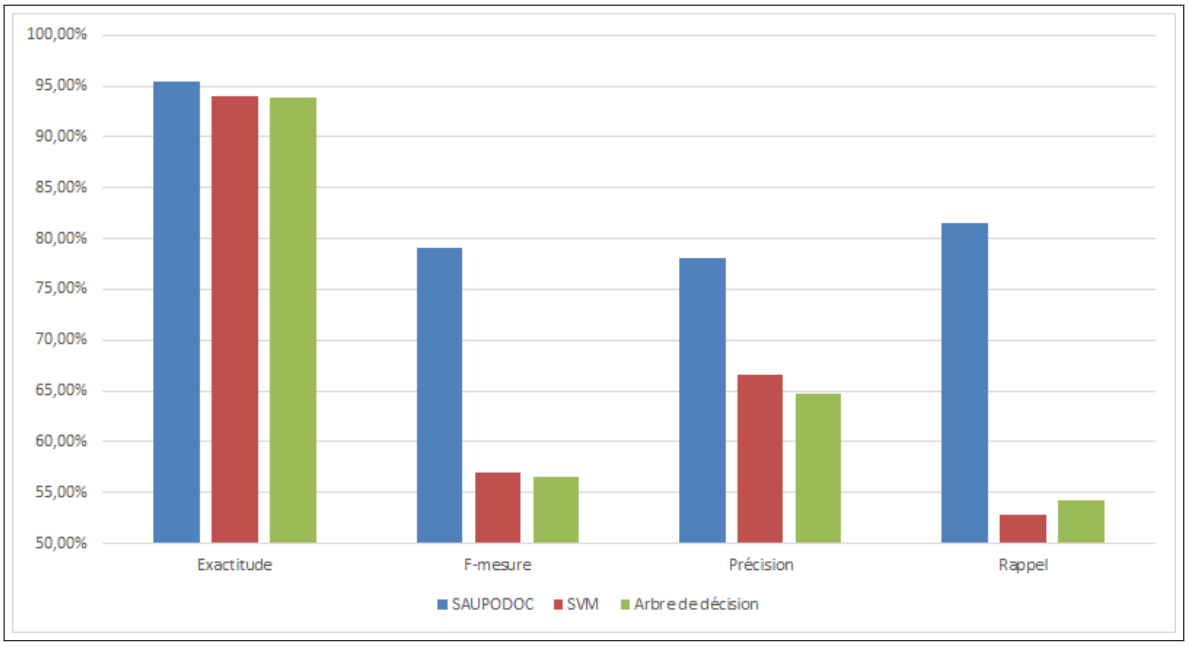


Fig. 5.7 Les résultats sur l'ensemble de validation (10 000 films)

5.4.3 Expérimentations évaluant les tâches d'extraction de données

Nos résultats combinent les performances des différentes tâches effectuées par SAUPODOC. La tâche d'apprentissage de définitions (tâche 2.a) permet une bonne classification, mais les tâches en amont de celle-ci (tâches 1.a et 1.b d'extraction de données à partir des documents et du LOD) ont aussi un impact sur les résultats dans le sens où elles affectent la qualité des données utilisées pour apprendre les définitions. Dans ce qui suit, nous analysons les deux tâches d'extraction sur le domaine des destinations. Une évaluation manuelle peut être faite puisque ce corpus contient peu de documents.

Nous commençons par analyser la phase d'extraction des propriétés à partir des textes (tâche 1.a). Parmi les 2 375 assertions de propriétés extraites, nous avons dénombré 52 assertions de propriété fausses (faux positifs). Donc le bruit est de 2,19% et la précision de 97,81%. Le rappel est supposé être égal à 1. En effet, si une assertion de propriété n'est pas mentionnée dans le texte, alors cette propriété ne caractérise pas l'instance décrite puisque toutes les caractéristiques importantes sont supposées être mentionnées dans les descriptions. Ainsi, le nombre de faux négatifs (les assertions manquantes) devrait être extrêmement limité. La valeur de précision

montre clairement que les résultats de la tâche d'extraction à partir de textes sont de bonne qualité. Les assertions pertinentes sont donc introduites dans l'ontologie avec un minimum de bruit.

Pour la tâche d'extraction à partir du LOD (tâche 1.b), les techniques proposées pour traiter les propriétés multiples, multi-valuées ou manquantes ont prouvé leur utilité. Seules 29 des 80 destinations disposaient des données souhaitées sur les températures. La spécification de chemins d'exploration a permis d'obtenir des valeurs approchées : par exemple, les températures pour Boston ont été obtenues à partir de la page de Quincy,_Massachusetts. L'annexe C regroupe plus d'informations sur les expérimentations effectuées sur cette tâche.

5.5 Expérimentations évaluant l'intérêt de la complétion des données et de l'ontologie

Un avantage important de notre approche est l'exploitation de DBpedia afin de compléter les informations des documents. Rappelons que cette complétion correspond :

- dans le cas des destinations, à des données géographiques et météorologiques,
- dans le cas des films, essentiellement à traiter les contre-sens possibles entre les données sur la langue et le pays d'origine du film.

Hormis la complétion, la principale différence entre l'exploitation des données par les classifieurs et par Saupodoc réside dans la manière de représenter les textes :

- Pour les classifieurs, l'utilisation d'un sac-de-mots TF-IDF permet d'avoir une notion de fréquence qu'il n'y a pas dans l'ontologie. En effet, dans l'ontologie, la présence ou l'absence d'une assertion de propriété est une notion binaire, bien moins fine que le TF-IDF.
- Pour Saupodoc, l'avantage de l'utilisation d'une ontologie par rapport à un sac-de-mots est la structuration dans celle-ci. Dans un sac-de-mots, il n'y a aucune notion de proximité entre les mots, au contraire de l'ontologie où les individus similaires sont des instances de classes communes, et où les classes similaires sont des sous-classes de super-classes communes.

Les Figures 5.8 et 5.9 montrent les résultats obtenus sur les quatre mesures pour les trois approches ainsi que pour l'approche Saupodoc sans la tâche de complétion des assertions par des données de DBpedia (tâche 1.b).

Pour les destinations, sans la complétion des données géographiques et météorologiques, Figure 5.8, Saupodoc est moins performant, comme on peut s'y attendre. Cependant, elle continue de surpasser les deux classifieurs sur les quatre

mesures. Par exemple, pour apprendre la définition d'un concept cible relatif à un beau temps en hiver, SAUPODOC sans la complétion ne peut pas utiliser de données météo, puisque ces données sont obtenues grâce à DBpedia. Néanmoins, cette version simplifiée de SAUPODOC est capable d'obtenir une définition s'inspirant de l'environnement des destinations de ce type, par exemple contenant l'assertion hasEnvironment some (Jungle or Vegetation). En effet, les descriptions des destinations où il fait beau en hiver mentionnent souvent une jungle dans les alentours ou bien au moins de la végétation qui n'est pas mentionnée dans les autres destinations. Grâce à la structure de l'ontologie, les individus instances de la classe Jungle (ou de la classe Vegetation) sont automatiquement vus comme des individus proches, contrairement à leur prise en compte dans les classifieurs où ils sont interprétés comme des attributs sans liens particuliers. De ce fait, en permettant de rapprocher les individus similaires en les rendant instances d'un même concept, la structuration de l'ontologie a permis à SAUPODOC sans la tâche 1.b d'obtenir des annotations plus correctes que les classifieurs dans le cas des destinations de vacances.

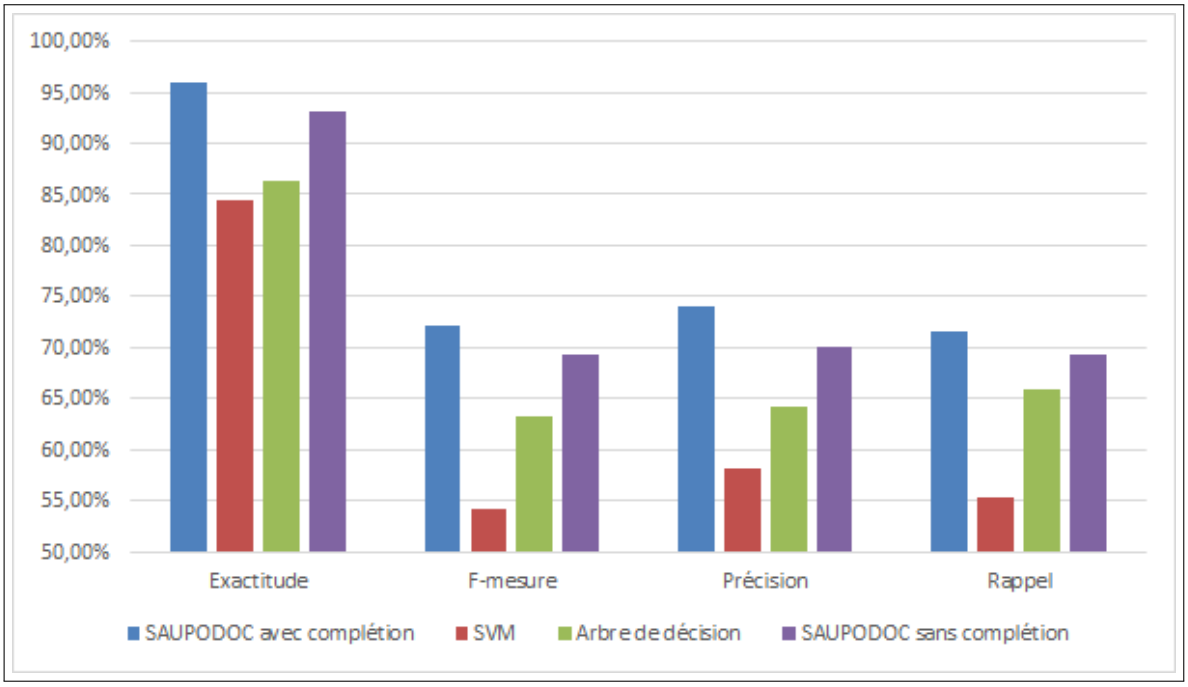


Fig. 5.8 Les mesures sur le corpus des destinations (réalisées sur l'ensemble de test)

Dans le cas des films, sans la complétion des langues et des pays, comme le montre la Figure 5.9, les quatre mesures baissent considérablement. L'exactitude de SAUPODOC sans la complétion est un peu moins bonne que celle des classifieurs. La précision est clairement plus faible que celle des classifieurs tandis que le rappel est meilleur, créant ainsi à peu près la même F-mesure. En somme, la performance de SAUPODOC sans complétion est très proche de celle des classifieurs mais un peu moins bonne. Cela est dû au fait que, dans cette expérimentation, l'ontologie des films utilisée n'a que peu de structure. Elle relie les films à 5 concepts, mais la structuration interne des concepts

est très faible. De ce fait, la puissance de l'ontologie, qui réside dans sa capacité à rapprocher les individus similaires en les associant à des mêmes classes, n'est que peu présente ici. Ainsi, SAUPODOC ne peut profiter ni de l'avantage de la structuration de l'ontologie, ni de la notion de fréquence dont profitent les classifieurs basés sur les sacs-de-mots TF-IDF.

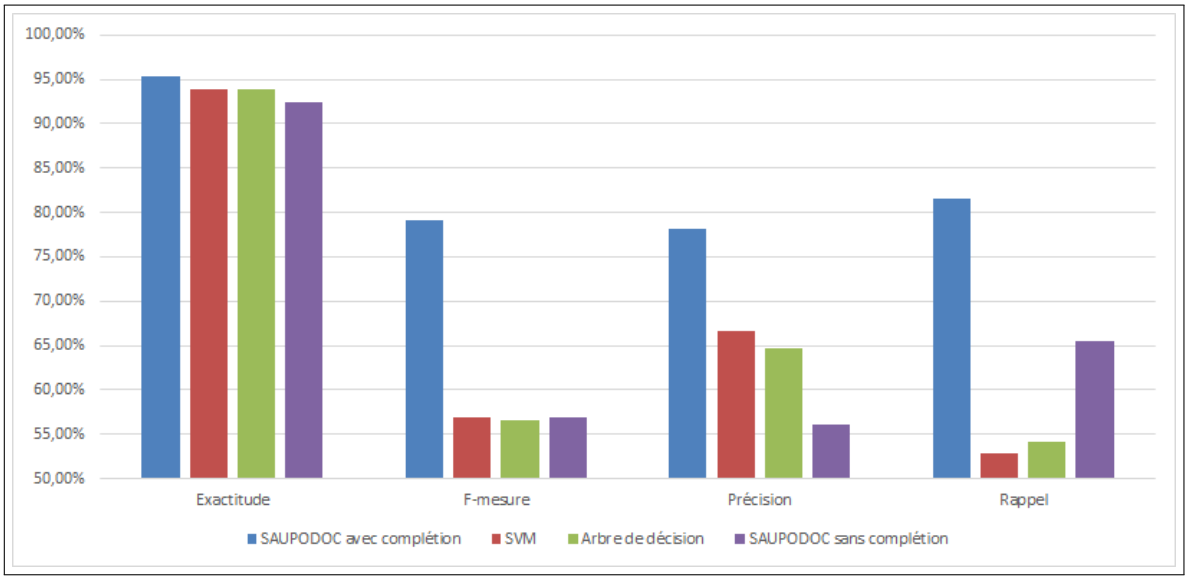


Fig. 5.9 Les mesures sur le corpus des films (réalisées sur l'ensemble de validation)

De ces expériences, nous pouvons déduire deux conclusions :

- la complétion a un intérêt pour l'approche puisqu'elle augmente sensiblement les pourcentages des mesures. Pour les destinations, les valeurs météorologiques ajoutées permettent un gain entre 2 et 4% dans nos mesures, cf. Tableau 5.2. Pour les films, les contre-sens étant très présents dans les documents textuels, la prise en compte des données externes permet un gain important : autour de 3% pour l'exactitude et d'une vingtaine de pour cent pour les autres mesures.

- l'avantage de l'utilisation d'une ontologie par rapport à un sacs-de-mots classique réside dans l'exploitation de la structuration de l'ontologie. En effet, ici le sac-de-mots contient des notions de similarité (plusieurs labels pour un même individu). Celui-ci est en quelque sorte un "sac de relations extraites". Cependant, il ne tient pas compte des concepts liant les relations extraites entre elles, ce qu'une ontologie fait. Par exemple, une entité ayant pour activité _diving et une entité ayant pour activité _waterskiing ont une similitude : elles ont toutes deux une activité de sport aquatique (ces deux individus sont des instances de sous-classes de Watersport). Cependant, il s'agit de deux mots indépendants pour un classifieur. De ce fait, la structure de l'ontologie permet un apport conséquent dans l'apprentissage. Le Tableau 5.3 montre que, avec les mêmes informations (uniquement celles des documents) représentées dans un sac-de-mots

ou dans une ontologie bien structurée, l'utilisation d'une ontologie permet un gain dans l'approche d'annotation. Au contraire, sans structure dans l'ontologie, cf. Tableau 5.4, l'ontologie en soit n'a pas d'intérêt dans l'approche d'annotation.

Corpus	Delta Exactitude	Delta F-mesure	Delta Précision	Delta Rappel
Destination	2,82%	2,97%	3,80%	2,26%
Film	2,98%	22,15%	22,08%	16,00%

Tableau 5.2 Apport de la complétion dans SAUPODOC
Le delta correspond à l'écart entre la mesure considérée pour SAUPODOC avec et sans complétion.

Corpus des destinations	Delta Exactitude	Delta F-mesure	Delta Précision	Delta Rappel
Delta SAUPODOC sans complétion par rapport à SVM	8,56%	15,11%	12,04%	14,00%
Delta SAUPODOC sans complétion par rapport à Arbre de décision	6,85%	6,04%	5,91%	3,42%
Delta moyen	7,70%	10,58%	8,98%	8,71%

Tableau 5.3 Apport de la structuration de l'ontologie : cas des destinations
Le delta correspond à l'écart entre la mesure considérée pour SAUPODOC sans complétion et les classifieurs.

5.6 Obtenir des définitions explicites : un avantage pour raffiner les annotations

Sur l'ensemble des concepts cibles mis en jeu dans les deux corpus (39 pour les destinations + 12 pour les films donc 51), il y a 8 concepts cibles pour lesquels SAUPODOC génère une définition conduisant à des annotations uniquement négatives sur l'ensemble de test[3]. Ce genre de définitions aura tendance à poser problème dans le contexte d'application. En effet, si pour un concept cible donné, on ne dispose que d'annotations négatives, Wepingo n'est alors pas en mesure de proposer de produits qui correspondent à ce besoin utilisateur. Pour traiter ce problème, comme les définitions sont intelligibles, le concepteur peut facilement les raffiner pour qu'elles génèrent

[3]Voir les histogrammes détaillés dans l'Annexe B où les valeurs de précision, rappel et F-mesure peuvent être 0.

Corpus des films	Delta Exactitude	Delta F-mesure	Delta Précision	Delta Rappel
Delta SAUPODOC sans complétion par rapport à SVM	-1,52%	-0,02%	-10,58%	12,74%
Delta SAUPODOC sans complétion par rapport à Arbre de décision	-1,45%	0,30%	-8,66%	11,31%
Delta moyen	-1,49%	0,14%	-9,62%	12,03%

Tableau 5.4 Absence d'apport sans structuration de l'ontologie : cas des films
Le delta correspond à l'écart entre la mesure considérée pour SAUPODOC sans complétion et les classifieurs.

quelques annotations positives.

Ce raffinement peut d'ailleurs être fait de manière semi-automatique. Une méthodologie peut être mise en place afin d'élargir la définition trouvée. Par exemple, on peut tenter de supprimer une clause dans le cas d'une définition avec "and". On peut aussi exploiter la structure de l'ontologie pour remplacer une classe par un de ses ascendants (par exemple remplacer "hasActivity some WaterSport" par "hasActivity some WaterActivity" car WaterActivity est un concept généralisant WaterSport). Enfin, une modification du seuil en cas d'inégalités numériques est envisageable (avgTemperatureC some double[>= 20.0] au lieu de avgTemperatureC some double[>= 23.0]). Ces diverses définitions raffinées pourront être proposées au concepteur si elles permettent de générer des annotations positives. Le concepteur pourra alors les valider s'il estime que l'élargissement proposé est suffisamment cohérent avec le concept considéré.

Les classifieurs rencontrent le même problème : certains concepts cibles n'ont que des annotations négatives sur l'ensemble de test. C'est le cas pour 11 concepts cibles pour SVM et 6 concepts cibles pour les arbres de décision. Cependant, ceux-ci ne fournissent pas de définitions explicites donc un raffinement serait plus difficile à mettre en place. En effet, les classifieurs SVM créent un modèle qui n'est pas compréhensible par un humain. Les résultats des arbres de décision le sont un peu plus puisque les arbres peuvent être vus comme des ensembles de règles. Toutefois ces règles font référence aux valeurs de TF-IDF associées aux mots du dictionnaire, ce qui les rend difficilement interprétables par un humain. Par exemple, l'arbre de décision pour le concept cible coastal destinations est donné dans la Figure 5.10 (exactitude de 96,30%). Cela signifie que les annotations positives sont faites quand la valeur TF-IDF du mot urban du dictionnaire est inférieure à 0,18893 et que celle de beach est 0 et que celle de sea est supérieure à 0,005502, ou bien si la valeur de urban est inférieure à 0,18893 et celle de beach supérieure à 0. Un tel arbre est difficile à ajuster par le concepteur au cas où un raffinement est nécessaire.

```
_urban <= 0.018893
|   _beach <= 0
|   |   _sea <= 0.005502: 0
|   |   _sea > 0.005502: 1
|   _beach > 0: 1
_urban > 0.018893: 0
```

Fig. 5.10 L'arbre de décision pour les destinations côtières

Outre le raffinement, avoir des définitions explicites permet aussi de se rendre compte plus facilement d'éventuelles erreurs introduites dans les entrées. Une analyse a été menée à ce sujet en Annexe D.

Conclusion

Pour évaluer notre approche, il a été difficile de trouver des approches similaires auxquelles la comparer. En effet, les approches composites mentionnées dans l'état de l'art [Petasis *et al.* 2013, Yelagina & Panteleyev 2014] utilisent des définitions, dont nous ne disposons pas au départ et que SAUPODOC doit justement apprendre. De ce fait, nous ne pouvons pas nous comparer à de telles approches. Nous avons donc choisi de nous comparer à des techniques de classification par apprentissage automatique.

Ce chapitre montre que l'approche SAUPODOC génère des annotations satisfaisantes, meilleures que les techniques classiques d'apprentissage automatique. Les étapes préalables à l'apprentissage des définitions ne génèrent que peu de bruit et permettent d'obtenir des définitions pertinentes en termes d'exactitude, précision, rappel et F-mesure.

La complétion des données via le LOD améliore les résultats de l'approche. Néanmoins, sans celle-ci, l'approche est capable de trouver des annotations convenables, meilleures que celles des classifieurs classiques, quand l'ontologie est structurée.

Enfin, un avantage important de notre approche est la création de définitions explicites, donc compréhensibles par un humain et interprétables par une machine. Cela permettra dans un travail futur de raffiner les définitions ne générant que des annotations négatives pour obtenir des annotations positives. Ce raffinement pourrait être automatisé.

La partie suivante relate notre travail d'acquisition des données du LOD effectué dans la tâche 1.b de l'approche SAUPODOC.

Partie II

Peupler une ontologie avec des données du LOD

Chapitre 6

État de l'art : Acquisition de données du Web des données

Sommaire

Dans le cadre de l'approche décrite dans cette thèse, nous devons acquérir des données du Web des données et les insérer dans une ontologie. Ce chapitre définit le Web des données. Ce dernier présente des problèmes de qualité, notamment d'incomplétude. De plus, acquérir des données du Web des données pour les mettre dans une ontologie pose un certain nombre de problèmes. D'une part, cela crée un problème d'hétérogénéité sémantique puisque les schémas et le vocabulaire utilisés dans ces deux sources ne sont pas les mêmes. D'autre part, arriver à accéder aux données peut être une tâche complexe. Ainsi, ce chapitre dresse un état de l'art des travaux traitant de problèmes d'incomplétude, d'hétérogénéité sémantique et des difficultés d'accès aux données.

Un des faits marquants de l'évolution de l'ingénierie des connaissances est d'avoir diversifié les sources de connaissances utilisées dans les systèmes de traitement d'information intelligents, permettant ainsi de tirer profit de leur complémentarité

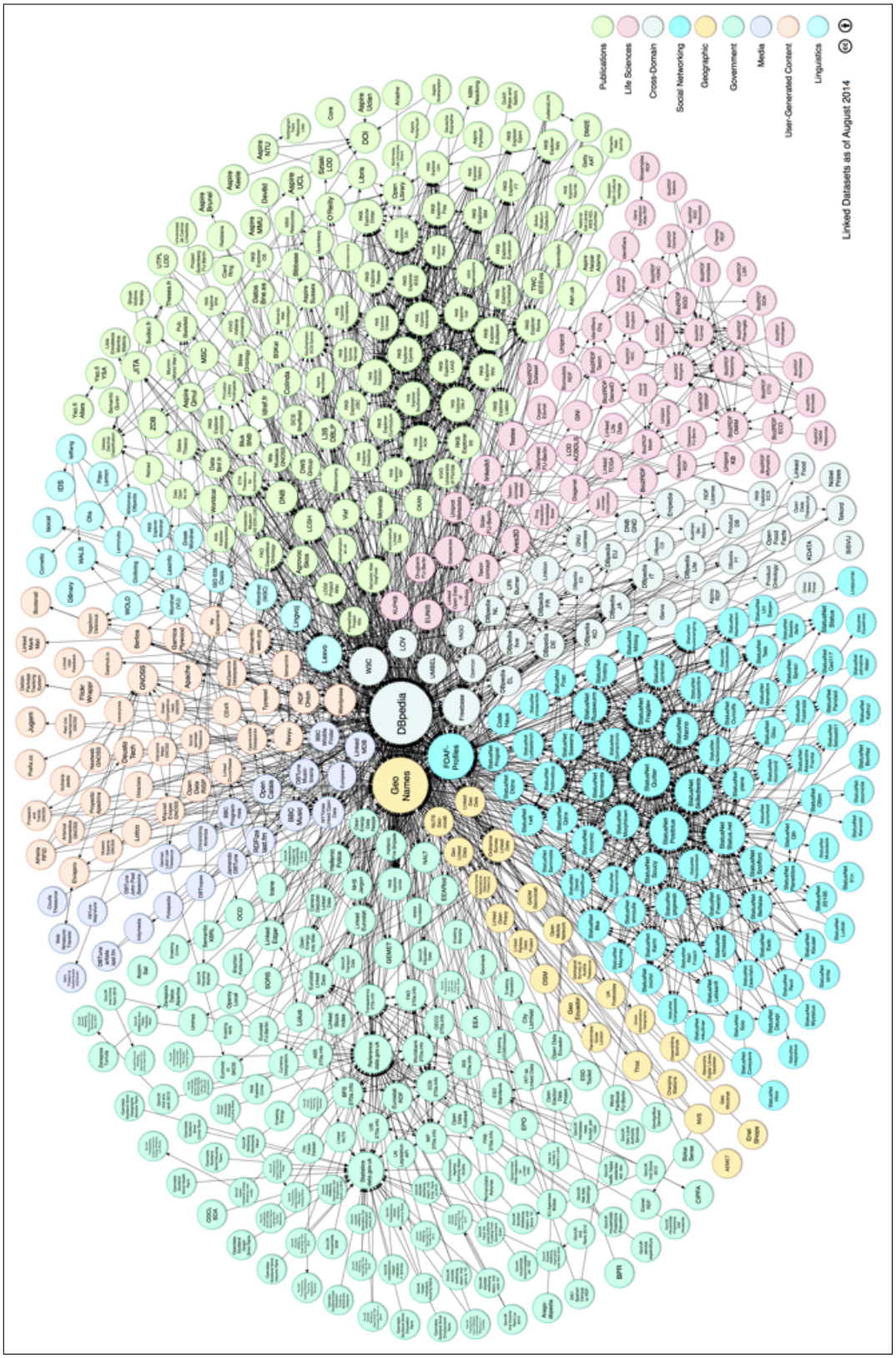


Fig. 6.1 Les sources de données du Linked Open Data cloud[a]

[a]Linking open data cloud diagram 2014, http://lod-cloud.net/

[Aussenac-Gilles *et al.* 2014]. Le Web des données liées est aujourd'hui une source de connaissances utilisable, tout comme les documents numériques exploités en complément d'experts spécialistes du domaine, les données dont sont extraites des connaissances, les modèles, ressources, thesaurus, représentations structurées comme les ontologies. La Figure 6.1 montre les sources de données (datasets) publiées au mois d'août 2014 sous la forme de données liées.

Les données liées réfèrent à un style de publication et d'interconnexion de données du Web structurées, représentées en RDF. Par définition, on peut y accéder par des mécanismes d'accès normalisés. Ces sources de données peuvent aussi être facilement explorées par les moteurs de recherche. Elles sont liées avec d'autres sources de données. Le nombre de données publiées selon ce principe croît rapidement. On parle aujourd'hui de dizaines de milliards de triplets publiés sur Internet. Cette masse de données représente une source prometteuse utilisable dans de très nombreuses applications du Web sémantique, à condition toutefois de développer des techniques de collecte de données adaptées. En effet, des études [Zaveri *et al.* 2014] ont mis en évidence des problèmes de qualité tels que l'incomplétude, la présence d'informations redondantes, l'inexactitude de certaines données. Dans cet état de l'art, nous nous intéressons à l'incomplétude, qui peut entraver l'utilisation de jeux de données lorsqu'on s'intéresse à des applications du monde réel basées sur le LOD.

Ce chapitre décrit tout d'abord des travaux liés à l'incomplétude du Web des données puis s'intéresse à l'accès aux données du Web des données : une section traite du problème d'hétérogénéité sémantique tandis qu'une autre section traite du problème d'accès complexe. Enfin, nous positionnons notre travail par rapport à l'état de l'art et concluons.

6.1 L'incomplétude du Web des données

Le Web des données est une ressource incomplète : certaines informations, bien que vraies, ne sont pas forcément mentionnées. Cette incomplétude n'est pas un problème d'un point de vue logique puisque la sémantique RDF suppose l'hypothèse du monde ouvert qui suppose que certaines informations peuvent être manquantes. En revanche, certaines applications ont besoin d'avoir des informations complètes. Pour cette raison, divers travaux se sont intéressés à cette incomplétude. Nous décrivons ici quelques travaux cherchant à identifier les propriétés manquantes d'un objet, les typages d'entités manquants ou encore s'intéressant à l'incomplétude relative de valeurs.

Certains travaux détectent les valeurs de propriétés manquantes et guident ainsi l'utilisateur pour les rajouter. Ainsi, LOD Miner [Simonic *et al.* 2013] s'intéresse au problème de prédiction de propriétés manquantes pour un objet donné. Par exemple, considérons le graphe RDF de la Figure 6.2. Trois objets sont considérés : Audi, Mercedes-Benz et Fiat. Fiat a quatre propriétés instanciées : *location, founder,*

industry, manufacturer. Les objets partagent des propriétés communes (*founder, manufacturer,* etc.) mais certaines propriétés (*parentCompany, name*) ne sont présentes que pour Audi et Mercedes-Benz. Les propriétés potentiellement manquante de Fiat peuvent être *name, parentCompany, subsidiary, formationYear.* L'approche cherche à ranger dans un ordre pertinent les propriétés potentiellement manquantes d'un objet en trouvant quels objets sont similaires puis en utilisant leurs propriétés pour prédire les propriétés potentiellement manquantes. Dans le cas de Fiat, la liste des objets et leur similarité pourrait être (0.7, Mercedes-Benz), (0.6, Audi). La liste des propriétés manquantes et leur pertinence pourrait être (0.62, *name*), (0.54, *parentCompany*), (0.32, *formationYear*), (0.13, *subsidiary*). La similarité entre objets est basée sur plusieurs mesures tenant compte de la structure du graphe ou encore de la distribution des propriétés. Si beaucoup d'objets similaires à celui considéré partagent des propriétés mais que celles-ci n'apparaissent pas comme propriétés de l'objet considéré, alors ces dernières sont considérées comme manquantes.

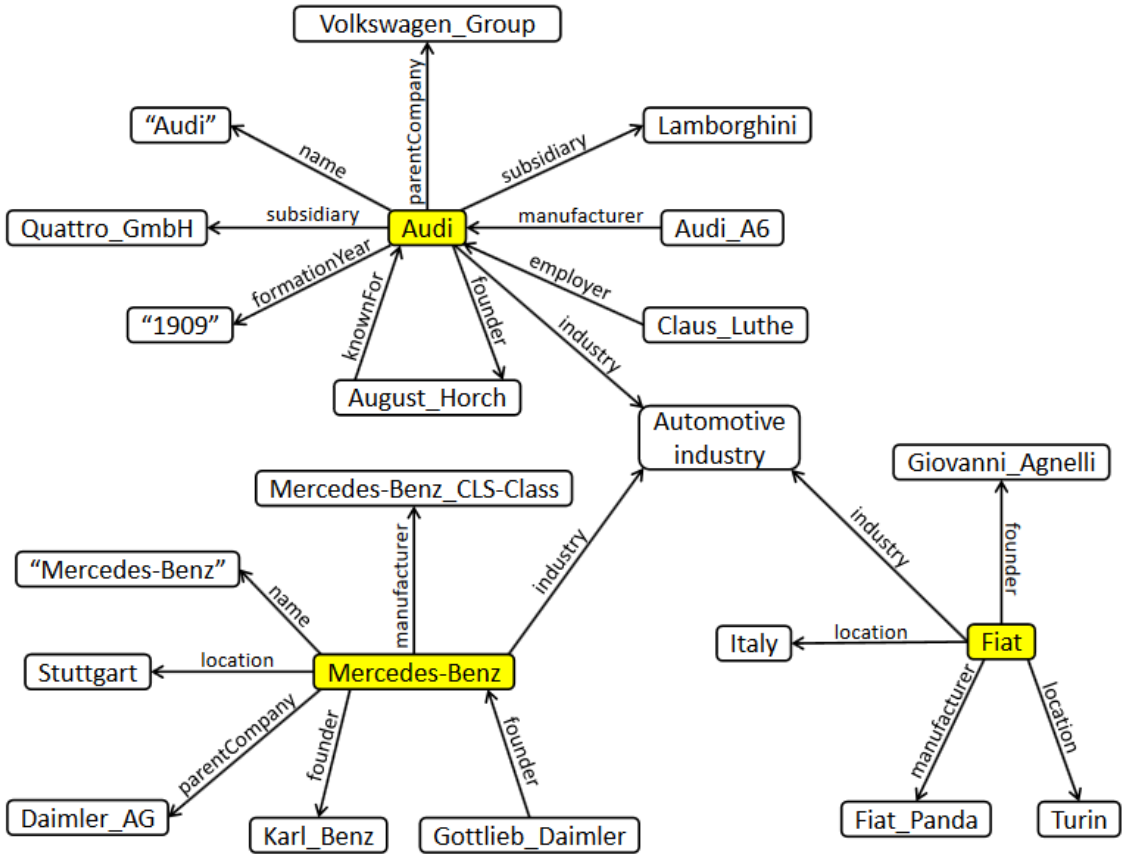


Fig. 6.2 Un exemple d'un petit dataset du LOD décrivant trois ressources, Audi, Mercedes-Benz et Fiat extrait du papier [Simonic *et al.* 2013]

D'autres travaux proposent des solutions pour le problème spécifique de typage d'entités. Par exemple, SDType [Paulheim & Bizer 2014] est un algorithme qui permet

d'ajouter des déclarations de type manquantes. Pour cela, SDType s'intéresse aux propriétés qui relient deux ressources. L'idée est d'utiliser chaque propriété entrante et sortante d'une ressource donnée comme un indicateur du type de la ressource. Pour chaque propriété, la distribution statistique des types du sujet ou de l'objet de la propriété est utilisée pour prédire les types des instances. Il s'agit donc d'une approche de vote (pondérée par propriété), où chaque propriété peut voter sur le type de la ressource à typer en utilisant sa distribution statistique. Par exemple, la probabilité que le sujet de la propriété *location* soit une instance de la classe "Place" est de 0,698. Pour la classe "Building", celle-ci est de 0,34. Et ainsi de suite. Ainsi, on peut calculer la probabilité qu'une ressource soit d'un certain type en moyennant les probabilités obtenues pour chacune des propriétés entrantes et sortantes de la ressource. Il s'agit d'une moyenne pondérée car un poids est calculé pour chaque propriété.

Le système Tìpalo [Gangemi *et al.* 2012] identifie les types les plus appropriés pour une entité en interprétant sa définition donnée en langue naturelle provenant du résumé de sa page Wikipedia associée. Pour cela, il utilise l'outil FRED qui traduit des phrases en représentation RDF ou OWL. Les résultats obtenus sont exploités via des patrons (graph patterns) prédéfinis permettant d'inférer le type d'une entité (ou d'inférer des relations de subsomption entre entités).

[Sleeman & Finin 2013] a pour but de prédire le type d'une instance décrite dans un graphe sémantique (RDF ou autre) en se basant sur un classifieur SVM. Le classifieur utilisé se base sur les propriétés des instances. Les propriétés sélectionnées dans le classifieur sont choisies grâce à une mesure pour calculer le gain d'information d'une propriété, i.e., pour mesurer quels attributs distinguent le mieux une classe étant donné un ensemble de classes (par exemple, "*populationDensityKm*" semble intéressante pour déterminer le type "*Place*", ou bien "*logo*" pour le type "*Organisation*"). Pour être capable d'exploiter plusieurs sources, les propriétés considérées dans le classifieur sont alignées avec les propriétés d'autres ressources (alignement par le nom des propriétés, la forme des valeurs, ou grâce à des synonymes en utilisant WordNet).

Les travaux cités ici s'appuient sur des techniques d'apprentissage automatique [Sleeman & Finin 2013] ou sur des techniques statistiques [Paulheim & Bizer 2014] ou utilisent des connaissances externes [Gangemi *et al.* 2012]. Ils exploitent soit des caractéristiques spécifiques de DBpedia [Gangemi *et al.* 2012] ou sont adaptés à n'importe quelle base de connaissance RDF [Paulheim & Bizer 2014, Sleeman & Finin 2013].

Enfin, d'autres travaux portent sur l'incomplétude relative des valeurs de propriétés. On entend par incomplétude relative, le fait que des valeurs de propriétés multi-valuées soient manquantes. Dans [Yuan *et al.* 2014], des mesures sont proposées pour évaluer l'incomplétude dans les valeurs de propriétés multi-valuées dans plusieurs jeux de données. Le système AMIE [Galárraga *et al.* 2013] traite de l'ajout de nouveaux faits dans une base de connaissances afin de réduire l'incomplétude relative. Cet ajout est basé sur l'application de règles logiques déduites par une approche utilisant des techniques de programmation logique inductive. Ces règles sont créées en adéquation avec les connaissances de la base de connaissances. Un exemple de règle pourrait être

que le mari de la mère d'un enfant est le père de cet enfant.

En conséquence, le Web des données constitue un immense réseau d'informations. Cependant, l'exploitation de ces informations dans le cadre d'applications nécessite d'être conscient de leurs caractéristiques et de trouver des solutions adaptées à leur exploitation.

6.2 Accès aux données du LOD : problème d'hétérogénéité sémantique

Notre objectif étant de rechercher des assertions de propriété d'une ontologie dans le LOD, nous devons trouver des propriétés équivalentes dans l'ontologie et la source du LOD exploitée. Ce problème d'alignement est complexe du fait de l'hétérogénéité des sources : schémas différents et vocabulaire différent. En effet, nous pouvons faire face à de multiples cas : des propriétés correspondantes équivalentes, des problèmes de conversion de mesure, des propriétés correspondantes non explicitées dans le LOD mais obtenables via des transformations d'une ou plusieurs propriétés existantes, des problèmes de valeurs manquantes, des propriétés à agréger (par exemple, moyenner, prendre le maximum, etc.). De ce fait, les correspondances à considérer ne sont en général pas de simples correspondances 1 : 1. Ces correspondances sont complexes car elles doivent considérer les multiples cas évoqués ici, sachant que ceux-ci peuvent se combiner. Nous nous focalisons sur les travaux d'alignement d'ontologies qui cherchent à établir des correspondances complexes.

L'alignement d'ontologie est un processus permettant d'exprimer des correspondances (mappings) entre deux ontologies différentes. Ces correspondances concernent diverses entités ontologiques : correspondances entre classes, entre propriétés, entre instances. Beaucoup de travaux ont été réalisées dans ce domaine ces dernières années [Euzenat & Shvaiko 2013] mais la plupart des outils ne génèrent que des correspondances simples entre entités atomiques. Dans cet état de l'art, nous ne nous intéresserons qu'aux travaux définissant ou générant des correspondances complexes.

[Scharffe 2009, Scharffe *et al.* 2014] définissent des correspondances complexes. Des patrons de correspondance sont présentés comme des outils pour faciliter la définition de correspondances complexes. Certains permettent de spécifier les conditions qu'une entité d'une ontologie doit satisfaire pour être mise en correspondance avec une entité d'une autre ontologie. D'autres indiquent des conversions d'unités de mesure ou une modification du type d'une donnée. La Figure 6.3 montre la classification des patrons définis dans ce cadre. Ce travail est vu comme une première étape dans la définition des correspondances complexes. Les patrons aident à affiner des correspondances simples préalablement calculées.

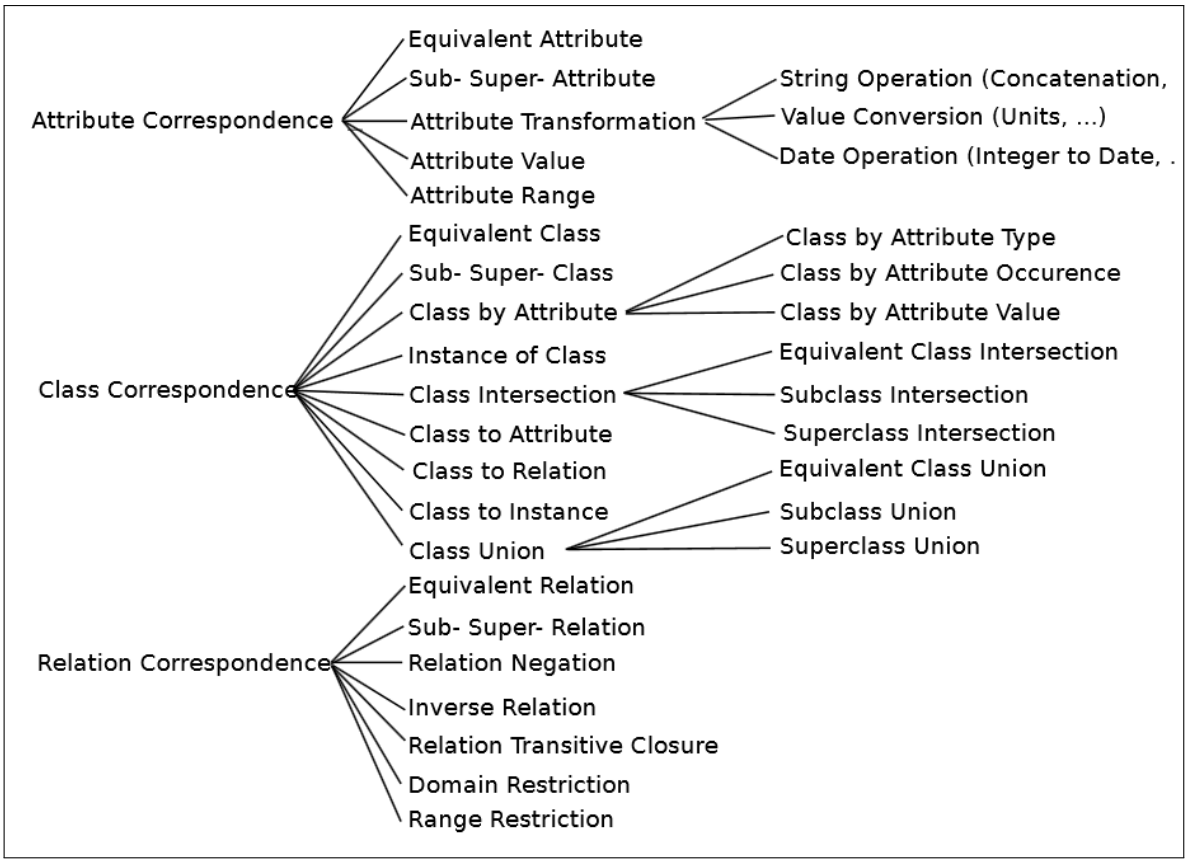


Fig. 6.3 La librairie de patrons proposée par [Scharffe *et al.* 2014]

Les travaux générant des correspondances complexes le font en général à partir de correspondances simples. C'est le cas, par exemple, dans [Pereira Nunes *et al.* 2013] qui porte sur la recherche de correspondances complexes entre propriétés datatype. L'alignement se base sur des techniques d'analyse d'instances de propriétés datatype et se fait en deux temps. Tout d'abord, des correspondances simples sont détectées en comparant les valeurs des propriétés datatype d'une classe source avec celles d'une classe cible. Par exemple, une correspondance entre la propriété source "*eMail*" et la propriété cible "*Electronic Address*" peut être trouvée. La seconde phase trouve des correspondances plus complexes $(1 : n)$ entre les propriétés corrélées identifiées dans la première phase. Par exemple une correspondance pourra être établie entre d'une part les propriétés sources "*FirstName*" et "*LastName*" et d'autre part la propriété cible "*FullName*". Dans ce cas, une fonction d'alignement de propriétés, qui concatène les valeurs de "*FirstName*" et de "*LastName*" pour obtenir la valeur de "*FullName*", est générée.

Une analyse linguistique peut permettre d'établir des correspondances complexes. Le travail de [Ritze *et al.* 2010] s'intéresse à la génération de correspondances entre un concept complexe et des descriptions de propriétés. Pour cela, des techniques de

traitement automatique de la langue sont utilisées sur le nom du concept complexe (source). La structure de l'ontologie cible (domaine et co-domaine de propriétés) aide à exploiter les résultats de ce traitement de la langue. Par exemple, considérons le nom de concept complexe "accepted_paper" d'une ontologie source et une propriété "hasDecision" d'une ontologie cible dont le domaine est "Paper" et le co-domaine est "Decision". Grâce à des techniques de traitement automatique de la langue, on peut déduire que le concept "accepted_paper" est une sous-classe du concept "Paper" de l'ontologie cible. De plus, la nominalisation de "accepted" donne "acceptance" qui est une sous-classe de "Decision" dans l'ontologie cible. On peut donc en déduire une correspondance entre le concept source "accepted_paper" et l'expression cible $\exists hasDecision.Acceptance$.

Nous remarquons donc que l'établissement automatique de correspondances complexes n'est pas une chose aisée. Des éléments sont nécessaires pour parvenir à leur établissement : par exemple il faut que les ontologies traitées aient des instances alignées ou bien qu'une analyse linguistique soit possible.

6.3 Accès aux données du LOD : problème d'accès complexe

De multiples applications concernent les problèmes d'accès complexe : intégration de données, médiation de données, aide à l'accès aux données. Cette section présente divers travaux issus de ces domaines.

6.3.1 Intégration de données

L'intégration de données consiste à combiner des données de différentes sources afin d'en fournir une vue unifiée. Cela peut consister à transférer des données d'une (ou plusieurs) source(s) dans une autre source. Une approche d'intégration de données entre des schémas XML et des schémas relationnels est présentée dans [Popa *et al.* 2002]. Des correspondances entre un schéma source et un schéma cible sont exprimées par un utilisateur. L'approche se déroule ensuite en deux temps. Tout d'abord, les correspondances sont converties en un ensemble de correspondances sémantiques, c'est-à-dire qu'on tient compte des connaissances exprimées, par exemple via les clés étrangères des relations. On en déduit alors que certaines correspondances seront à interpréter ensemble (*mappings logiques*). Par exemple, considérons le cas de subventions d'entreprise où une subvention est donnée à une entreprise pour un projet spécifique. L'utilisateur a spécifié, entre autres, une première correspondance entre les attributs dénotant le nom de l'entreprise dans les deux schémas et une deuxième correspondance entre les attributs dénotant le nom du responsable de la subvention. La source est une base de données relationnelle où une subvention et une entreprise sont liées par une clé étrangère

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

et la cible est un schéma XML où les subventions sont imbriquées dans l'entreprise qui les reçoit. Ainsi, pour intégrer les données d'une entreprise dans la cible, nous devons considérer toutes les subventions (et donc leur responsable) reçues par cette entreprise. De ce fait, il faut interpréter en même temps ces deux correspondances. Le système génère tous les *mappings logiques* cohérents avec les spécifications des schémas considérés. En pratique, beaucoup plus de *mappings logiques* que nécessaire sont générés et un utilisateur doit donc les valider. Par la suite, les *mappings logiques* sont automatiquement convertis en un ensemble de règles, une pour chaque *mapping logique*. Ces règles ont une traduction directe en requêtes dans le langage voulu (XSLT, XQuery ou SQL) et peuvent ainsi transformer les données sources en données cibles.

6.3.2 Médiation de données

La médiation de données vise à faire interopérer des systèmes différents. Une requête posée sur une ressource source ayant son propre schéma doit pouvoir être traduite en une requête similaire sur une ressource cible ayant un tout autre schéma. Plusieurs travaux concernent alors la ré-écriture de requêtes SPARQL de façon à pouvoir les exécuter sur différents jeux de données du LOD.

[Makris *et al.* 2012] présentent une méthode générique de ré-écriture de requêtes SPARQL qui rend le processus d'interrogation des données liées totalement transparent pour l'utilisateur. Cette ré-écriture est faite par rapport à un ensemble de mappings entre ontologies, pré-définis et représentés en OWL [Makris *et al.* 2010]. L'accent est mis sur la richesse des types de mappings considérés, ceux-ci étant établis entre des expressions de classes, de propriétés ou d'individus à l'aide de relations d'équivalence ou de subsomption. Par exemple, considérons les quatre mappings suivants :

$m_1 : src : name \sqsupseteq trg : title$,
$m_2 : src : author \equiv trg : author \circ trg : name$,
$m_3 : src : Science \equiv trg : ComputerScience \sqcup trg : Mathematics$,
$m_4 : src : Pocket \equiv trg : Textbook.(trg : size \leq 14)$.

La Figure 6.4 montre un exemple de procédure de ré-écriture. Les constructeurs utilisés dans la définition des mappings sont associés à des patrons SPARQL. Par exemple, pour le mapping m_3 l'opérateur d'union ($\sqcup$) est associé à l'opérateur `UNION` de SPARQL.

[Correndo *et al.* 2010] proposent un algorithme de ré-écriture de requêtes qui repose sur l'utilisation de règles de ré-écriture de motifs de graphe et exploite l'alignement entre les ontologies associées aux jeux de données. Les règles de ré-écriture peuvent faire état de matchings non triviaux (par exemple, l'association has-author correspond à la composition creator-Info o hasCreator). L'originalité de ce travail réside dans l'expression de contraintes qui conditionnent l'usage des règles. Ces contraintes peuvent porter sur les unités de mesure dans lesquelles sont exprimées les données, ce qui doit déclencher un processus de conversion au moment de la ré-écriture de la

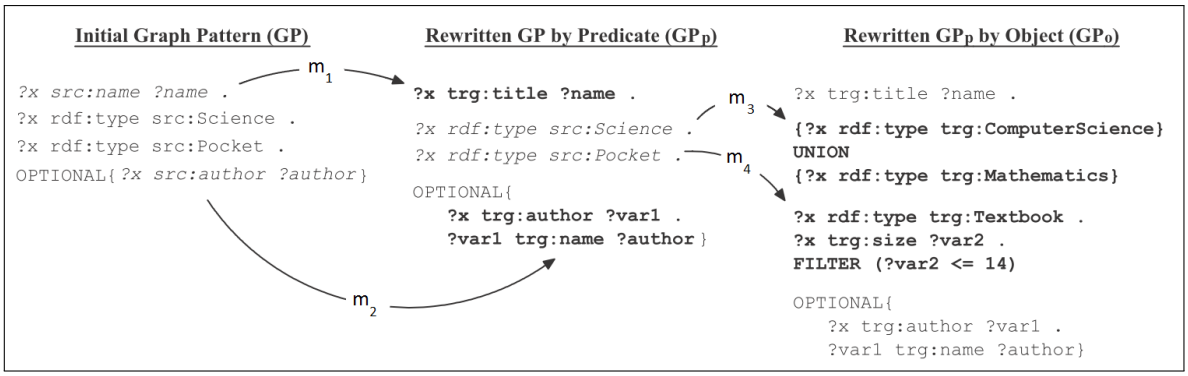


Fig. 6.4 Un exemple de ré-écriture de requête SPARQL en appliquant la méthode décrite dans [Makris *et al.* 2012]

requête, ou sur le format des données (adresse correspondant à une seule valeur ou à plusieurs : adresse rue, code postal et ville). Ces contraintes correspondent donc à des traitements à effectuer sur les données, même si ces derniers restent relativement simples.

Le travail de [Thiéblin *et al.* 2016] a pour but de traduire automatiquement des requêtes SPARQL de type SELECT formulées pour une ontologie source en requêtes SPARQL formulées pour une ontologie cible. L'approche repose sur des alignements complexes exprimés via la syntaxe EDOAL [Euzenat *et al.* 2007] et des règles de transformation. Le mécanisme de traduction se limite à des requêtes formatées, uniquement composées de triplets dont le sujet est une variable.

Enfin, nous citerons le travail de [Gillet *et al.* 2013], qui définit la notion de correspondances complexes pour la ré-écriture de patrons de requêtes. Ces correspondances sont des associations de type $1:n$ ou $n:1$ ou $n:m$ exprimées à l'aide des constructeurs de la logique de description. Les relations modélisées sont les relations Equivalence, Plus général, Plus spécifique. Pour les expérimentations réalisées, les correspondances complexes ont été établies manuellement sur la base de patrons proposés dans la littérature, dont la couverture a été jugée suffisante. La ré-écriture est appliquée à un ensemble de patrons de requêtes. Des règles ont été définies pour traduire une correspondance complexe formalisée en un motif de graphe RDF et guider le processus de ré-écriture.

6.3.3 Facilitation de l'accès aux données

Diverses techniques ont pour but de faciliter l'accès aux données liées, via le développement d'interfaces qui exploitent l'expressivité du modèle de données sous-jacent et du langage de requêtes, tout en cachant leur complexité.

Les systèmes de questions-réponses sont des solutions permettant à un utilisateur d'interroger le web des données liées en formulant des requêtes en langage naturel

de façon assez intuitive [Kaufmann & Bernstein 2010]. Dans beaucoup de ces systèmes, les questions sont associées à des représentations sous forme de triplets RDF. Des sous-graphes sont alors extraits de jeux de données RDF après application d'heuristiques et de mesures de similarité. Cette solution fonctionne assez bien pour des requêtes facilement compréhensibles pour lesquelles la structure de la question peut aisément et automatiquement être associée à une représentation sous forme de triplets. Des correspondances de vocabulaires entre les termes de la question et ceux des jeux de données sont supposées exister. Le système de question-réponse décrit dans [Unger *et al.* 2012] considère des situations plus complexes. Il s'agit de questions en anglais comprenant des comparateurs comme more than ou the most, qui nécessiteront une clause `HAVING` ou `ORDER BY` en SPARQL. L'approche proposée consiste à analyser la question, via des techniques de traitement automatique de la langue, pour produire un modèle de requête SPARQL reflétant la structure de la question, indépendant du domaine d'application, qui sera ensuite instancié.

D'autres travaux, comme celui décrit dans [Shekarpour *et al.* 2011], se donnent pour objectif de faciliter la construction de requêtes SPARQL pour interroger le web des données liées à partir de requêtes mots-clefs, moins ambiguës que des requêtes en langage naturel, et dont le traitement peut être plus efficace. Il s'agit de proposer des réponses les plus pertinentes possibles alors que les relations entre les mots clefs dans la question posée ne sont pas précisées. Par exemple, un utilisateur recherchant les noms des universités en Allemagne donnera les mots-clés suivants : "university" et "Germany". L'accent est mis sur la capacité à rechercher les tuples du LOD satisfaisant la requête mots-clefs quand ils existent. Des patrons de requêtes SPARQL sont associés aux requêtes mots-clefs en fonction du type de recherche : recherche de valeurs de propriétés d'individus, d'individus partageant des propriétés, ou d'associations/compositions d'associations entre individus. Le processus est efficace car les requêtes ne donnent pas lieu à des calculs complexes.

Nous pouvons conclure que la plupart des travaux d'accès complexe à des données se basent sur un alignement pré-défini. En général, les correspondances entre la source et la cible sont données manuellement et doivent être interprétées. Pour cela, la plupart des approches se basent sur des patrons de requête.

6.4 Positionnement de notre travail par rapport à l'état de l'art

Contrairement aux travaux relatifs à l'incomplétude du Web des données, notre problème ne consiste pas à détecter ou évaluer l'incomplétude. Nous cherchons des solutions pour pallier l'incomplétude des valeurs de propriétés au sein d'une unique base de connaissances dans laquelle les valeurs manquantes ne peuvent pas être inférées.

Dans notre contexte, nous nous intéressons à des correspondances sur lesquelles des traitements complexes doivent être effectués. Par exemple, les propriétés du LOD peuvent ne pas être strictement équivalentes aux propriétés recherchées de l'ontologie. Dans certains cas, une propriété ontologique peut correspondre à un ensemble de propriétés du LOD intervenant dans un calcul. L'exécution de ce calcul permettra d'obtenir des données équivalentes aux données de l'ontologie recherchées. Or, dans tous les travaux d'alignement décrits, nous constatons que les correspondances complexes sont toujours définies à partir de correspondances simples. C'est sur ce point que nous nous différencions. En effet lorsque l'entité cible avec laquelle l'entité source est associée est le résultat d'un traitement complexe mettant en œuvre successivement différents agrégats, la correspondance ne peut pas être dérivée de correspondances simples. Les entités faisant l'objet de tels traitements complexes ne peuvent pas être découvertes par un système d'alignement.

Notre travail est proche de [Makris *et al.* 2012] du fait de la représentation en OWL des mappings considérés et donc de la possibilité d'exprimer également des contraintes OWL sur les éléments mis en correspondance, mais nous nous intéressons à des mappings plus complexes qui peuvent être exprimés à l'aide de plusieurs agrégats appliqués successivement. De plus, nos correspondances doivent respecter les connaissances de l'ontologie telles que la cardinalité des propriétés par exemple. C'est une difficulté dont les travaux de médiation de données, transformant une requête SPARQL source en requête SPARQL cible, n'ont pas à se préoccuper. Cela peut potentiellement impliquer d'agréger des données sources. Or, les opérateurs d'agrégation de SPARQL ne sont présents que depuis la version 1.1 qui est assez récente (recommandation W3C du 21 Mars 2013)[1].

Les travaux de question-réponse sont intéressants car ils visent à simplifier l'accès aux données du LOD. Ils supposent toutefois que, modulo des mappings donnés ou découverts, l'information recherchée est explicitement représentée (ou peut être facilement calculée par application d'une fonction agrégation), qu'elle ne résulte pas d'un traitement à réaliser portant sur plusieurs données différentes, et qu'elle peut donc être obtenue par une seule requête. L'idée de la construction d'un modèle de requête SPARQL par analyse linguistique des questions, proposée dans [Unger *et al.* 2012], est toutefois intéressante. De façon analogue, nous proposons de construire des requêtes SPARQL à partir de patrons qui sont ensuite instanciés. Nos patrons sont basés sur les modèles de correspondance et d'accès aux données qui intègrent la spécification de traitements complexes.

Nous nous différencions des travaux qui visent à faciliter l'accès aux données liées par le fait que nous nous intéressons, en plus, à l'obtention d'informations non explicitement représentées, nécessitant la recherche au préalable de données sur lesquelles des traitements complexes doivent être effectués. Les données exploitées peuvent ne

[1]https://www.w3.org/TR/sparql11-overview/

pas être équivalentes ou similaires aux données recherchées mais correspondre à des données intervenant dans un calcul. La requête à écrire pour obtenir les données recherchées issues de traitements complexes doit bien souvent comprendre plusieurs sous-requêtes imbriquées, qui ne peuvent être dérivées de l'analyse du contenu des questions posées.

En conclusion, notre travail se distingue des travaux de l'état de l'art par la prise en compte de traitements complexes permettant d'accéder à des informations non explicitement représentées dans le LOD, tels qu'un calcul entre plusieurs propriétés dont l'exécution permet d'obtenir des données équivalentes aux données de l'ontologie recherchées ou encore l'application d'agrégats, et par la spécification de ces traitements à leur exécution, et ce d'une façon totalement transparente. Ce problème n'a, à notre connaissance, pas encore été traité dans la littérature.

Conclusion

Ce chapitre a dressé un état de l'art des techniques existantes d'accès aux données du LOD. Nous avons positionné notre travail par rapport à des travaux de différents domaines : définition et génération de correspondances complexes, ré-écriture de requêtes, aide à l'accès aux données. La suite de cette thèse présente le modèle d'acquisition des valeurs de propriétés du LOD à partir d'une ontologie OWL que nous proposons puis le processus de génération de requêtes SPARQL qui prend appui sur ce modèle.

Chapitre 7

Modèle d'acquisition de données du LOD

Sommaire

Ce chapitre présente le modèle d'acquisition [Alec *et al.* 2016a, Alec *et al.* 2016d] utilisé dans l'approche Saupodoc et qui permet de spécifier les correspondances entre notre ontologie et un jeu de données du LOD, puis de générer automatiquement les requêtes SPARQL associées.

Le modèle présenté n'est pas propre à l'approche Saupodoc. Il est exploitable pour toute application dont le but est de peupler une ontologie cible avec des assertions de propriétés à partir de données du LOD. Le peuplement de l'ontologie cible doit tenir compte des caractéristiques des propriétés du LOD : propriétés multiples, valeurs manquantes, propriétés multi-valuées.

Notre travail porte donc sur la collecte de données du LOD compte tenu du fait que l'information qui y est explicitement représentée ne correspond pas forcément à la structure et au format de l'information recherchée, que les données comportent des redondances, et qu'elles peuvent, en plus, être incomplètes. Ceci nécessite de spécifier précisément les traitements portant sur les données multiples disponibles dans le LOD qui vont permettre d'obtenir l'information recherchée. Ces traitements

peuvent être complexes, complexifiant d'autant plus l'interrogation des données en SPARQL, le langage standard d'interrogation des données RDF [Harris *et al.* 2013]. Afin de faciliter ce processus, et d'éviter au concepteur la phase d'écriture des requêtes en SPARQL, nous proposons un modèle de correspondances et d'accès aux données du LOD à partir d'une ontologie. Ce modèle, complété par des mécanismes de traitement des valeurs de propriétés acquises pour satisfaire les contraintes de l'ontologie à peupler, est adapté à la spécification de traitements complexes.

Ce chapitre est structuré de la façon suivante. Dans un premier temps, nous présentons des exemples de traitements complexes dont la mise en œuvre motive notre travail. Nous présentons ensuite le modèle d'acquisition des valeurs de propriétés du LOD à partir d'une ontologie OWL. Enfin, nous concluons.

7.1 Cas d'utilisation illustrant nos objectifs

Dans cette section, nous donnons des exemples de cas d'utilisation ayant motivé notre travail.

Exemple 3. Supposons que l'on recherche, dans DBpedia, la valeur de la propriété supPopulationKm2 d'une ontologie myOnto, correspondant à la densité maximale de la population au km^2 d'un lieu. Une analyse des données de DBpedia montre (cf. figure 7.1) qu'un lieu peut avoir une propriété exprimant la densité de population avec éventuellement plusieurs valeurs et également d'autres propriétés, comme la superficie et le nombre d'habitants, qui peuvent aussi permettre de calculer la densité de la population. Si on veut connaître la valeur de la densité maximale de la population au km^2 d'un lieu, il faut utiliser toutes ces données. La valeur de la propriété cherchée supPopulationKm2 de myOnto correspond donc à Max(Max(populationDensity),Max(populationTotal)/Min(areaTotal)) dans DBpedia. L'obtention de la valeur de supPopulationKm2 nécessite de collecter les valeurs de populationTotal et areaTotal en ne retenant que la valeur maximale et minimale respectivement, d'extraire la densité maximale donnée par populationDensity puis celle calculée à partir du rapport entre Max(populationTotal) et Min(areaTotal), puis de ne retenir que la valeur maximale de ces deux densités. Cela passe par l'écriture d'une requête SPARQL avec des requêtes imbriquées, des sous-requêtes reliées par l'opérateur `UNION`, l'usage d'opérateurs d'agrégation et de l'opérateur `BIND` pour spécifier les calculs ainsi que les changements de format nécessaires. Cette requête, dont un exemple appliqué au Canada est présenté ci-dessous, n'est pas simple. On supposera connue la correspondance entre l'individu représentant le Canada dans myOnto, noté myOnto:Canada, et celui de DBpedia, noté dbr:Canada.

```
CONSTRUCT { myOnto:Canada myOnto:supPopulationKm2 ?val0 }
WHERE{
   {
      SELECT (MAX(?val2) AS ?val1)
      WHERE {
         {
            SELECT (MAX(?val3) AS ?val2)
```

```
            WHERE { dbr:Canada dbo:populationDensity ?val3. }
        } UNION {
            {
                SELECT (MIN(?val4) AS ?val3a)
                WHERE {dbr:Canada <dbo:PopulatedPlace/areaTotal> ?val4.}
            }
            {
                SELECT (MAX(?val4) AS ?val3b)
                WHERE { dbr:Canada dbo:populationTotal ?val4. }
            }
            BIND (?val3b/?val3a AS ?val2)
        }
    }
    BIND (xsd:double(?val1) AS ?val0)
}
```

dbo:PopulatedPlace/areaTotal	▪ 9982034.326224321 ▪ 9984670.0
dbo:populationDensity	▪ 3.204648 (xsd:double) ▪ 3.410000 (xsd:double)
dbo:populationTotal	▪ 35675834 (xsd:integer)

Fig. 7.1 Propriétés de Canada dans DBpedia

dbp:janPrecipitationDays	▪ 15 (xsd:integer)

Fig. 7.2 Propriétés de Sarnia dans DBpedia

dbp:janPrecipitationDays	▪ 20.600000 (xsd:double) ▪ 20.800000 (xsd:double)

Fig. 7.3 Propriétés de Juneau dans DBpedia

dbo:part	▪ dbr:Brooke-Alvinston ▪ dbr:Dawn-Euphemia ▪ dbr:Enniskillen,_Ontario ▪ dbr:Lambton_Shores ▪ dbr:Oil_Springs,_Ontario ▪ dbr:Petrolia,_Ontario ▪ dbr:Point_Edward,_Ontario ▪ dbr:Sarnia ▪ dbr:St._Clair,_Ontario ▪ dbr:Warwick,_Ontario ▪ dbr:Plympton–Wyoming
is dbo:isPartOf of	▪ dbr:Brooke-Alvinston ▪ dbr:Dawn-Euphemia ▪ dbr:Enniskillen,_Ontario ▪ dbr:Kettle_Point_44,_Ontario ▪ dbr:Lambton_Shores ▪ dbr:Oil_Springs,_Ontario ▪ dbr:Petrolia,_Ontario ▪ dbr:Point_Edward,_Ontario ▪ dbr:Sarnia ▪ dbr:St._Clair,_Ontario ▪ dbr:Warwick,_Ontario ▪ dbr:Walpole_Island_Indian_Reserve_No._46 ▪ dbr:Sombra,_Ontario ▪ dbr:Grand_Bend ▪ dbr:Aamjiwnaang_First_Nation ▪ dbr:Uttoxeter,_Ontario ▪ dbr:Plympton–Wyoming

Fig. 7.4 Propriétés de Lambton County dans DBpedia

Exemple 4. La recherche porte dans ce second cas sur la valeur de precipitationDays, une propriété fonctionnelle dans myOnto qui représente le nombre de jours pluvieux caractérisant la classe Weather. Les instances de la classe Weather correspondent aux données météorologiques d'un lieu donné pour un mois donné. Ainsi, myOnto:WeatherJanuarySarnia est un exemple d'instance de Weather correspondant aux données météorologiques de Sarnia en janvier. Cette instance est caractérisée par les deux propriétés suivantes représentées sous forme de triplets : <myOnto:Sarnia myOnto:hasWeather myOnto:WeatherJanuarySarnia> <myOnto:WeatherJanuarySarnia myOnto:concernMonth myOnto:January>. Supposons que la recherche porte sur la valeur du nombre de jours pluvieux à Sarnia en janvier, soit la valeur de PrecipitationDays de l'instance myOnto:WeatherJanuarySarnia. Une analyse des

données de DBpedia (cf. Figures 7.2 et 7.3) montre que la propriété janPrecipitationDays caractérise les lieux décrits dans DBpedia. Toutefois, cette propriété est parfois multi-valuée, et dans ce cas on pourra être intéressé par la moyenne des valeurs indiquées. En généralisant, on peut donc établir une correspondance entre precipitationDays avec la contrainte sur son domaine restreignant cette propriété aux instances liées par la relation concernMonth au mois de janvier dans myOnto et la moyenne des valeurs de janPrecipitationDays dans DBpedia de façon à ne collecter qu'une seule valeur puisque la propriété est fonctionnelle. Dans ce cas, la complexité de la mise en correspondance provient de l'expression de la contrainte associée à la propriété dans l'ontologie pour laquelle on cherche la valeur, à laquelle s'ajoute le calcul de moyenne. L'expression de contraintes s'explique par le fait que les modèles dans l'ontologie et dans les sources de données, comme DBpedia, sont différents.

Notons que la correspondance concernant precipitationDays décrite ci-dessus ne retournera le résultat attendu que si janPrecipitationDays est valuée dans DBpedia. Or, les lieux décrits dans DBpedia n'ont pas tous une valeur pour cette propriété. Nous souhaitons, dans ce cas, trouver des valeurs de substitution qui, bien qu'approximatives, pourraient être utiles (par exemple, dans le cas de Lambton County où cette propriété est absente, le nombre de jours pluvieux de Lambton County peut être remplacé par la valeur de cette propriété pour une de ses sous-partie géographique cf. Figure 7.4). Ce processus de substitution va toutefois compliquer l'écriture des requêtes (cf. requête ci-dessous correspondant à l'exemple cité).

```
CONSTRUCT { myOnto:WeatherJanuaryLambtonCounty
    myOnto:precipitationDays ?val0 }
WHERE{
   {
      SELECT (AVG(?val2) AS ?val1)
      WHERE{

         SELECT ?ind1 (AVG(?val3) AS ?val2)
         WHERE {
            {?ind1 dbo:isPartOf dbr:Lambton_County} UNION
            {dbr:Lambton_County dbo:part ?ind1}
            ?ind1 dbp:janPrecipitationDays ?val3.
         }
         GROUP BY ?ind1
      }
   }
   BIND (xsd:double(?val1) AS ?val0)
}
```

7.2 Modèle d'acquisition de valeurs de propriétés du LOD à partir d'une ontologie OWL

On considère une ontologie source O_s et une ontologie cible O_t. O_s est une ontologie OWL. O_t une ontologie fournissant un accès à des sources RDF, telles qu'un jeu de données du LOD. Chaque ontologie est définie comme un tuple ($\mathcal{C}$, $\mathcal{P}$, $\mathcal{I}$, $\mathcal{A}$) où $\mathcal{C}$ est l'ensemble des classes, $\mathcal{P}$ l'ensemble des propriétés (datatype, objet et annotation)

caractérisant les classes, $\mathcal{I}$ un ensemble d'instances et d'assertions de propriété, et $\mathcal{A}$ un ensemble d'axiomes.

Cette section définit un modèle d'acquisition des propriétés de O_t utiles pour valuer des propriétés de l'ontologie source $\mathcal{O}_s$ supposées connues. Remplir cette tâche impose de résoudre deux problèmes : (1) trouver l'individu i_t de $\mathcal{O}_t$ correspondant à un individu i_s de O_s dont on veut valuer une propriété, (2) trouver les propriétés O_t, dites propriétés cibles, correspondant à la propriété de l'ontologie source $\mathcal{O}_s$, dite propriété source, à valuer. Dans le cas de l'approche SAUPODOC, le problème (1) est résolu grâce à l'application de DBpedia Spotlight, permettant d'associer l'entité décrite dans un document à sa page DBpedia correspondante. De ce fait, dans cette thèse, nous nous intéressons uniquement au problème (2). Nous supposons donc qu'un travail effectué en amont a conduit à un ensemble $Ind_{s,t}$ de couples (i_s,i_t) où un individu de l'ontologie source correspond à un individu de l'ontologie cible. Nous avons défini un modèle d'acquisition de valeurs de propriétés d'un jeu de données du LOD composé de trois sous-parties :

- un modèle qui établit une correspondance entre une propriété source et une propriété cible,
- un modèle qui définit l'accès aux valeurs des propriétés recherchées,
- des mécanismes de traitement des valeurs finales acquises, permettant entre autres d'agréger de différentes manières les informations obtenues via les modèles de correspondance et d'accès.

7.2.1 Modèle de correspondance

Le modèle de correspondance définit des correspondances (PE_s,PE_t) entre une Expression de Propriété source (PE_s) et une Expression de Propriété cible (PE_t). Une correspondance est applicable pour tout couple tel que :

$$\begin{cases} (i_s, i_t) \in & Ind_{s,t} \\ i_s \in & domaine(PE_s) \end{cases}$$

Dans ce qui suit, on appellera *propriété datatype*, une propriété dont le co-domaine est un littéral, et *propriété objet*, une propriété dont le co-domaine est une expression de classes. Définissons tout d'abord la notion d'expression de propriétés (PE_s) dans $\mathcal{O}_s$.

Définition 1. *Une expression de propriété dans $\mathcal{O}_s$ (PE_s), est une propriété objet (op) ou une propriété datatype (dp), sur le domaine desquelles on peut exprimer une restriction (notée $PE_s.Restrict(d)$), en OWL DL.*

PE_s = *op* / *dp* / $PE_s.Restrict(d)$

Exemple 5. La propriété objet country est une PE_s. La propriété datatype latitude est une PE_s. La propriété precipitationMm.(concernMonth VALUE January) est une PE_s. Elle concerne la propriété datatype precipitationMm dont le domaine (Weather) est contraint par (concernMonth VALUE January) exprimée en syntaxe Manchester [Horridge *et al.* 2006]. Elle permet de valuer la propriété precipitationMm uniquement pour les i_s correspondant à des données météo qui concernent le mois de Janvier.

Nous définissons maintenant la notion d'expression de propriétés (PE_t) dans $\mathcal{O}_t$, en débutant par la notion de propriété élémentaire sur laquelle la définition de PE_t s'appuie.

Définition 2. *Une propriété élémentaire p_e est une propriété dans $\mathcal{O}_t$ ou son inverse.*
$p_e = op \mid dp \mid op^{-1}$

Définition 3. *Une expression de propriété dans $\mathcal{O}_t$ (PE_t) est une propriété élémentaire (p_e) dans $\mathcal{O}_t$, ou une expression (f) utilisant une ou plusieurs expressions de propriété dans $\mathcal{O}_t$. Une PE_t peut inclure des contraintes sur le domaine de la propriété ($PE_t.Constr(d)$), sur son co-domaine ($PE_t.Constr(r)$) ou sur un élément mis en jeu dans une contrainte de co-domaine ($PE_t.Constr(f(Constr(r)))$).*
$PE_t = p_e \mid f(PE_t) \mid f(PE_t,\ PE_t) \mid PE_t.Constr(d) \mid PE_t.Constr(r) \mid PE_t.Constr(f(Constr(r)))$

Par récursivité, une PE_t peut être une fonction de n PE_t. Par exemple, $PE_{t1} = f(PE_{t2},\ f(PE_{t3},\ PE_{t4})) = f(PE_{t2},PE_{t3},PE_{t4})$. Nous donnons ci-dessous des explications sur les différents éléments intervenant dans la définition d'une PE_t.

La fonction f est spécifiée par le concepteur via une interface. Lorsque f est unaire, il doit indiquer s'il s'agit d'une opération d'agrégation ou de transformation et, pour chacune d'elles, il doit préciser sa nature. Pour une opération d'agrégation, il doit choisir entre le comptage d'un nombre d'éléments, une somme, le calcul d'un minimum, d'un maximum, d'une moyenne, etc. Pour une opération de transformation de valeurs, le choix s'effectue entre un calcul mathématique, le calcul de la longueur d'une chaîne de caractères, un changement de majuscules en minuscules ou l'inverse, le calcul de la valeur absolue d'un nombre, etc. Lorsque f est n-aire, le concepteur doit indiquer si la fonction correspond à une opération ensembliste (union ou différence) ou de transformation (calcul mathématique, concaténation, etc.).

Constr représente n'importe quelle contrainte exprimable sous la forme d'un motif de graphe SPARQL (Graph Pattern en anglais) correspondant à un ensemble de motifs de triplets (et éventuellement un filtre). $Constr(f(Constr(r)))$ est une contrainte portant sur l'agrégation (f) d'une variable mise en jeu dans une contrainte de co-domaine définie au préalable. Le concepteur doit la préciser comme indiqué ci-dessus. Par exemple, le co-domaine r doit être lié à des éléments x (contrainte de co-domaine) tels que la somme de ces x doit être supérieure à 10.

La valuation d'une PE_t se définit comme suit : $\text{val}(PE_t) = \text{val}(p_e) \mid f(\text{val}(PE_t)) \mid f(\text{val}(PE_t), \text{val}(PE_t)) \mid \text{val}(PE_t.Constr(d)) \mid \text{val}(PE_t.Constr(r)) \mid \text{val}(PE_t.Constr(f(Constr(r))))$

Les exemples de propriétés cibles donnés dans la suite de ce chapitre correspondent à des propriétés de DBpedia.

Exemple 6. Avg(janPrecipitationMm, janRainMm) est une PE_t dont la valeur est la moyenne de toutes les valeurs de l'union des propriétés janPrecipitationMm et janRainMm. Pour s'assurer que cette moyenne ne prendra pas en compte les valeurs négatives erronées fréquentes dans DBpedia, on peut introduire une contrainte sur le co-domaine qui permettra de rejeter les valeurs négatives des p_e, par exemple `FILTER(r >=0)`.

Exemple 7. *artist* est une propriété de DBpedia dont le domaine est MusicalWork et le co-domaine est Agent. De ce fait, $artist^{-1}$ est une p_e de DBpedia (donc par extension une PE_t) qui, associée aux contraintes de co-domaine `r rdf:type dbo:Album` et `r dbo:genre ?gen`, correspond à des noms d'albums de chanteurs (ces albums ayant un genre). L'ajout de la contrainte `COUNT(DISTINCT ?gen)>=3` portant sur la variable `?gen` mise en jeu dans les contraintes citées précédemment impose qu'il y ait au moins 3 genres différents.

En résumé, une PE_t peut être vue comme une propriété de $\mathcal{O}_t$ dont la définition peut être très complexe car elle peut combiner différentes fonctions f et/ou des contraintes. De plus, elle peut ne pas être explicitement représentée dans $\mathcal{O}_t$ mais être le résultat d'une construction. Le concepteur définit des correspondances entre les propriétés dans $\mathcal{O}_s$ et dans $\mathcal{O}_t$ en se basant sur le modèle que nous venons de présenter. Ce modèle tel que nous l'avons décrit, n'est toutefois pas suffisant car, d'une part il ne prend pas en compte les connaissances représentées dans $\mathcal{O}_s$, comme par exemple, la restriction de cardinalité de certaines PE_s. D'autre part, il ne tient pas compte du problème de l'absence possible de valeurs des propriétés dans $\mathcal{O}_t$. Nous proposons des solutions à ces deux types de problèmes dans les sections suivantes. Un modèle de spécification de chemins d'accès aux propriétés permet de pallier le problème des valeurs manquantes. Il fait l'objet de la section 7.2.2. Des mécanismes de traitement des valeurs de propriétés collectées, pouvant être multiples, sont présentés en section 7.2.3.

7.2.2 Modèle de spécification de chemins d'accès à des propriétés

La section précédente a présenté les correspondances entre propriétés. Nous nous intéressons maintenant à la façon d'accéder à la valeur (ou aux valeurs) des PE_t. L'accès à une PE_t, non explicitement représentée dans $\mathcal{O}_t$, correspond à l'accès à l'ensemble des éléments permettant de la définir. Nous présentons successivement les différents types d'accès possibles aux valeurs, puis la notion de chemin d'accès et enfin les algorithmes développés pour parcourir les chemins d'accès à une PE_t dans le LOD.

Types d'accès associés à une PE_t

Le modèle de correspondance requiert un accès aux valeurs de tous les éléments définissant une PE_t. Lorsque les éléments entrant dans la définition d'une PE_t sont des propriétés valuées de i_t, l'accès est direct et ne pose aucun problème. En revanche, lorsque ce n'est pas le cas, le LOD comportant de nombreuses propriétés sans valeurs, ceci peut poser problème pour l'application exploitant les données recherchées. Certaines applications ont besoin de disposer de valeurs, même si celles-ci ne sont qu'approximatives. Nous avons travaillé sous cette hypothèse. Ainsi, lorsqu'une propriété d'un individu i_t n'est pas valuée, nous proposons de rechercher la valeur de cette même propriété pour un (ou d') autre(s) individu(s) que i_t, en considérant ces valeurs comme des approximations de la valeur manquante. Par exemple, si on ne trouve pas la température moyenne en Alaska, on peut rechercher la température moyenne de Juneau, sa capitale. La valeur obtenue pour Juneau sera une approximation de la valeur cherchée pour Alaska. La propriété caractérise ici un autre individu que i_t. L'accès à cette propriété pour i_t n'est donc pas direct puisqu'il faut d'abord atteindre (par un certain chemin) un autre individu que i_t pour obtenir une valeur pour la propriété. Ceci nous a conduit à définir deux types d'accès à une PE_t, un accès direct et un accès composé. Nous donnons ci-dessous la définition des différents types d'accès à une PE_t, accompagnée d'un exemple. Cette définition est basée sur la notion de chemins qui fait l'objet de la section 7.2.2.

Définition 4. *L'accès à une PE_t est dit **composé** si on y accède par un chemin de propriétés d'ordre n ($n \geq 1$), **direct** sinon.*

Exemple 8. Soit $PE_t = \text{Avg}(\text{janPrecipitationMm})$ correspondant à la moyenne des valeurs de janPrecipitationMm. Son accès est direct pour Abu_Dhabi et composé pour Sri_Lanka (cf. figure 7.5).

Chemins d'accès à une PE_t

Un chemin de propriétés est défini par rapport à un chemin élémentaire de propriétés, qu'il nous faut donc d'abord préciser.

Définition 5. *Un chemin élémentaire ($chemin_n^{elem}$) de propriétés d'ordre n ($n \geq 1$) est une composition de propriétés objets ($p^1.p^2...p^n$). Pour un individu cible i_t donné, un chemin élémentaire de propriétés permet d'accéder à un ensemble d'individus cibles I_t dans $\mathcal{O}_t$.*

L'accès par un chemin élémentaire à un ensemble d'individus cibles s'explique par la présence éventuelle de propriétés multi-valuées.

Exemple 9. Dans la figure 7.5, country.capital est un chemin élémentaire de propriétés d'ordre 2 ($p^1.p^2$). Appliqué sur l'individu Cephalonia dans DBpedia, il permet d'accéder à un ensemble I_t ne contenant que l'individu Athens. capital est un chemin élémentaire de propriétés d'ordre 1 (p^1). Appliqué sur l'individu Sri_Lanka dans DBpedia, il permet

d'accéder à l'ensemble $I_t = \{\text{Colombo; Sri\_Jayawardenepura\_Kotte}\}$, car la propriété capital est multivaluée pour Sri_Lanka.

Définition 6. *Un chemin de propriétés d'ordre n ($n \geq 1$) est un chemin élémentaire ou un ensemble de chemins élémentaires, éventuellement muni d'une contrainte sur I_t et/ou sur i_t.*
$chemin_n = chemin_n^{elem} \;/\; Ens(chemin_n^{elem}) \;/\; chemin_n.Constr(i_t) \;/\; chemin_n.Constr(I_t)$

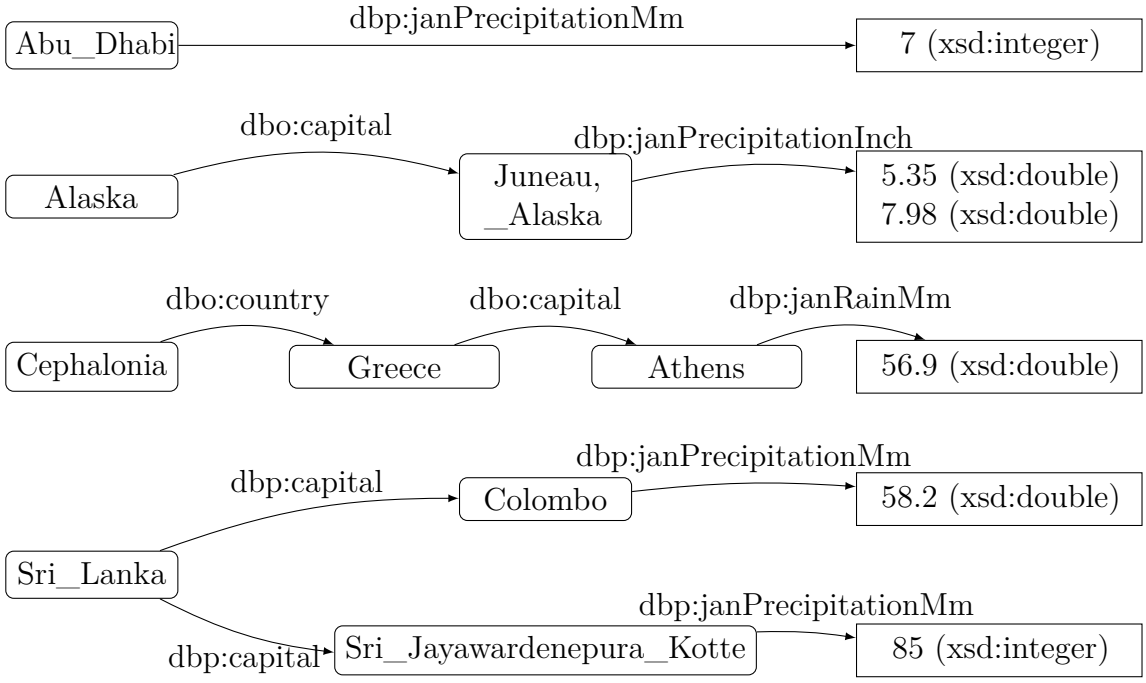


Fig. 7.5 Exemples de chemins d'accès à des précipitations dans DBpedia

Comme vu dans la définition 6, un chemin peut être un ensemble de chemins élémentaires. Cela sera souvent le cas lorsque le concepteur veut faire appel dans le chemin à une propriété redondante. Par exemple, *part*, $isPartOf^{-1}$ et *p* sont trois façons d'exprimer la sous-partie dans DBpedia.

Exemple 10. Le tableau 7.1 représente un chemin de propriétés d'ordre 2 composé d'un ensemble de cinq chemins élémentaires (chaque ligne correspond à un chemin élémentaire). Ce chemin sera utilisé pour accéder à une PE_t permettant l'obtention de données météorologiques sur Dubai. Les valeurs météorologiques qui peuvent être ainsi obtenues sont celles de villes, auxquelles on accède par les propriétés listées en colonne 2. Ces villes sont des sous-parties géographiques de régions obtenues par les propriétés de la colonne 1, correspondant à des régions dont Dubai est soit la plus grande ville soit à l'ouest ou au nord. Les données météorologiques ainsi accessibles sont supposées être de bonnes approximations des données météorologiques de Dubai.

Propriétés régions (p^1)	Propriétés sous-parties (p^2)
$dbp{:}west^{-1}$	$dbo{:}part$
$dbp{:}north^{-1}$	$dbo{:}part$
$dbp{:}north^{-1}$	$dbo{:}isPartOf^{-1}$
$dbp{:}largestCity^{-1}$	$dbo{:}isPartOf^{-1}$
$dbo{:}largestCity^{-1}$	$dbo{:}isPartOf^{-1}$

Tableau 7.1 Chemin appliqué sur Dubai pour une PE_t visant à obtenir des données météorologiques

D'après la définition 6, des contraintes sur i_t et/ou I_t sont possibles. En somme, une contrainte de chemin peut porter sur l'individu initial, ou sur l'ensemble d'individus après l'application du chemin sur l'individu initial.

Exemple 11. Supposons qu'on veuille connaître le pays d'origine d'un film. La PE_t correspondra à l'union des propriétés correspondantes, par exemple *country* ou encore *nationality* dans DBpedia. En cas de valeur manquante, nous supposerons que la nationalité du réalisateur est une bonne approximation de cette propriété. Ainsi, Windy_City_Heat est un film dont on ignore le pays d'origine dans DBpedia. Mais, sachant que Windy_City_Heat.director.nationality = Americans, on pourra en déduire que Windy_City_Heat est un film américain.
Contrainte sur i_t : Maintenant, si le concepteur estime qu'avec l'émergence de la mondialisation, beaucoup de réalisateurs réalisent de nos jours des films ailleurs que dans leur pays d'origine, il peut ajouter une contrainte exprimant que cette approximation n'est valable que si le film n'est pas récent (par exemple, avant 2000). Cette contrainte porte donc sur le film (i_t), et peut être exprimée par le motif de graphe suivant :
i_t `dbp:released ?year. FILTER (?year <= 2000).`
Contrainte sur I_t : D'un autre côté, le concepteur peut considérer que les approximations trouvées seront assez bonnes en général sauf pour les films de certains réalisateurs connus comme James Cameron (canadien) ou Ridley Scott (britannique) qui ont fait beaucoup de films américains. La contrainte porte donc ici sur le réalisateur (I_t) et peut être exprimée par le motif de graphe suivant :

```
I_t rdfs:label ?name. FILTER (LANGMATCHES(LANG(?name), "en") &&
!regex(?name, "Ridley Scott") && !regex(?name, "James Cameron"))
```

Algorithmes de parcours des chemins d'accès

Le concepteur définit tous les chemins d'accès (un maximum) pour chaque PE_t impliquée dans des correspondances avec des propriétés de $\mathcal{O}_s$, en les ordonnant par pertinence.

L'algorithme 1 implémente cette idée. Le principe est de parcourir les chemins menant à une PE_t par ordre de priorité jusqu'à ce qu'on obtienne une valuation de PE_t. L'algorithme appelle la fonction userAlgo spécifiant les chemins par ordre de priorité.

Un exemple d'implémentation de la partie switch est présentée. Dans l'expression des chemins de ce switch, `getVal()` est une fonction renvoyant les valeurs des propriétés considérées. Par exemple, l'expression du cas 2 renvoie les valeurs de la PE_t considérée pour l'ensemble des individus correspondant à la capitale de i_t. L'expression *subparts* des cas 4 et 8 correspond aux propriétés DBpedia isPartOf^{-1}, part et p. En conséquence, le chemin considéré dans le cas 4 correspond à un ensemble de trois chemins élémentaires. Enfin, notons que ?*prop*, dans ces chemins, signifie n'importe quelle propriété DBpedia.

Un même ensemble de chemins peut être utilisé pour plusieurs PE_t. Le paramètre `maxNbAttempts`, fixé par le concepteur, indique le nombre maximal de chemins de l'ensemble à explorer pour une PE_t particulière. Par exemple, la partie switch présentée ici est réutilisable pour la recherche de valeurs de la latitude, de la longitude, des températures, des précipitations par mois, etc. mais pas forcément avec la même valeur de `maxNbAttempts`.

Algorithme 1 : Algorithme pour extraire des données de $\mathcal{O}_t$

Entrée : i_t l'individu cible
Sortie : info à stocker dans $\mathcal{O}_s$
Param : maxNbAttempts le nombre maximum de chemins à explorer

```
1 info ← null ;
2 attempt ← 1 ;
3 while info = null and attempt ≤ maxNbAttempts do
4 |   info ← userAlgo(i_t, attempt) ;
5 |   attempt ← attempt + 1 ;
6 end
7 return info;
```

Fonction userAlgo

Entrée : i_t l'individu cible,
attempt le numéro de tentative
Sortie : info à stocker dans $\mathcal{O}_s$

```
1 info ← null ;
2 switch attempt do
3 |   case i            /* i=1...maxNbAttempts */          /* ordre n */
4 |   |   info ← i_t.p^1.p^2...p^n.PE_t.getVal(); break
5 endsw
6 return info;
```

Partie switch : Exemple avec des données sur des destinations

```
1 case 1 info ← i_t.PE_t.getVal(); break;                          /* accès direct */
2 case 2 info ← i_t.capital.PE_t.getVal(); break;                  /* ordre 1 */
3 case 3 info ← i_t.largestCity.PE_t.getVal(); break;              /* ordre 1 */
4 case 4 info ← i_t.subparts.PE_t.getVal(); break;                 /* ordre 1 */
5 case 5 info ← i_t.country^-1.PE_t.getVal(); break;               /* ordre 1 */
6 case 6 info ← i_t.?prop.PE_t.getVal() ∪ i_t.?prop^-.PE_t.getVal(); break;
                                                                   /* ordre 1 */
7 case 7 info ← i_t.country.capital.PE_t.getVal(); break;          /* ordre 2 */
8 case 8 info ← i_t.?prop^-1.subparts.PE_t.getVal(); break;        /* ordre 2 */
```

7.2.3 Mécanismes de traitement des valeurs de propriétés collectées

Des traitements sur les valeurs des propriétés auxquelles les mécanismes précédents ont permis d'accéder peuvent être nécessaires. Dans une première partie, nous décrivons les choix que le concepteur peut faire, puis nous présentons de façon intuitive et sur un exemple les différents mécanismes d'agrégation utiles. Dans un dernier temps, nous définissons les mécanismes d'agrégation proposés.

Choix des mécanismes de traitement

Le concepteur doit spécifier la façon de traiter les données collectées via l'application des modèles de correspondance et d'accès précédemment décrits.

S'il n'y a aucune restriction de cardinalité sur les valeurs d'une PE_s, aucun traitement spécifique n'est a priori nécessaire. Dans le cas contraire, deux cas sont possibles.

Tout d'abord, le concepteur peut souhaiter **réduire le nombre de valeurs retournées**, en ne gardant que les n premières valeurs différentes (n étant défini par le concepteur). Ceci peut être lié à la présence d'une restriction de cardinalité ou bien à la fonctionnalité de la propriété (dans ce dernier cas, $n = 1$).

Le concepteur peut aussi opter pour un **mécanisme d'agrégation**. Dans ce cas, il choisit la façon d'agréger les données collectées. Les mécanismes d'agrégation sont nécessaires pour gérer, en particulier, le cas des propriétés sources fonctionnelles. Ils sont toutefois également applicables dans d'autres situations, même si la propriété source n'est pas fonctionnelle, si le concepteur le décide. Nous ne considérons ici que le cas où l'accès est composé. En effet, lorsque l'accès est direct, l'application d'une fonction d'agrégation (cf. f dans Définition 3) à l'ensemble des valeurs de la PE_t suffit. Divers mécanismes d'agrégation ont été conçus. Nous démontrons leur intérêt avant de les définir formellement.

Intérêt des mécanismes d'agrégation

Quand l'accès est composé, le modèle défini section 7.2.2 nous conduit à prendre en compte un ensemble d'individus cibles I_t. Même si la PE_t est munie d'une fonction d'agrégation, une valeur sera obtenue pour chaque individu de I_t. Plusieurs valeurs seront donc retournées. Des mécanismes d'agrégation complémentaires sont nécessaires. Nous illustrons ces mécanismes sur des exemples dans ce qui suit.

Supposons qu'il soit possible d'accéder aux chansons d'un artiste via ses albums. Dans la figure 7.6, chaque chanson est caractérisée par sa durée. La durée est unique car la PE_t exprimant la durée intègre le calcul de la moyenne des durées en cas de valeurs multiples. Une même chanson peut être dans plusieurs albums différents (comme les chansons 3 et 4).

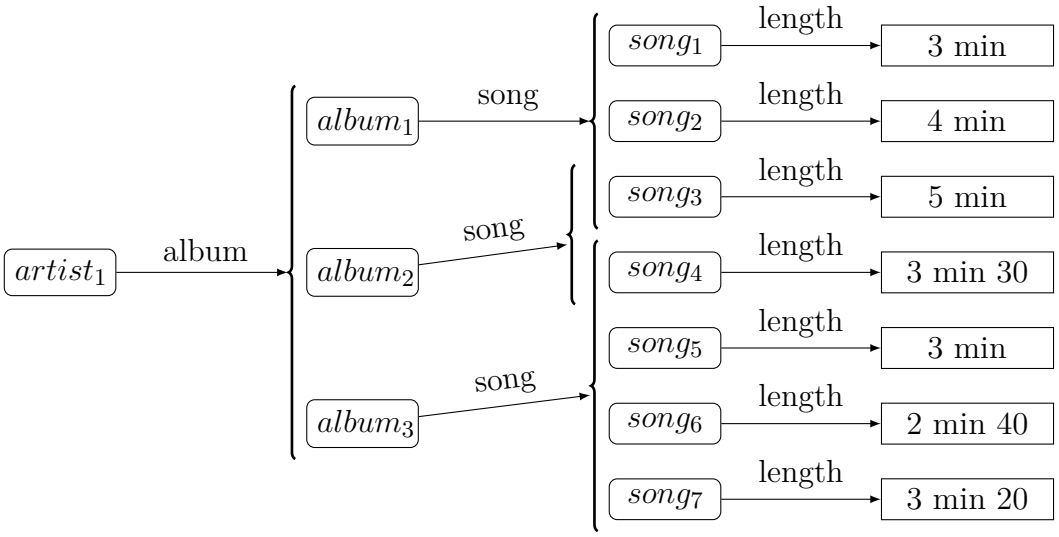


Fig. 7.6 Exemple d'accès composé pour obtenir les chansons d'un artiste

Sur cet exemple, si on recherche la durée moyenne des chansons figurant dans un album, il faut s'intéresser à toutes les chansons des albums, une chanson pouvant être considérée plusieurs fois si elle est présente sur plusieurs albums. On obtient une durée moyenne de 3min40. Ce type d'agrégation sera qualifié de simple. On agrège toutes les valeurs collectées, quel que soit l'objet caractérisé et quel que soit le chemin d'accès emprunté pour les obtenir.

En revanche, la recherche de la durée moyenne d'une chanson de l'artiste nécessite le calcul de la moyenne de la durée des 7 chansons, chaque chanson n'étant considérée qu'une seule fois. On obtient une durée de 3min30. Pour cela, l'agrégation se fait sur des valeurs associées aux chansons, indépendamment des albums. Les chansons représentent les individus cibles. Le fait de ne pas considérer les albums revient à ne pas considérer les chemins d'accès aux chansons. Ce type d'agrégation sera appelé agrégation sur les I_t (les chansons).

Enfin, si on recherche la durée moyenne d'un album, il faut faire la somme des durées des chansons pour chaque album. L'$album_1$ dure 12min, l'$album_2$ dure 8min30 et l'$album_3$ dure 12min30. La durée moyenne d'un album sera donc de 11 min. Ces deux agrégations successives, l'une sur *album* et la seconde sur *artist*, nécessitent la mise en œuvre d'un mécanisme que nous appellerons agrégation propagée.

Spécification des mécanismes d'agrégation

Nous présentons les trois mécanismes d'agrégation cités ci-dessus et les illustrons sur l'exemple du chemin donné en cas 8 dans la partie switch présentée, pour l'individu Dubai. Les chemins correspondants sont ceux présentés précédemment dans le tableau 7.1. Le tableau 7.2 indique les individus auxquels les différentes propriétés du tableau 7.1 permettent d'accéder. Notons que certaines lignes sont redondantes. Il s'agit de chemins élémentaires différents conduisant aux mêmes individus. Rappelons par ailleurs que les mécanismes décrits ici sont mis en œuvre après l'application des fonctions d'agrégation liées à la PE_t (cf. fonction f dans la définition 3). Ainsi, tout individu de I_t n'a qu'une valeur de PE_t. Dans cette section, nous adopterons les notations suivantes :

- Les expressions explicitant les mécanismes d'agrégation seront notées en respectant la syntaxe de l'algèbre relationnelle. Entre autres, nous utiliserons l'expression $G_1, G_2, \ldots, G_m\ \mathcal{F}_{f_1(A_1),f_2(A_2),\ldots,f_k(A_k)}\ (R)$, où chaque A_j est un attribut de la relation R sur lequel une fonction d'agrégation f_j s'applique. Les attributs précédant $\mathcal{F}$ sont des attributs de groupement (facultatifs). Pour plus de commodité, nous permettrons le renommage ($f_j(A_j)$ *as renommage*) dans l'opération d'agrégation.
- Pour un chemin $p^1.p^2...p^j$, on nommera I_t^j les individus atteints via ce chemin à partir de i_t. Selon cette notation, I_t^n correspond à l'ensemble des individus dont les propriétés sont considérées comme de bonnes approximations des propriétés du i_t que l'on cherche à valuer (I_t^n correspond ainsi à I_t). Dans notre exemple, $n = 2$.
- L'ensemble des éléments mis en jeu dans un accès composé (individus et valeurs de la propriété recherchée) sera noté $R(I_t^1, I_t^2, ..., I_t^n, PE_t)$, PE_t étant une propriété de I_t^n. Dans notre exemple, la relation R correspond au tableau 7.2.

Agrégation simple L'agrégation simple porte sur l'ensemble des valeurs de PE_t trouvées en considérant tous les chemins pour y accéder (cf. tableau 7.1) mais uniquement les tuples de R différents (cf. tableau 7.3). Rappelons que l'algèbre relationnelle ne considère pas les doublons. En conséquence, la relation R considérée dans la formule ci-dessous intègre déjà cette contrainte. Dans notre exemple, si on prend la moyenne

Régions liées à Dubai (I_t^1)	Villes (I_t^2)	Moy. températures août (PE_t)
Emirate_of_Sharjah	Sharjah	34.7
Emirate_of_Abu_Dhabi	Abu_Dhabi	36.2
United_Arab_Emirates	Abu_Dhabi	36.2
Emirate_of_Abu_Dhabi	Abu_Dhabi	36.2
United_Arab_Emirates	Abu_Dhabi	36.2

Tableau 7.2 Chemin du cas 8 appliqué sur Dubai pour une PE_t calculant la moyenne de la température en Août

Régions liées à Dubai (I_t^1)	Villes (I_t^2)	PE_t
Emirate_of_Sharjah	Sharjah	34.7
Emirate_of_Abu_Dhabi	Abu_Dhabi	36.2
United_Arab_Emirates	Abu_Dhabi	36.2

Tableau 7.3 Tableau 7.2 sans doublons

comme opération d'agrégation, la valuation finale considérée sera la moyenne des 3 valeurs, soit 35.7.

$$\mathcal{F}_{AGR(PE_t)\ as\ res}\ (R)$$

Agrégation sur I_t L'agrégation sur I_t porte sur les valeurs associées aux individus de l'ensemble I_t^n (sans doublons), cf. tableau 7.4. Les chemins qui ont permis d'accéder à ces individus sont ignorés. Dans notre exemple, l'agrégation porte sur (I_t^2, PE_t). Si on prend la moyenne comme opération d'agrégation, la valuation finale considérée sera la moyenne des 2 valeurs, i.e. 35.45.

$$\mathcal{F}_{AGR(PE_t)\ as\ res}\ (\Pi_{I_t^n,PE_t}(R))$$

Villes (I_t^2)	PE_t
Sharjah	34.7
Abu_Dhabi	36.2

Tableau 7.4 Représentation de $\Pi_{I_t^n,PE_t}(R)$ sur notre exemple

Agrégation propagée L'agrégation propagée consiste à appliquer une suite d'opérations d'agrégation (éventuellement différentes) en effectuant différents groupements, d'abord par $I_t^1, I_t^2 ..., I_t^{n-1}$ puis par $I_t^1, I_t^2, ..., I_t^{n-2}$, et ainsi de suite jusqu'à I_t^1. Ensuite, on agrège l'ensemble de valeurs obtenues. Ce mécanisme donne de l'importance aux différents individus composant les chemins. Dans l'exemple, son application consiste d'abord à faire un regroupement par région (I_t^1) comme le montre le tableau 7.5.

On agrège ainsi les températures des différentes sous-parties géographiques (ici des villes) de chaque région. Dans un second temps, on agrège les températures obtenues pour les différentes régions en faisant la moyenne des trois valeurs trouvées, soit 35.7. Remarquons qu'on obtient ici la même valeur qu'avec une agrégation simple car, dans cet exemple, il n'y a qu'un seul I_t^2 associé à chaque I_t^1.

$$\mathcal{F}_{AGR_1(val_1)\ as\ res}\ (I_t^1 \mathcal{F}_{AGR_2(val_2)\ as\ val_1}\ (I_t^1, I_t^2\ \mathcal{F}_{AGR_3(val_3)\ as\ val_2}\ (...($$

$$I_t^1, I_t^2, ..., I_t^{n-1}\ \mathcal{F}_{AGR_n(PE_t)\ as\ val_{n-1}}\ (R)$$

$$)\qquad\qquad)\qquad\qquad)...)$$

Régions liées à Dubai ($I_t^1 = I_t^{n-1}$)	PE_t ($val_1 = val_{n-1}$)
Emirate_of_Sharjah	34.7
Emirate_of_Abu_Dhabi	36.2
United_Arab_Emirate	36.2

Tableau 7.5 Représentation de $I_t^1, I_t^2, ..., I_t^{n-1}\ \mathcal{F}_{AGR_n(val)\ as\ val_{n-1}}\ (R)$ sur notre exemple

7.3 Conclusion

Le modèle d'acquisition présenté ici permet de spécifier les correspondances entre une ontologie source et une ontologie cible. Il est exploité dans le cadre de l'approche Saupodoc pour spécifier les correspondances entre l'ontologie utilisée et un jeu de données du LOD. Pour que cette approche soit performante, nous devons disposer de données les plus complètes possibles, quitte à ce que celles-ci ne soient que des approximations de valeurs. Cela est possible grâce au modèle d'accès explorant les chemins alternatifs. De plus, des contraintes de l'ontologie (telles que les restrictions de cardinalité de certaines propriétés ou encore le type du co-domaine) doivent être respectées si l'on souhaite effectuer un quelconque raisonnement sur celle-ci. Cette nécessité est remplie par les mécanismes de traitement proposés. Le concepteur de l'application exprime les correspondances (correspondances et chemins d'accès) et cette spécification est par la suite utilisée pour générer automatiquement des requêtes SPARQL permettant de collecter les données du LOD et de les insérer dans l'ontologie de Saupodoc. Cette génération automatique fait l'objet du Chapitre 8.

Chapitre 8

Génération automatique de requêtes à partir du modèle d'acquisition

Sommaire

Le modèle d'acquisition décrit dans le Chapitre 7 est utilisé comme support à la fois, pour la collecte de données et pour la génération automatique de requêtes. Nous considérons en effet que la structure des requêtes SPARQL peut être déterminée par le modèle de correspondance et d'accès proposé, et ceci de façon indépendante du domaine d'application.

Ce chapitre est structuré de la façon suivante. Dans un premier temps, nous expliquons comment le modèle d'acquisition sert de support à la génération automatique de requêtes SPARQL. Puis, nous déroulons des exemples illustrant l'intérêt du modèle proposé et le processus de génération des requêtes. Enfin, nous concluons et énonçons quelques perspectives.

8.1 Génération automatique des requêtes SPARQL basée sur le modèle d'acquisition

```
CONSTRUCT { i_s  p_s  ?val_0 }
WHERE{
    i_t  traduction(traitement(chemin_n.PE_t))  ?val_0
}
```

Le modèle décrit dans la section précédente sert de support pour générer automatiquement des requêtes SPARQL de type `CONSTRUCT`. Ces requêtes retournent un graphe RDF (assertions de propriétés) qui sera ajouté dans $\mathcal{O}_s$. On notera p_s la propriété de l'individu i_s considérée dans PE_s dont on cherche la valeur ($?val_0$). Le triplet à ajouter dans $\mathcal{O}_s$ sera donc de la forme $\{i_s\ p_s\ ?val_0\}$. Une clause `WHERE` dans la requête de type `CONSTRUCT` indiquera ensuite comment obtenir la (ou les) valeur(s) de $?val_0$. Cette clause contiendra un motif de graphe à apparier sur le jeu de données cible, de la forme $\{i_t\ \ traduction(traitement(chemin_n.PE_t))\ \ ?val_0\}$ et tel que :

- $(i_s, i_t) \in Ind_{s,t}$ et $i_s \in domaine(PE_s)$,
- PE_t est l'expression de propriété cible mise en correspondance avec PE_s via le modèle de correspondance défini dans la section 7.2.1,
- $chemin_n$ est un chemin de propriétés d'ordre n, éventuellement vide, définissant un éventuel accès composé à PE_t,
- `traitement` correspondant aux traitements éventuellement appliqués in fine (cf. section 7.2.3),
- et enfin `traduction` traduit, si besoin, les valeurs obtenues pour que celles-ci soient compatibles avec O_s (cf. section 8.1.2).

L'exécution des requêtes suit l'algorithme 1 donné dans la section précédente. Par exemple, si la partie switch contient 8 cas comme dans l'exemple de partie switch donné dans le Chapitre 7, alors 8 requêtes de type `CONSTRUCT` sont créées (une par cas) contenant les variables i_s et i_t. Ensuite, pour chaque couple (i_s, i_t) considéré, nous exécutons la requête du cas 1. Si son résultat n'est pas vide, nous continuons avec le couple (i_s, i_t) suivant. Sinon, nous exécutons la requête du cas 2 et ainsi de suite jusqu'à ce que le résultat ne soit pas vide pour le couple courant.

Nous décrivons dans ce qui suit comment générer le contenu du bloc `WHERE` de cette requête pour une PE_s donnée, en adoptant une approche à base de patrons. Tous les patrons proposés génèrent des parties d'expressions SPARQL devant être introduites dans le bloc `WHERE` de la requête. Dans une première partie, nous présentons le processus général de génération automatique du bloc `WHERE` de la requête de type `CONSTRUCT`, puis, dans une seconde partie, les patrons construits.

8.1.1 Processus de génération de requêtes SPARQL 1.1

Nous décrivons ci-dessous le processus de génération du bloc `WHERE` de la requête `CONSTRUCT` qui ajoutera les assertions de propriétés souhaitées dans $\mathcal{O}_s$. Le bloc `WHERE` est généré à partir du modèle d'acquisition présenté en section 7.2 et à partir de patrons.

Nous proposons un ensemble de patrons classés en quatre catégories, ces dernières portant sur des éléments pouvant être considérés indépendamment les uns des autres. En effet, des expressions différentes en SPARQL 1.1 permettent de tenir compte du format de la PE_t considérée, de l'existence d'un chemin pour accéder aux valeurs de cette propriété, des traitements finaux à exécuter ou de la traduction des valeurs acquises. Nous proposons donc 4 catégories de patrons, une pour chaque type de problème : Format (de la PE_t), Chemin, Traitement, Traduction. Un patron correspond à un sous-problème d'une catégorie.

La construction du bloc `WHERE` de la requête à générer s'effectue en deux phases qui peuvent être réalisées en parallèle, chaque phase générant une partie de la requête. Une phase dite externe génère la partie de requête, que nous appellerons *partie externe*, liée à la traduction et aux traitements des valeurs. Une autre phase, dite interne, génère la partie de requête, que nous appellerons *partie interne*, liée au format de la PE_t et au chemin d'accès. La partie interne est ensuite imbriquée dans la partie externe afin d'obtenir la requête SPARQL finale dont l'exécution délivrera les valeurs de propriétés recherchées.
La phase externe a deux étapes. La première étape génère la partie SPARQL portant sur la traduction (par application des patrons de la catégorie Traduction). La deuxième étape y insère la partie portant sur le Traitement (par application des patrons de la catégorie Traitement).
La phase interne a 3 étapes. La première étape génère la partie SPARQL portant sur le format de la PE_t (par application des patrons de la catégorie Format). La deuxième étape effectue un pré-traitement avant l'insertion d'un éventuel chemin (recherche de l'emplacement dans la requête pour sa déclaration et changement de noms de variables). L'étape 3 insère le chemin (par application des patrons de la catégorie Chemin).
L'ordre d'application des patrons au sein d'une catégorie sera décrit dans la section 8.1.2.

Au départ, le bloc `WHERE` contient uniquement l'expression "i_t $traduction(traitement(chemin_n.PE_t))$ $?val_0$". Dans les patrons de traduction, cette expression devient "i_t $traitement(chemin_n.PE_t))$ $?val$". Puisque la phase de traduction a été prise en compte, le terme *traduction* n'est plus mentionné. De la même façon, dans les patrons dits de traitement, elle devient "i_t $chemin_n.PE_t$ $?val$", le terme *traitement* n'ayant plus besoin d'être mentionné. L'expression, "i_t $chemin_n.PE_t$ $?val$", est traitée par la phase interne. Les patrons dits de Format liés à PE_t permettent de trouver le motif de graphe correspondant à "i_t PE_t $?val$" pour la requête en cours de construction. Les patrons liés aux chemins interviennent ensuite pour intégrer la prise en compte éventuelle d'un chemin afin d'obtenir un motif de graphe de la forme "i_t $chemin_n.PE_t$ $?val$".

Notons que lors de l'application de chaque patron, les noms de variables sont modifiés si nécessaire. La variable dont le contenu est renvoyé par la requête reste toutefois $?val_0$.

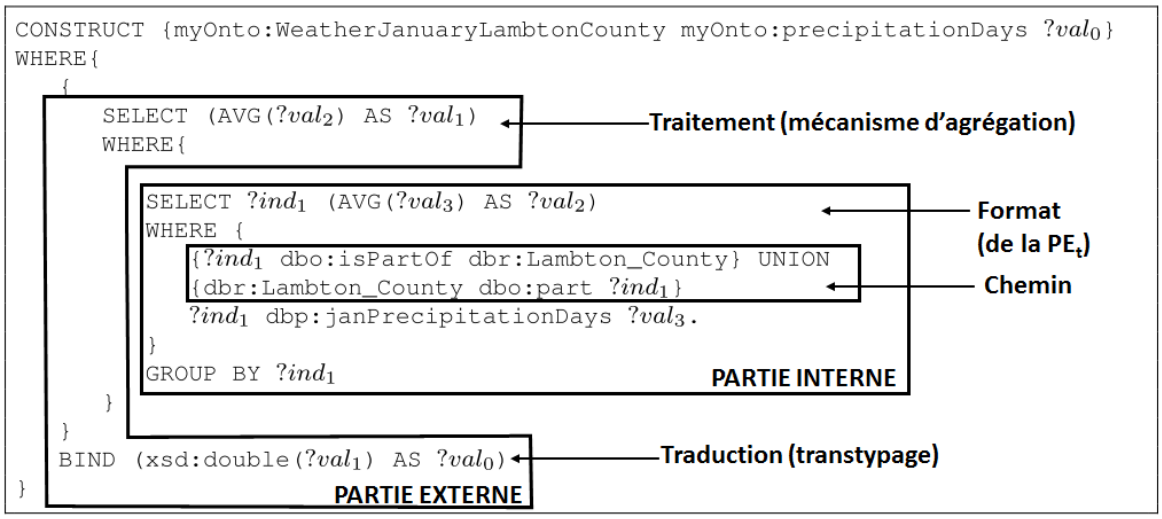


Fig. 8.1 Les parties interne et externe d'une requête

La figure 8.1 reprend l'exemple 4 de la page 97 en indiquant les deux parties, interne et externe, de la requête, la partie interne étant imbriquée dans la partie externe. Les parties de la requête générées par les différentes catégories de patrons sont également précisées.

8.1.2 Présentation des différents patrons

Nous présentons ici les patrons construits par catégorie. Dans ces patrons, la partie **Cond** représente la condition d'application du patron, i.e. le contexte, et la partie **Action**, la solution mise en œuvre.

Patrons liés au format d'une PE_t

7 patrons permettent de tenir compte de l'ensemble des formats d'une expression de propriété PE_t (cf. Définition 3). Ces patrons s'appliquent lorsqu'on cherche à générer la partie de requête correspondant à "i_t PE_t $?val_j$" pour tenir compte de la spécificité de l'expression de propriété dont on cherche les valeurs. Le processus d'application des patrons de cette catégorie est guidé par la définition de la PE_t. Ainsi, si une PE_t s'exprime en fonction d'une ou plusieurs autres PE_t, le patron associé est exprimé en fonction d'une ou plusieurs expressions "i_t PE_t $?val$". L'imbrication des patrons s'effectuera de la même manière que l'imbrication de la PE_t, en commençant par la partie la plus externe. Par exemple si $PE_t = f(PE_t)$, on appliquera d'abord le patron associé à la fonction f. Puis, l'expression "i_t PE_t $?val$" qu'il contient sera récursivement remplacée par application du patron correspondant à la PE_t sur laquelle porte f. Dans le cas d'une PE_t contrainte, on appliquera d'abord le patron de contrainte.

Patron$_{elem}$
Cond : $PE_t = p_e$
Action : Remplacer p_e par le nom de la propriété élémentaire.
À noter que si p_e vaut op^{-1}, alors "$i_t\ \ p_e\ \ ?val_j$" sera remplacé par "$?val_j\ \ op\ \ i_t$".

```
i_t  p_e  ?val_j .
```

Patron$_{transfo_1}$
Cond : $PE_t = f(PE_t)$ avec f une fonction de transformation.
Action : Remplacer f par l'expression mathématique souhaitée ou la fonction SPARQL 1.1 correspondant à la fonction de transformation à appliquer (STRLEN, LCASE, ABS, etc).

```
{i_t  PE_t1  ?val_j+1}
BIND (f(?val_j+1) AS ?val_j)
```

Patron$_{agr}$
Cond : $PE_t = f(PE_t)$ avec f une fonction d'agrégation.
Action : Remplacer f par l'opérateur SPARQL 1.1 correspondant à la fonction d'agrégation à appliquer (COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT et SAMPLE).

```
SELECT (f(?val_j+1) AS ?val_j)
WHERE {
    i_t  PE_t  ?val_j+1
}
```

Patron$_{ens}$
Cond : $PE_t = f(PE_t, PE_t)$ avec f une opération ensembliste.
Action : Remplacer f par l'opérateur SPARQL 1.1 (UNION ou MINUS selon le cas).

```
{i_t  PE_t1  ?val_j}
f
{i_t  PE_t2  ?val_j}
```

Patron$_{transfo_2}$
Cond : $PE_t = f(PE_t, PE_t)$ avec f une opération de transformation.
Action : Remplacer f par l'expression mathématique ou la fonction SPARQL 1.1 souhaitée (CONCAT, CONTAINS, etc).

```
{i_t  PE_t1  ?val_j+1_a}
{i_t  PE_t2  ?val_j+1_b}
BIND (f(?val_j+1_a,?val_j+1_b) AS ?val_j)
```

Patron$_{constr}$
Cond : $PE_t = PE_t.Constr(d) \mid PE_t.Constr(r)$.
La contrainte $Constr$ peut porter sur le domaine de la PE_t, c'est-à-dire i_t, ou bien sur le co-domaine r, c'est-à-dire $?val_j$.
Action : Remplacer $Constr$ par son expression en motif de graphe. Si c'est une contrainte de co-domaine, remplacer r par le nom de la variable associée, c'est-à-dire $?val_j$.

```
{i_t  PE_t  ?val_j}
Constr
```

Patron$_{having}$
Cond : PE_t=$PE_t.Constr(f(Constr(r)))$
Action : Instancier l'expression du `HAVING` en fonction de la con-

```
SELECT ?val_j
WHERE {
    i_t  PE_t  ?val_j
}
GROUP BY ?val_j
HAVING (f(?contrainte) operateur valeur)
```

trainte formulée. Entre autres, remplacer f par l'opérateur SPARQL 1.1 correspondant (COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, SAMPLE). Si la variable de contrainte `?contrainte`, correspondant à Constr(r), est dans une requête imbriquée, on ajoute l'appel à cette variable dans chaque `SELECT` imbriquant la variable ainsi qu'un groupement sur cette variable.

Patrons liés aux chemins d'accès

On distingue trois patrons liés aux chemins d'accès selon qu'il s'agit de déclarer un chemin élémentaire, un ensemble de chemins élémentaires ou des contraintes associées. Ces patrons s'appliquent lorsqu'on cherche à générer la partie de requête correspondant à "i_t $chemin_n$ $?ind_n$".

L'application de ces patrons fait suite à une étape de pré-traitement qui consiste à sélectionner l'emplacement où cette expression du chemin doit être déclarée et à effectuer les changements de noms de variables appropriés. En effet, dans la requête exprimant le format de la PE_t, on pourrait remplacer toute référence à i_t par le chemin considéré. Cependant, ce chemin serait répété de nombreuses fois dans la requête si la PE_t fait appel à de nombreuses propriétés élémentaires. Grâce à l'étape de pré-traitement, le chemin ne sera cité qu'une seule fois.

L'emplacement du chemin dépend de la forme de la partie interne de la requête en cours de construction.

- Si celle-ci débute par une clause `SELECT`, on insère l'expression "i_t $chemin_n$ $?ind_n$" au sein de la clause `WHERE` liée à la clause `SELECT`. Ce sera le cas lorsque le format de la PE_t est $f(PE_t)$ ou $PE_t.Constr(f(Constr(r))$, avec f une fonction d'agrégation, et que la PE_t est éventuellement munie de contraintes de domaine et/ou de co-domaine. De plus, on ajoute dans cette clause `SELECT` les variables correspondant aux individus mis en jeu dans le chemin, c'est-à-dire [$?ind_1$, $?ind_2$, ..., $?ind_{n-1}$,] $?ind_n$ et un groupement sur ces variables. Les variables mises entre crochets sont à ignorer s'il n'y a pas de mécanisme de traitement, ou s'il s'agit d'une restriction de nombre de valeurs, ou d'une agrégation sur les I_t.
- Si la partie interne de la requête en cours de construction ne débute pas par une clause `SELECT`, on insère "i_t $chemin_n$ $?ind_n$" au début de celle-ci.

Dans les deux cas, s'il existe des clauses `SELECT` imbriquées, on ajoutera $?ind_n$ dans ces clauses et on groupera sur cette variable.

Le second pré-traitement consiste à remplacer les i_t de la partie interne en cours de construction par la variable $?ind_n$ correspondant aux I_t^n.

Les patrons de Chemin s'imbriquent en suivant la définition de $chemin_n$. Pour instancier les patrons, on partira du patron de contrainte $Patron_{chem_{constr}}$ qui

s'exprime en fonction de $chemin_n$ et qui pourra imbriquer tout type de patron de Chemin. Le patron d'ensemble $Patron_{chem_{ens}}$ qui s'exprime en fonction de chemin élémentaire $p^1.p^2...p^n$ pourra imbriquer des patrons de chemins élémentaires $Patron_{chem_{elem}}$.

Patron$_{chem_{elem}}$
Cond : $chemin_n = p^1.p^2...p^n$

i_t p^1 $?ind_1$. $?ind_1$ p^2 $?ind_2$ . ... $?ind_{n-1}$ p^n $?ind_n$.

Action : Remplacer les propriétés p^j par leur nom. À noter que si p^j vaut op^{-1}, alors "x p^j $?ind_j$" sera remplacé par "$?ind_j$ op x".

Patron$_{chem_{ens}}$
Cond : $chemin_n = Ens(p_1^1.p_1^2...p_1^n,\ p_2^1.p_2^2...p_2^n,\ ...,\ p_m^1.p_m^2...p_m^n)$

{i_t $p_1^1.p_1^2...p_1^n$ $?ind_n$} UNION {i_t $p_2^1.p_2^2...p_2^n$ $?ind_n$} UNION...UNION {i_t $p_m^1.p_m^2...p_m^n$ $?ind_n$}

Action : Adapter le nombre d'opérateurs `UNION` au nombre de chemins élémentaires de l'ensemble.

Patron$_{chem_{constr}}$
Cond : $chemin_n = chemin_n.Constr(i_t) \mid chemin_n.Constr(I_t)$

i_t $chemin_n$ $?ind_n$ $Constr$

Action : Remplacer $Constr$ par son expression. S'il s'agit d'une contrainte sur I_t, on remplacera I_t par le nom de la variable qui lui est associée ($?ind_n$). Suivant la forme du chemin, "i_t $chemin_n$ $?ind_n$" sera remplacé par l'application du patron de Chemin correspondant.

Exemple 12. Soit $PE_t = MOYENNE(\text{janRainMm}.FILTER(r >= 0))$, la moyenne des valeurs positives de janRainMm. Dans la première étape de la phase interne, les patrons $Patron_{agr}$, $Patron_{Const}$ et $Patron_{elem}$ sont appliqués (dans cet ordre). On obtient :

```
1 SELECT (AVG(?val1) AS ?val0)
2 WHERE {
3
4     it dbp:janRainMm ?val1.
5     FILTER(?val1 >=0)
6 }
```

Dans un second temps, on positionne le chemin (d'ordre 2 dans cet exemple) au sein de la clause `WHERE` (ligne 3). De plus, on ajoute les variables représentant les individus mis en jeu dans le chemin (lignes 1 et 7). Enfin, on fait le changement de variable nécessaire (ligne 4).

```
1 SELECT [?ind1] ?ind2 (AVG(?val1) AS ?val0)
2 WHERE {
3     it chemin2 ?ind2.
```

```
4     ?ind2 dbp:janRainMm ?val1.
5     FILTER(?val1 >=0)
6  }
7  GROUP BY [?ind1] ?ind2
```

Dans un troisième temps, on applique les patrons liés aux chemins. Dans le cas présent, supposons que le chemin à considérer soit country.capital (exemple du cas 7 du Switch donné page 106). Le patron $Patron_{chem_{elem}}$ s'applique, instanciant le chemin (ligne 3 modifiée).

```
1  SELECT [?ind1] ?ind2 (AVG(?val1) AS ?val0)
2  WHERE {
3     it dbo:country ?ind1. ?ind1 dbo:capital ?ind2.
4     ?ind2 dbp:janRainMm ?val1.
5     FILTER(?val1 >=0)
6  }
7  GROUP BY [?ind1] ?ind2
```

Exemple 13. Reprenons maintenant l'exemple 11 donné page 104 où $PE_t = UNION(country, nationality)$. Nous appliquons tout d'abord les patrons $Patron_{ens}$ et $Patron_{elem}$ de la catégorie Format, ce qui permet d'obtenir :

```
1
2
3
4  { it dbp:country ?val0 } UNION {it dbp:nationality ?val0 }
```

L'étape de pré-traitement positionne le chemin (d'ordre 1) au début (ligne 1) et fait le changement de variable nécessaire (ligne 4 modifiée).

```
1  it chemin1 ?ind1.
2
3
4  { ?ind1 dbp:country ?val0 } UNION { ?ind1 dbp:nationality ?val0 }
```

Dans la troisième étape, nous considérons ensuite le chemin $director.Constr(i_t$ `dbp:released ?year. ?ind1 rdfs:label ?name. FILTER(?year<=2000 && !regex(?name, "Ridley Scott") && !regex(?name, "James Cameron"))`). Ce chemin sera construit via les patrons $Patron_{chem_{constr}}$ puis $Patron_{chem_{elem}}$ (chemin élémentaire en ligne 1 et contrainte lignes 2-3). Notons qu'ici, par simplification, nous avons considéré les deux contraintes sur i_t et I_t en une seule et même contrainte.

```
1  it dbo:director ?ind1.
2  it dbp:released ?year. ?ind1 rdfs:label ?name. FILTER(?year<=2000 &&
3  !regex(?name, "Ridley Scott") && !regex(?name, "James Cameron"))
4  { ?ind1 dbp:country ?val0 } UNION { ?ind1 dbp:nationality ?val0 }
```

Patrons liés aux traitements

On distingue 3 patrons liés au traitement des valeurs finales collectées, un portant sur la réduction du nombre de valeurs retournées et les deux suivants concernant les mécanismes d'agrégation décrits section 7.2.3. Les patrons liés aux traitements d'agrégation contiennent la contrainte `FinalConstr(?val)`. Cette expression exprime

une éventuelle contrainte donnée par le concepteur sur la valeur finale après application du mécanisme d'agrégation.

Patron$_{trait_{reduc}}$
Cond : `Traitement` =
Réduction du nombre de valeurs.

```
SELECT DISTINCT ?val_j
WHERE {
    i_t chemin_n.PE_t ?val_j
}
LIMIT n
```

Patron$_{trait_{agr}}$
Cond : `Traitement` =
1) Agrégation simple si "i_t $chemin_n.PE_t$ $?val_{j+1}$" commence par une clause `SELECT` contenant toutes les variables représentant les I_t^j $(1 \leq j \leq n)$.
2) Agrégation sur les I_t si "i_t $chemin_n.PE_t$ $?val_{j+1}$" commence par une clause `SELECT` ne contenant que la variable représentant les I_t^n.

```
{
    SELECT (AGR(?val_{j+1}) AS ?val_j)
    WHERE{
        i_t chemin_n.PE_t ?val_{j+1}.
    }
}
FinalConstr(?val_j)
```

Action : Remplacer `AGR` par l'agrégat à appliquer avec les opérateurs SPARQL 1.1. Remplacer `FinalConstr(`$?val_j$`)` par la contrainte portant sur la valeur finale agrégée obtenue.

Patron$_{trait_{agrProp}}$
Cond : `Traitement` = Agrégation propagée.

```
{
    SELECT (AGR_1(?val_{j+1}) AS ?val_j)
    WHERE{
        SELECT ?ind_1 (AGR_2(?val_{j+2}) AS ?val_{j+1})
        WHERE{
            SELECT ?ind_1 ?ind_2 (AGR_3(?val_{j+3}) AS ?val_{j+2})
                WHERE{
                ...
                    SELECT ?ind_1 ?ind_2 ... ?ind_{n-1} (AGR_n(?val_{j+n}) AS ?val_{j+n-1})
                    WHERE{
                        i_t chemin_n.PE_t ?val_{j+n}.
                    }
                    GROUP BY ?ind_1 ?ind_2 ... ?ind_{n-1}
                ...
                }
                GROUP BY ?ind_1 ?ind_2
        }
        GROUP BY ?ind_1
    }
}
FinalConstr(?val_j)
```

Action : Remplacer tous les AGR_k par les agrégats à appliquer avec les opérateurs SPARQL 1.1. Remplacer `FinalConstr(`$?val_j$`)` par la contrainte portant sur la valeur finale agrégée obtenue.

Patrons liés à la traduction des valeurs

Cette dernière catégorie de patrons permet d'effectuer les traductions de valeurs. En effet, si p_s est une propriété datatype dont on connaît le co-domaine, un transtypage peut être nécessaire pour que les valeurs collectées soient du type voulu. Si p_s est une propriété objet, les données acquises doivent être transformées en individus de O_s. Dans les deux cas, cela se fait automatiquement.

Patron$_{transty}$
Cond : p_s est une propriété datatype dont le co-domaine correspond à un type bien précis.

```
{i_t Traitement(chemin_n.PE_t) ?val_{j+1}}
BIND (transtypage(?val_{j+1}) AS ?val_j)
```

Action : Remplacer `transtypage` par le type du co-domaine considéré.

Patron$_{indiv}$
Cond : p_s est une propriété objet.

```
{i_t Traitement(chemin_n.PE_t) ?val_{j+1}}
BIND (xsd:String(?val_{j+1}) AS ?stringVal)
BIND (replace(?stringVal, prefix(O_t), "") AS ?string)
FILTER (!(?string=""))
BIND (iri(concat(prefix(O_s), encode_for_uri(?string))) AS ?val_j)
```

Action : Remplacer $\mathcal{O}_s$ et $\mathcal{O}_t$ par le préfixe de l'ontologie considérée.
Le premier `BIND` transforme en chaîne de caractères les valeurs obtenues qui peuvent être initialement de types variés (string, IRI, etc). Le second est utile en cas d'IRI. Il permet de ne garder que le fragment dénominatif de l'entité considérée. Enfin le dernier `BIND` permet de créer une IRI pour O_s reprenant la partie dénominative de l'entité.

Notons que $Patron_{indiv}$ est une version simplifiée du patron réellement utilisé dans l'approche SAUPODOC. En effet, dans le cadre de SAUPODOC, ce dernier patron est un peu plus compliqué car nous avons souhaité insérer les labels du nouvel individu considéré. On type aussi le nouvel individu créé en fonction du co-domaine de la propriété source. Ainsi, nous considérons le patron $Patron_{indiv_{SAUPODOC}}$.

Patron$_{indiv_{SAUPODOC}}$
Cond : p_s est une propriété objet.

```
SELECT ?val_j (IF(isIRI(?val_{j+1}), ?lab, ?val_{j+1}) AS ?label)
WHERE {
   {i_t Traitement(chemin_n.PE_t) ?val_{j+1}}
   BIND (xsd:String(?val_{j+1}) AS ?stringVal)
   BIND (replace(?stringVal, (O_t, "") AS ?string)
   FILTER (!(?string=""))
   BIND (iri(concat((O_s, encode_for_uri(?string))) AS ?val_j)
   OPTIONAL {
      FILTER isIRI(?val_{j+1}).
      ?val_{j+1} rdfs:label ?lab.
```

```
    }
}
```

Action : Remplacer $\mathcal{O}_s$ et $\mathcal{O}_t$ par le préfixe de l'ontologie considérée.
Les 3 premiers BIND n'ont pas changé. Le `OPTIONAL` s'applique quand la valeur est une IRI. Dans ce cas, on sauvegarde ses labels dans ?lab. Ainsi, la requête retourne 2 variables $?val_j$ (l'individu source à créer) et ?label (les labels de l'IRI cible). Lorsqu'une valeur trouvée dans l'ontologie source est une adresse IRI, $?val_j$ est basée sur le même fragment que cette IRI et ?label est l'ensemble des labels associés à l'IRI. Lorsqu'une valeur trouvée dans l'ontologie cible n'est pas une adresse IRI (c'est alors en général une chaîne de caractère), $?val_j$ est basée sur cette valeur et ?label est la valeur trouvée.

Dans le cas où $Patron_{indiv_{SAUPODOC}}$ s'applique, la requête CONSTRUCT est un peu plus compliquée. En effet, celle-ci correspond au squelette suivant :

```
CONSTRUCT {
    i_t  p_s  ?val_0 .
    ?val_0  rdfs:label  ?label .
    ?val_0  rdf:type  O_s:range(p_s) .
}
WHERE{
    i_t  traduction(traitement(chemin_n.PE_t))  ?val_0
}
```

où le bloc WHERE fait appel à $Patron_{indiv_{SAUPODOC}}$. Ainsi, l'ensemble des nouveaux individus $?val_0$ est créé avec des labels. S'il s'agissait d'une IRI dans l'ontologie source, alors les labels sont les labels de l'IRI, sinon le label est la valeur trouvée. On type le nouvel individu en fonction du co-domaine de la propriété p_s.

Par exemple, la Figure 8.2 montre un extrait de la page DBpedia décrivant le film "008: Operation Exterminate". Cette page contient entre autres, le pays d'origine du film, sa durée et sa langue. La requête générée permet d'insérer ces trois informations dans l'ontologie cible, cf. Figure 8.3. Pour instancier les propriétés objets "isFromCountry" et "hasLanguage", deux individus ont été créés : _Italy cf. Figure 8.4 et _Italian cf. Figure 8.5. _Italy vient de l'IRI dbr:Italy donc cet individu a les mêmes labels que dbr:Italy. _Italian provient de la chaîne de caractère "Italian", c'est donc son label. De plus, comme les propriétés "isFromCountry" et "hasLanguage" ont respectivement pour co-domaine Country et Language, les individus _Italy et _Italian sont respectivement typés par ce co-domaine.

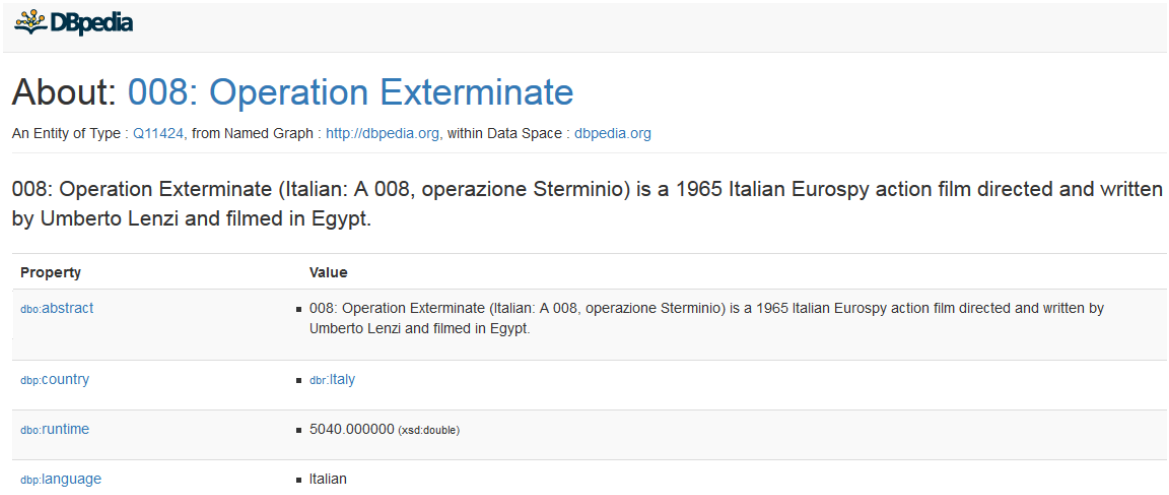


Fig. 8.2 Extrait d'une page DBpedia d'un film

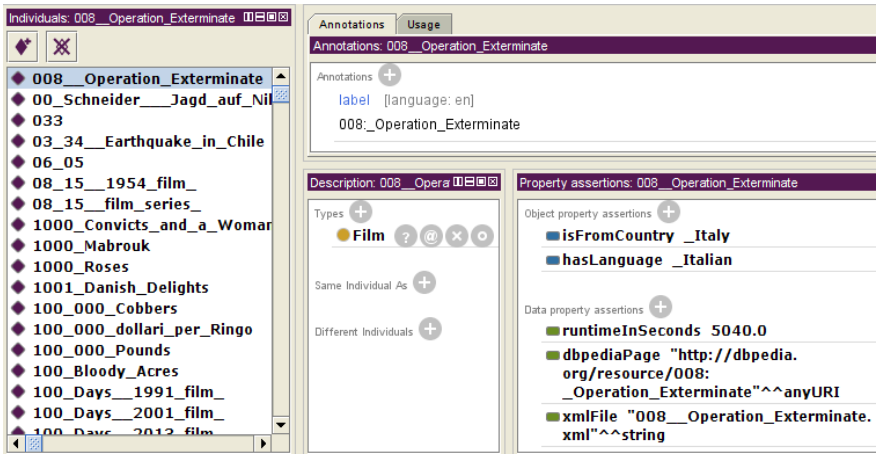


Fig. 8.3 Exemple d'assertions créées dans l'ontologie cible

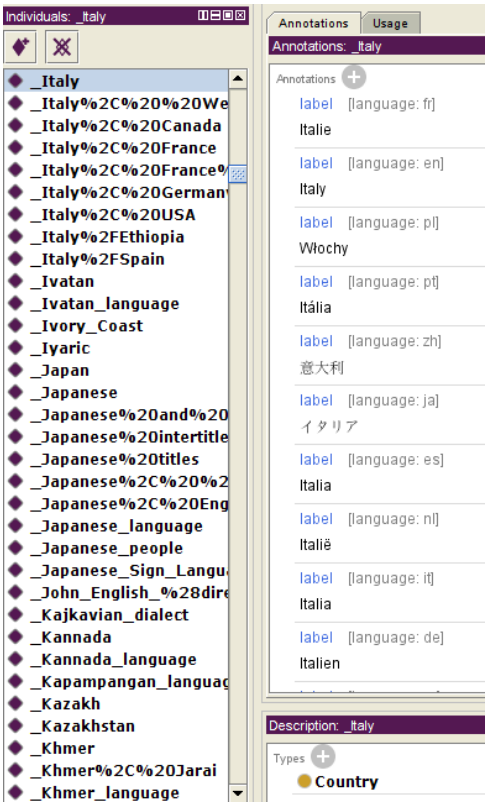


Fig. 8.4 Individu créé pour l'IRI dbr:Italy

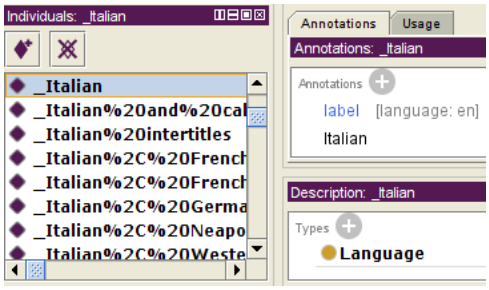


Fig. 8.5 Individu créé pour la chaîne de caractère "Italian"

8.2 Déroulement de la génération de requêtes

Dans cette section, nous déroulons la génération des requêtes sur les deux exemples donnés en section 7.1 telle qu'elle est exécutée par le système implémenté pour supporter ce processus. Ces exemples illustrent l'apport de notre travail par rapport à l'état de l'art. En effet, à notre connaissance, aucun travail ne porte sur le traitement d'expressions complexes telles que nous les avons définies. [Correndo *et al.* 2010, Makris *et al.* 2012] ne considèrent pas les agrégats introduits dans la version 1.1 de SPARQL. [Makris *et al.* 2012] estiment que le travail d'agrégation peut être vu comme un post-traitement d'une requête. Cela est vrai pour des cas simples, mais s'il faut appliquer plusieurs agrégats, chacun portant sur des résultats de sous-requêtes, et si les résultats de ces agrégations font ensuite l'objet de transformations, ce n'est plus vrai. Le premier exemple en est une illustration.

Soit $PE_t = Max(Max(\mathsf{populationDensity}), Max(\mathsf{populationTotal})/Min(\mathsf{areaTotal}))$, correspondant à la densité maximale de la population au km^2 du Canada, citée dans l'exemple 3 donné page 96. L'accès aux valeurs de l'ensemble des éléments composant cette PE_t dans l'ontologie cible est direct. En revanche, une seule valeur doit être retournée, la densité maximale.

La valeur recherchée ne peut pas être obtenue par un post-traitement d'une unique requête SPARQL retournant l'ensemble des valeurs nécessaires au calcul. Il faudrait 3 requêtes indépendantes retournant toutes les valeurs des 3 propriétés mises en jeu, suivies d'un certain nombre d'étapes manuelles pour réaliser le post-traitement :

- agréger chacune des valeurs des 3 propriétés,
- calculer le résultat de la division demandée,
- prendre le max entre les valeurs.

La deuxième étape est de plus basée sur le résultat de la première et la 3ème étape sur le résultat de la seconde. Un tel post-traitement est complexe à mettre en œuvre car il est pratiquement entièrement à la charge du concepteur. Générer une requête prenant automatiquement en compte les agrégats décharge totalement le concepteur de l'écriture de requêtes SPARQL et semble beaucoup plus approprié. C'est l'objet de notre approche illustrée sur cet exemple par la figure 8.6.

Dans la figure 8.6, nous montrons l'évolution de la construction du bloc `WHERE` de la requête finale traitant l'expression "`traduction(`i_t PE_t $?val_0$`)`" (sans mécanisme de traitement, ni chemin). PE_s, associée à PE_t, a pour co-domaine `xsd:double`, d'où la présence d'une traduction. La partie externe consiste alors à appliquer les patrons de la catégorie Traduction (cf. ①). La partie interne consiste à appliquer ceux de la catégorie Format (cf. ② à ⑥). Les patrons des catégories Chemin et Traitement ne sont pas utiles ici. L'évolution de la construction de la requête est détaillée ci-après :

① $Patron_{transty}$

```
{i_t PE_t ?val_1}
BIND (xsd:double(?val_1) AS ?val_0)
```

⇓

② $Patron_{agr}$

```
{
 SELECT (MAX(?val_2) AS ?val_1)
 WHERE {
  i_t PE_t' ?val_2
 }
}
BIND (xsd:double(?val_1) AS ?val_0)
```

⇓

③ $Patron_{ens}$

```
{
 SELECT (MAX(?val_2) AS ?val_1)
 WHERE {
  {
    i_t PE_t1 ?val_2
  } UNION {
    i_t PE_t2 ?val_2
  }
 }
}
BIND (xsd:double(?val_1) AS ?val_0)
```

⇓

④ $Patron_{agr} + Patron_{transfo_2}$

```
{
 SELECT (MAX(?val_2) AS ?val_1)
 WHERE {
  {
   SELECT (MAX(?val_3) AS ?val_2)
   WHERE {
    i_t PE_t1a ?val_3
   }
  } UNION {
   {
    i_t PE_t2a ?val_3a
   }
   {
    i_t PE_t2b ?val_3b
   }
   BIND (?val_3a/?val_3b AS ?val_2)
  }
 }
}
BIND (xsd:double(?val_1) AS ?val_0)
```

⇒

⑤ $Patron_{elem} + Patron_{agr} + Patron_{agr}$

```
{
 SELECT (MAX(?val_2) AS ?val_1)
 WHERE {
  {
   SELECT (MAX(?val_3) AS ?val_2)
   WHERE {
    dbr:Canada dbo:populationDensity ?val_3
   }
  } UNION {
   {
    SELECT (MAX(?val_4) AS ?val_3a)
    WHERE {
     dbr:Canada dbo:populationTotal ?val_4
    }
   }
   {
    SELECT (MIN(?val_4) AS ?val_3b)
    WHERE {
     dbr:Canada
     <dbo:PopulatedPlace/areaTotal> ?val_4
    }
   }
   BIND (?val_3a/?val_3b AS ?val_2)
  }
 }
}
BIND (xsd:double(?val_1) AS ?val_0)
```

⇑

⑥ $Patron_{elem} + Patron_{elem}$

```
{
 SELECT (MAX(?val_2) AS ?val_1)
 WHERE {
  {
   SELECT (MAX(?val_3) AS ?val_2)
   WHERE {
    dbr:Canada dbo:populationDensity ?val_3
   }
  } UNION {
   {
    SELECT (MAX(?val_4) AS ?val_3a)
    WHERE {
     i_t PE_t2a' ?val_4
    }
   }
   {
    SELECT (MIN(?val_4) AS ?val_3b)
    WHERE {
     i_t PE_t2b' ?val_4
    }
   }
   BIND (?val_3a/?val_3b AS ?val_2)
  }
 }
}
BIND (xsd:double(?val_1) AS ?val_0)
```

Fig. 8.6 Déroulement sur un accès direct

① correspond à l'application de $Patron_{transty}$. *transtypage* a été instancié par `xsd:double`.

② représente le début de l'étape d'application des patrons de Format. La PE_t est de la forme Max(i_t $PE_{t'}$ $?val_2$). On instancie $Patron_{agr}$ avec l'agrégat `MAX`.

③ représente l'application de $Patron_{ens}$ instancié avec `UNION`. L'expression "i_t $PE_{t'}$ $?val_2$" est donc transformée en une union de deux PE_t nommées PE_{t_1} et PE_{t_2}. On a donc représenté Max(i_t PE_{t_1} $?val_2$, i_t PE_{t_2} $?val_2$).

④ représente l'application de $Patron_{agr}$ pour la PE_{t_1} instancié avec `MAX`, et de $Patron_{tranfo2}$ pour la PE_{t_2} instancié avec la division de deux PE_t. On a donc représenté Max(Max(i_t $PE_{t_{1_a}}$ $?val_3$), i_t $PE_{t_{2_a}}$ $?val_{3_a}$ / i_t $PE_{t_{2_b}}$ $?val_{3_b}$).

⑤ représente l'application de $Patron_{elem}$ pour la $PE_{t_{1_a}}$ instancié sur la propriété populationDensity, ainsi que de $Patron_{agr}$ instancié avec `MAX` pour la $PE_{t_{2_a}}$ et `MIN` pour la $PE_{t_{2_b}}$. On a donc représenté Max(Max(populationDensity), Max(i_t $PE_{t_{2_{a'}}}$ $?val_4$) / Min(i_t $PE_{t_{2_{b'}}}$ $?val_4$)).

⑥ représente l'application de $Patron_{elem}$ pour les deux PE_t restantes instancié sur les propriétés populationTotal et areaTotal. On a donc représenté la PE_t demandée : Max(Max(populationDensity), Max(populationTotal) / Min(areaTotal)).

Nous considérons maintenant l'exemple 4 page 97. Dans ce cas, on détaille les deux phases parallèles (cf. figure 8.7). La colonne de droite montre la phase externe (Traduction et Traitement) tandis que celle de gauche montre la phase interne (Format et Chemin). L'application des patrons de Format n'est pas détaillée ici car elle est basée sur le même principe que l'exemple précédent. On imbrique le résultat de la phase de la colonne de gauche (partie interne) dans celui de la colonne de droite (partie externe). L'évolution de la construction de la requête est détaillée ci-après :

① représente la requête suite à l'application des patrons de la catégorie Format ($Patron_{agr}$ et $Patron_{elem}$), en suivant le même processus que celui décrit dans la figure 8.6.

② représente le pré-traitement d'insertion du chemin : ajout de l'expression "`Lambton_County` $chemin_1$ $?ind_1$" dans la clause `WHERE` et ajout de la variable $?ind_1$ représentant les $I_t(=I_t^1)$ dans la clause `SELECT` (et donc dans une clause `GROUP BY` ajoutée).

③ représente l'application de $Patron_{chem_{ens}}$, transformant l'expression du chemin en un ensemble de deux chemins élémentaires.

④ représente l'application double de $Patron_{chem_{elem}}$, instancié avec les deux expressions de chemins élémentaires ($isPartOf^{-1}$ et $part$). On obtient ainsi la partie interne permettant de collecter les valeurs de la PE_t considérée en passant par le chemin donné.

❶ représente l'application de $Patron_{transty}$ instancié avec `xsd:double` car il s'agit du co-domaine de la PE_s considérée.

❷ représente l'application de $Patron_{trait_{agr}}$. On obtient ainsi la partie externe qui correspond aux traitements finaux (transtypage et agrégation sur les I_t).

❸ et ❺ permettent de générer la requête finale en imbriquant la partie interne dans la partie externe.

PARTIE INTERNE

① Format de la PE_t ($Patron_{agr}$ + $Patron_{elem}$)

```
SELECT (AVG(?val3) AS ?val2)
WHERE {
  dbr:Lambton_County
  dbp:janPrecipitationDays ?val3.
}
```

⇓

② Pré-traitement

```
SELECT ?ind1 (AVG(?val3) AS ?val2)
WHERE {
  dbr:Lambton_County chemin1 ?ind1
  ?ind1 dbp:janPrecipitationDays ?val3.
}
GROUP BY ?ind1
```

⇓

③ $Patron_{chem_{ens}}$

```
SELECT ?ind1 (AVG(?val3) AS ?val2)
WHERE {
  {it p1^1 ?ind1} UNION {it p2^1 ?ind1}
  ?ind1 dbp:janPrecipitationDays ?val3.
}
GROUP BY ?ind1
```

⇓

④ $Patron_{chem_{elem}} + Patron_{chem_{elem}}$

```
SELECT ?ind1 (AVG(?val3) AS ?val2)
WHERE {
  {?ind1 dbo:isPartOf dbr:Lambton_County}
  UNION
  {dbr:Lambton_County dbo:part ?ind1}
  ?ind1 dbp:janPrecipitationDays ?val3.
}
GROUP BY ?ind1
```

⇒

⑤ Insertion dans la partie externe

Fig. 8.7 Déroulement sur un accès composé

PARTIE EXTERNE

❶ $Patron_{transty}$

```
{it Traitement(p^1.p^2...p^n.PEt) ?val1}
BIND (xsd:double(?val1) AS ?val0)
```

⇓

❷ $Patron_{agr}$

```
{
  SELECT (AVG(?val2) AS ?val1)
  WHERE{
    it p^1.p^2...p^n.PEt ?val2.
  }
}
BIND (xsd:double(?val1) AS ?val0)
```

⇓

❸ Insertion de la partie interne

```
{
  SELECT (AVG(?val2) AS ?val1)
  WHERE{
    SELECT ?ind1 (AVG(?val3) AS ?val2)
    WHERE {
      {?ind1 dbo:isPartOf
      dbr:Lambton_County}
      UNION
      {dbr:Lambton_County dbo:part
      ?ind1}
      ?ind1 dbp:janPrecipitationDays
      ?val3.
    }
    GROUP BY ?ind1
  }
}
BIND (xsd:double(?val1) AS ?val0)
```

Conclusion

Nous avons proposé un modèle complexe pour acquérir des données issues de jeux de données du LOD. Grâce à ce modèle, une génération totalement automatique de requêtes en SPARQL 1.1 est possible et a été implémentée. En effet, dans certains cas où les correspondances entre la source et la cible sont complexes, la requête SPARQL

permettant de collecter des informations sur la cible pour les ajouter dans la source peut être complexe. Il est impératif de générer cette dernière automatiquement pour éviter les erreurs humaines que pourrait faire un concepteur. Ce processus a été mis en place dans le cadre de l'approche Saupodoc. Il a été appliqué avec succès sur des requêtes complexes que les travaux du domaine ne traitaient pas jusqu'alors. Ceci a permis de valider le modèle d'acquisition des données proposé.

Chapitre 9

Conclusion et perspectives de travail

9.1 Conclusion

Cette thèse a présenté une approche permettant de répondre à un problème d'étiquetage automatique de documents décrivant chacun une entité d'un domaine donné, applicable quel que soit le domaine. L'étiquetage est effectué avec une liste de concepts donnés qui sont spécifiques et non explicitement mentionnés dans les documents.

L'approche, nommée SAUPODOC, s'appuie sur une ontologie, et combine des étapes de peuplement et d'enrichissement. L'ontologie a un rôle de pivot entre les différents composants utilisés. Des mécanismes innovants ont été implémentés pour exploiter le Web des données. Les correspondances complexes entre les propriétés de l'ontologie et celles d'une source de données peuvent être définies et des alternatives aux données manquantes sont fournies. Les requêtes qui construisent les assertions de propriété à insérer dans l'ontologie à partir des informations du LOD sont automatiquement générées. L'approche est capable de générer automatiquement des définitions de concepts qui sont à la fois compréhensibles par des humains et interprétables par des machines.

Les résultats montrent clairement le bénéfice par rapport à des classifieurs connus et la pertinence de l'approche à base d'ontologie reposant sur une combinaison particulière de plusieurs techniques pour étiqueter les documents.

Le système peut évoluer facilement. En effet, pour un domaine déjà traité, de nouveaux documents et/ou de nouveaux concepts cibles peuvent être considérés. En cas de nouveaux documents, comme les définitions sont déjà connues, elles n'ont pas besoin d'être ré-apprises. En cas de nouveaux concepts cibles, le concepteur doit fournir des annotations positives et négatives pour chacun d'eux. Dans les deux cas

(nouveaux documents, nouveaux concepts cibles), l'ontologie initiale peut a priori être gardée, à condition qu'elle comporte les concepts et la terminologie liées aux nouveaux documents et/ou concepts cibles. Si ce n'est pas le cas, il faudra la modifier : ajouts de classes descriptives et propriétés associées, de terminologie, et/ou éventuellement de correspondances avec des ressources externes ; et relancer l'approche. En cas de nouveau domaine, le concepteur doit fournir les entrées nécessaires à l'approche :

- Ontologie du domaine spécifiant les propriétés des documents et les propriétés externes ;
- Spécification des correspondances complexes avec le LOD, des chemins d'accès et des mécanismes d'agrégation ;
- Corpus du domaine ;
- Annotations manuelles des documents du corpus avec les concepts cibles.

Ce travail a été validé par l'entreprise Wepingo. Il fait partie d'une approche plus large pour suggérer à des utilisateurs les produits qui correspondent le plus à leurs besoins spécifiques, en permettant d'identifier les documents liés aux instances de ces concepts.

Notons que l'approche est modulaire. Le contexte peut évoluer sans la remettre en cause. Par exemple, le concepteur peut connaître la définition de certains concepts cibles. Dans ce cas, il peut donner directement les définitions dans l'ontologie initiale et l'étape d'apprentissage des définitions ne se fera pas sur ces concepts cibles. De même, si les documents sont complets, l'étape de complétion ne s'effectuera pas. Si, au contraire, un document est quasiment vide et ne contient que le nom de l'entité, voire sa page DBpedia associée, il n'y aura pas d'extraction à partir du document mais seulement à partir de DBpedia. Par ailleurs, dans un contexte moins publicitaire avec des expressions négatives, l'entièreté de l'approche n'est pas à remettre en question. L'étape d'extraction d'informations à partir du texte sera à améliorer avec un traitement automatique de la langue plus poussé qu'à l'heure actuelle. Cependant, les autres étapes ne seront pas remises en question. Enfin, à l'heure actuelle, le LOD contient beaucoup d'informations sur les entités nommées mais pas vraiment encore sur des produits spécifiques, tels que les appareils photo par exemple. Le LOD n'est donc pas encore exploitable pour des corpus décrivant ce genre de produits. Cependant, une évolution future du LOD avec ces produits semble tout à fait probable. D'ailleurs [Suchanek 2015] explique que l'analyse du code barre d'un produit permet d'avoir déjà des informations sur celui-ci comme sa provenance, et annonce l'insertion de produits dans le Web de données comme un but futur.

9.2 Perspectives

Ce travail donne lieu à plusieurs perspectives de travaux futurs.

9.2.1 Les perspectives à court terme

Tout d'abord, nous aimerions exploiter l'approche SAUPODOC sur d'autres domaines d'application. Les premiers domaines considérés seront les livres et la musique pour lesquels des informations sont disponibles sur le Web des données.

Un travail important pour Wepingo est l'élargissement des définitions obtenues. L'idée consiste à regarder les définitions ne générant pas (ou pas assez) d'annotations positives pour un concept cible donné. Une étape semi-automatique de raffinement permettrait d'élargir automatiquement les définitions problématiques en utilisant l'ontologie jusqu'à ce que celles-ci génèrent suffisamment d'annotations positives. Les définitions raffinées automatiquement seraient proposées au concepteur qui validerait, pour un concept cible donné, celle qui lui semble le mieux convenir.

Une autre perspective serait de concevoir une interface (un langage graphique) basée sur le modèle d'acquisition présenté, pour faciliter la spécification des correspondances et des chemins.

9.2.2 Les perspectives à moyen terme

Dans un second temps, il serait intéressant d'étudier de façon plus approfondie la phase d'utilisation de DBpedia Spotlight. Cet outil a bien fonctionné sur notre corpus de destinations (95% d'associations correctes pour le corpus de destinations cf. Annexe C). Sur d'autres domaines, ce ne sera peut-être pas le cas. Cette phase doit donc faire l'objet d'expérimentations supplémentaires pour tester la qualité des réponses de DBpedia Spotlight.

La tâche de complétion avec les données du LOD n'a été appliquée dans nos expérimentations qu'avec des données issues de DBpedia. Pourtant, le modèle d'acquisition que nous avons défini permet de générer des requêtes pour tout jeu de données du LOD. Il serait intéressant d'expérimenter l'approche avec une complétion utilisant plusieurs jeux de données du LOD.

La spécification des correspondances et des chemins d'accès peut demander un travail laborieux. Il semble difficile d'automatiser complètement ce processus mais nous pouvons étudier comment certaines parties (détection de correspondances complexes, spécification des chemins d'accès) pourraient être davantage automatisées.

9.2.3 Les perspectives à long terme et les problèmes ouverts

La tâche courante d'extraction d'informations à partir des documents est actuellement assez simple. La présence d'un mot-clé dans un texte se traduit automatiquement par

une assertion de propriété. Cela est cohérent avec notre contexte où les documents n'ont pas d'expressions négatives. Cependant, si l'on souhaite étendre le contexte de l'approche à des documents avec des expressions négatives, un tel processus ne sera plus adéquat. Dans ce cas, une phrase contenant un mot-clé devra être analysée avant d'ajouter l'assertion de propriété correspondante.

Le système de règles JAPE devrait être étendu au cas où deux propriétés ontologiques ont le même co-domaine, dans le but de trouver la bonne propriété à peupler par rapport au contexte. Pour pouvoir réfléchir correctement à ce problème, il nous faudra travailler sur des domaines où de tels cas se présenteraient.

Les jeux de données du LOD ayant la particularité d'être interconnectés les uns aux autres via la déclaration de liens *sameAs*, il pourrait également être intéressant d'étudier leur exploitation conjointe, l'accès à une donnée pouvant nécessiter la définition de chemins composés de propriétés de jeux de données différents. En effet, avec SPARQL 1.1, il est devenu possible d'interroger des points d'accès SPARQL distants (avec le mot-clef `SERVICE`) et de les combiner. Ce dernier problème est complexe car il touche aux problèmes de liage des données du LOD qui représentent aujourd'hui un défi au cœur des recherches de la communauté du Web Sémantique, de par la nature hétérogène, distribuée et la qualité variable des données.

Le processus de génération de requêtes a été appliqué avec succès sur des requêtes comportant entre autres des successions d'agrégation, que les travaux du domaine ne traitaient pas jusqu'alors. Ceci a permis de valider le modèle d'acquisition des données proposé. En toute généralité, le modèle permet de trouver davantage de correspondances entre des jeux de données RDF correspondant à des ontologies différentes. Il a été appliqué dans le cadre de l'approche Saupodoc mais peut être utilisé pour d'autres usages, par exemple, pour faciliter la ré-écriture de requêtes et donc permettre l'interrogation de données correspondant à des ontologies différentes.

Références

[Alec *et al.* 2014a] Céline Alec, Chantal Reynaud-Delaître, Brigitte Safar, Zied Sellami et Uriel Berdugo. *Automatic Ontology Population from Product Catalogs.* Dans International Conference on Knowledge Engineering and Knowledge Management, EKAW, volume 8876, pages 1–12. Springer, November 2014.

[Alec *et al.* 2014b] Céline Alec, Brigitte Safar, Chantal Reynaud-Delaître, Zied Sellami et Uriel Berdugo. *Peuplement automatique d'ontologie à partir d'un catalogue de produits.* Dans Ingénierie des Connaissances (IC), pages 87–98, Clermont-Ferrand, France, Mai 2014.

[Alec *et al.* 2016a] Céline Alec, Chantal Reynaud-Delaître et Brigitte Safar. *A Model for Linked Open Data Acquisition and SPARQL Query Generation.* Dans International Conference on Conceptual Structures, ICCS, pages 237–251, Annecy, France, july 2016.

[Alec *et al.* 2016b] Céline Alec, Chantal Reynaud-Delaître et Brigitte Safar. *An Ontology-based Method for Discovering Specific Concepts from Texts via Knowledge Completion.* Dans Workshop on Knowledge Extraction and Semantic Annotation, KESA included in ALLDATA, pages 96–101, February 2016.

[Alec *et al.* 2016c] Céline Alec, Chantal Reynaud-Delaître et Brigitte Safar. *An Ontology-driven Approach for Semantic Annotation of Documents with Specific Concepts.* Dans Extended Semantic Web Conference, ESWC, Lecture Notes in Computer Science, pages 609–624. Springer, June 2016.

[Alec *et al.* 2016d] Céline Alec, Chantal Reynaud-Delaître et Brigitte Safar. *Modèle d'acquisition de données du LOD et génération automatique de requêtes.* Revue des Nouvelles Technologies de l'Information, vol. numéro spécial Open Data : bilans, avancées et nouveaux défis, 2016. À paraître.

[Alec *et al.* 2016e] Céline Alec, Chantal Reynaud-Delaître et Brigitte Safar. *Une approche combinée pour l'enrichissement d'ontologie à partir de textes et de données du LOD.* Dans Extraction et Gestion des Connaissances, EGC, pages 171–182. Hermann-Editions, January 2016.

[Alec *et al.* 2016f] Céline Alec, Chantal Reynaud-Delaître, Brigitte Safar, Zied Sellami et Uriel Berdugo. *Identification des catégories de produits issus de catalogues publicitaires.* Revue d'Intelligence Artificielle, vol. numéro spécial Ingénierie des Connaissances, 2016. À paraître.

[Amardeilh & Damljanovic 2009] Florence Amardeilh et Danica Damljanovic. *Du texte à la connaissance : annotation sémantique et peuplement d'ontologie appliqués à des artefacts logiciels*. Dans Journées Francophones d'Ingénierie des Connaissances (IC), pages 157–168, Hammamet, Tunisie, 2009.

[Amardeilh *et al.* 2005] Florence Amardeilh, Philippe Laublet et Jean-Luc Minel. *Document annotation and ontology population from linguistic extractions*. Dans Proceedings of the 3rd international conference on Knowledge Capture (K-CAP), pages 161–168, New York, NY, USA, 2005. ACM.

[Auer *et al.* 2008] Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak et Zachary Ives. *DBpedia: A Nucleus for a Web of Open Data*. Dans Proceedings of the 6th International Semantic Web Conference (ISWC), volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.

[Aussenac-Gilles *et al.* 2013] Nathalie Aussenac-Gilles, Mouna Kamel, Catherine Comparot et Davide Buscaldi. *Construction d'ontologies à partir de pages web structurées*. Dans Journées Francophones d'Ingénierie des Connaissances (IC), pages 1–17, Lille, France, juillet 2013. AFIA.

[Aussenac-Gilles *et al.* 2014] Nathalie Aussenac-Gilles, Jean Charlet et Chantal Reynaud-Delaître. *Ingénierie des connaissances*. Dans Pierre Marquis, Odile Papini et Henri Prade, editeurs, Panorama de l'Intelligence Artificielle - Ses bases méthodologiques - ses développements - Représentation des connaissances et formalisation des raisonnements, volume 1, page chapitre 20. Cepadues, 2014.

[Baader *et al.* 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi et Peter Patel-Schneider. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[Béchet *et al.* 2011] Nicolas Béchet, Marie-Aude Aufaure et Yves Lechevallier. *Construction et peuplement de structures hiérarchiques de concepts dans le domaine du e-tourisme*. Dans Ingénierie des Connaissances (IC), pages 475–490, 2011.

[Bontcheva *et al.* 2004] Kalina Bontcheva, Valentin Tablan, Diana Maynard et Hamish Cunningham. *Evolving GATE to Meet New Challenges in Language Engineering*. Natural Language Engineering, vol. 10, no. 3/4, pages 349–373, 2004.

[Brin 1998] Sergey Brin. *Extracting Patterns and Relations from the World Wide Web*. Dans WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98, pages 172–183, 1998.

[Burke 2000] R. Burke. *Knowledge-Based Recommender Systems*. Encyclopedia of Library and Information Science, vol. 69, no. 32, 2000.

[Cambria & White 2014] Erik Cambria et Bebo White. *Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]*. IEEE Comp. Int. Mag., vol. 9, no. 2, pages 48–57, 2014.

[Cambria *et al.* 2015] Erik Cambria, Jie Fu, Federica Bisio et Soujanya Poria. *AffectiveSpace 2: Enabling Affective Intuition for Concept-Level Sentiment Analysis*. Dans Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., pages 508–514, 2015.

[Caraballo 1999] Sharon A. Caraballo. *Automatic Construction of a Hypernym-Labeled Noun Hierarchy From Text*. Dans Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, 1999.

[Charlet *et al.* 2004] Jean Charlet, Bruno Bachimont et Raphaël Troncy. *Ontologies pour le Web sémantique*. I3 (Information, Intéraction, Intelligence), Numéro Hors Série "Web sémantique", pages 43–63, 2004.

[Cheng & Roth 2013] Xiao Cheng et Dan Roth. *Relational Inference for Wikification*. Dans Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1787–1796, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[Chitsaz 2013] Mahsa Chitsaz. *Enriching Ontologies through Data*. Dans Doctoral Consortium co-located with ISWC, Sydney, Australia, pages 1–8. CEUR-WS.org, 2013.

[Choueka 1988] Yaacov Choueka. *Looking for Needles in a Haystack or Locating Interesting Collocational Expressions in Large Textual Databases.* Dans Christian Fluhr et Donald E. Walker, editeurs, RIAO, pages 609–624. CID, 1988.

[Cimiano & Völker 2005] Philipp Cimiano et Johanna Völker. *Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery*. Dans NLDB, volume 3513, pages 227–238, Alicante, Spain, 2005. Springer.

[Cimiano *et al.* 2004] Philipp Cimiano, Siegfried Handschuh et Steffen Staab. *Towards the Self-annotating Web*. Dans Proceedings of the 13th International Conference on World Wide Web, WWW '04, pages 462–471, New York, NY, USA, 2004. ACM.

[Cimiano *et al.* 2005a] P. Cimiano, A. Hotho et S. Staab. *Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis*. Journal of Artificial Intelligence Research, vol. 24, pages 305–339, 2005.

[Cimiano *et al.* 2005b] Philipp Cimiano, Günter Ladwig et Steffen Staab. *Gimme' the Context: Context-driven Automatic Semantic Annotation with C-PANKOW*. Dans Proceedings of the 14th International Conference on World Wide Web, WWW '05, pages 332–341, New York, NY, USA, 2005. ACM.

[Cimiano *et al.* 2006] Philipp Cimiano, Johanna Völker et Rudi Studer. *Ontologies on Demand? - A Description of the State-of-the-Art, Applications, Challenges and Trends for Ontology Learning from Text*. Information, Wissenschaft und Praxis, vol. 57, no. 6-7, pages 315–320, Octobre 2006.

[Cimiano 2006] Philipp Cimiano. Ontology learning and population from text: Algorithms, evaluation and applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Corman *et al.* 2015] Julien Corman, Nathalie Aussenac-Gilles et Laure Vieu. *Trimming a consistent OWL knowledge base, relying on linguistic evidence (regular paper).* Dans Workshop on Language and Ontologies, page (on line), http://www.abdn.ac.uk, avril 2015. University of Aberdeen.

[Correndo *et al.* 2010] Gianluca Correndo, Manuel Salvadores, Ian Millard, Hugh Glaser et Nigel Shadbolt. *SPARQL Query Rewriting for Implementing Data Integration over Linked Data.* Dans Proceedings of the 2010 EDBT/ICDT Workshops, EDBT '10, pages 4:1–4:11, New York, NY, USA, 2010. ACM.

[Cortes & Vapnik 1995] Corinna Cortes et Vladimir Vapnik. *Support-Vector Networks.* Mach. Learn., vol. 20, no. 3, pages 273–297, Septembre 1995.

[Cunningham *et al.* 2011] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li et Wim Peters. Text Processing with GATE. University of Sheffield Department of Computer Science, 2011.

[Daiber *et al.* 2013] Joachim Daiber, Max Jakob, Chris Hokamp et Pablo N. Mendes. *Improving Efficiency and Accuracy in Multilingual Entity Extraction.* Dans Proceedings of the 9th International Conference on Semantic Systems (I-Semantics), 2013.

[Esposito *et al.* 2004] Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano et Giovanni Semeraro. *Knowledge-Intensive Induction of Terminologies from Metadata.* Dans The Semantic Web - ISWC 2004: Third International Semantic, pages 441–455, 2004.

[Euzenat & Shvaiko 2013] Jérôme Euzenat et Pavel Shvaiko. Ontology matching. Springer-Verlag, Heidelberg (DE), 2nd édition, 2013.

[Euzenat *et al.* 2007] Jérôme Euzenat, François Scharffe et Antoine Zimmermann. Expressive alignment language and implementation. Rapport interne, INRIA, 2007.

[Fanizzi *et al.* 2008] Nicola Fanizzi, Claudia d'Amato et Floriana Esposito. *DL-FOIL Concept Learning in Description Logics.* Dans ILP, pages 107–121, 2008.

[Faure & Nedellec 1999] D. Faure et C. Nedellec. *Knowledge Acquisition of Predicate Argument Structures from Technical Texts using Machine Learning: The System ASIUM.* Dans Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW), Dagstuhl Castle, Germany, 1999.

[Galárraga *et al.* 2013] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose et Fabian M. Suchanek. *AMIE: association rule mining under incomplete evidence in ontological knowledge bases.* Dans WWW '13, Rio de Janeiro, Brazil, May 13-17, pages 413–422, 2013.

[Gangemi *et al.* 2012] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti et Paolo Ciancarini. *Automatic Typing of DBpedia Entities*. Dans The Semantic Web - ISWC, Boston, MA, USA, November 11-15, 2012, pages 65–81, 2012.

[Garon *et al.* 2002] Denise Garon, Rolande Filion et Robert Chiasson. Le système ESAR: guide d'analyse, de classification et d'organisation d'une collection de jeux et jouets. Editions ASTED, 2002.

[Gillet *et al.* 2013] Pascal Gillet, Cássia Trojahn dos Santos, Ollivier Haemmerlé et Camille Pradel. *Complex Correspondences for Query Patterns Rewriting*. Dans Ontology Matching at ISWC 2013, Sydney, Australia. CEUR Workshop Proceedings, 2013.

[Gruber 1993] Thomas R. Gruber. *A Translation Approach to Portable Ontology Specifications*. Knowledge Acquisition, vol. 5, no. 2, pages 199–220, Juin 1993.

[Hall *et al.* 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann et Ian H. Witten. *The WEKA data mining software: an update*. SIGKDD Explorations, vol. 11, no. 1, pages 10–18, 2009.

[Handschuh *et al.* 2002] Siegfried Handschuh, Steffen Staab et Fabio Ciravegna. *S-CREAM - Semi-automatic CREAtion of Metadata*. Dans Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, EKAW '02, pages 358–372, London, UK, UK, 2002. Springer-Verlag.

[Harris *et al.* 2013] Steve Harris, Andy Seaborne et Eric Prud'hommeaux. *SPARQL 1.1 Query Language (2013)*. In W3C Recommendation (2013), 2013.

[Hastie & Tibshirani 1998] Trevor Hastie et Robert Tibshirani. *Classification by Pairwise Coupling*. Dans Michael I. Jordan, Michael J. Kearns et Sara A. Solla, editeurs, Advances in Neural Information Processing Systems, volume 10. MIT Press, 1998.

[Horridge *et al.* 2006] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens et Hai Wang. *The Manchester OWL Syntax.* Dans OWLED2006 Second Workshop on OWL Experiences and Directions, volume 216, Athens, Georgia, USA„ 2006.

[Joachims 1998] Thorsten Joachims. *Text Categorization with Suport Vector Machines: Learning with Many Relevant Features*. Dans Proceedings of the 10th European Conference on Machine Learning, ECML '98, pages 137–142, London, UK, 1998. Springer-Verlag.

[Kaufmann & Bernstein 2010] Esther Kaufmann et Abraham Bernstein. *Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, no. 4, pages 377–393, 2010.

[Keerthi *et al.* 2001] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya et K.R.K. Murthy. *Improvements to Platt's SMO Algorithm for SVM Classifier Design.* Neural Computation, vol. 13, no. 3, pages 637–649, 2001.

[Kessler *et al.* 2012] Rémy Kessler, Nicolas Béchet, Mathieu Roche, Juan Manuel Torres Moreno et Marc El-Bèze. *A hybrid approach to managing job offers and candidates.* Information Processing and Management, vol. 48, no. 6, pages 1124–1135, 2012.

[Lehmann & Hitzler 2007] Jens Lehmann et Pascal Hitzler. *A Refinement Operator Based Learning Algorithm for the ALC Description Logic.* Dans Proceedings of the 17th International Conference on Inductive Logic Programming (ILP), 2007.

[Lehmann & Hitzler 2010] Jens Lehmann et Pascal Hitzler. *Concept learning in description logics using refinement operators.* Machine Learning, vol. 78, no. 1-2, pages 203–250, 2010.

[Lehmann *et al.* 2011] Jens Lehmann, Sören Auer, Lorenz Bühmann et Sebastian Tramp. *Class expression learning for ontology engineering.* Journal of Web Semantics, vol. 9, pages 71–81, 2011.

[Lehmann 2009] Jens Lehmann. *DL-Learner: Learning Concepts in Description Logics.* Journal of Machine Learning Research, vol. 10, pages 2639–2642, 2009.

[Ma & Distel 2013a] Yue Ma et Felix Distel. *Concept Adjustment for Description Logics.* Dans Proceedings of the Seventh International Conference on Knowledge Capture, K-CAP '13, pages 65–72, New York, NY, USA, 2013. ACM.

[Ma & Distel 2013b] Yue Ma et Felix Distel. *Learning Formal Definitions for Snomed CT from Text.* Dans Proc. of Artificial Intelligence in Medicine, pages 73–77. Springer Berlin Heidelberg, 2013.

[Maedche & Staab 2004] Alexander Maedche et Steffen Staab. *Ontology Learning.* Dans Handbook on Ontologies, pages 173–190. 2004.

[Makris *et al.* 2010] Konstantinos Makris, Nektarios Gioldasis, Nikos Bikakis et Stavros Christodoulakis. *Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources.* Dans Proceedings of the 2010 International Conference on On the Move to Meaningful Internet Systems: Part II, OTM'10, pages 1108–1117, Berlin, Heidelberg, 2010. Springer-Verlag.

[Makris *et al.* 2012] Konstantinos Makris, Nikos Bikakis, Nektarios Gioldasis et Stavros Christodoulakis. *SPARQL-RW: Transparent Query Access over Mapped RDF Data Sources.* Dans Proceedings of the 15th International Conference on Extending Database Technology, EDBT 2012, EDBT '12, pages 610–613. ACM, 2012.

[Manning & Schütze 1999] Christopher D. Manning et Hinrich Schütze. Foundations of statistical natural language processing. The MIT Press, Cambridge, Massachusetts, 1999.

[Manning *et al.* 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard et David McClosky. *The Stanford CoreNLP Natural Language Processing Toolkit*. Dans 52nd ACL: System Demonstrations, pages 55–60, 2014.

[McGuinness & van Harmelen 2004] Deborah L. McGuinness et Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C recommendation, World Wide Web Consortium, February 2004.

[Mendes *et al.* 2011] Pablo N. Mendes, Max Jakob, Andrés García-Silva et Christian Bizer. *DBpedia Spotlight: Shedding Light on the Web of Documents*. Dans I-Semantics, pages 1–8, NY, USA, 2011. ACM.

[Nadeau & Sekine 2007] David Nadeau et Satoshi Sekine. *A survey of named entity recognition and classification*. Linguisticae Investigationes, vol. 30, pages 3–26, 2007.

[Oren *et al.* 2006] Eyal Oren, Knud Möller, Simon Scerri, Siegfried Handschuh et Michael Sintek. *What are Semantic Annotations?* Rapport technique, DERI Galway, 2006.

[Paulheim & Bizer 2014] Heiko Paulheim et Christian Bizer. *Improving the Quality of Linked Data Using Statistical Distributions*. Int. J. Semantic Web Inf. Syst., vol. 10, no. 2, pages 63–86, 2014.

[Pazzani & Billsus 2007] Michael Pazzani et Daniel Billsus. *Content-Based Recommendation Systems*. Dans Peter Brusilovsky, Alfred Kobsa et Wolfgang Nejdl, editeurs, The Adaptive Web, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, Berlin / Heidelberg, 2007.

[Pekar & Staab 2002] Viktor Pekar et Steffen Staab. *Taxonomy Learning - Factoring the Structure of a Taxonomy into a Semantic Classification Decision*. Dans 19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002, 2002.

[Pereira Nunes *et al.* 2013] Bernardo Pereira Nunes, Alexander Mera, Marco Antonio Casanova, Besnik Fetahu, Luiz André P. Paes Leme et Stefan Dietze. *Complex Matching of RDF Datatype Properties*. Dans Hendrik Decker, Lenka Lhotska, Sebastian Link, Josef Basl et A Min Tjoa, editeurs, Database and Expert Systems Applications, DEXA 2013, Prague, Czech Republic, August 26-29, volume 8055 of *LNCS*, pages 195–208. Springer Berlin Heidelberg, 2013.

[Petasis *et al.* 2011] Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara et Elias Zavitsanos. *Ontology Population and Enrichment: State of the Art*. Dans Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, pages 134–166, 2011.

[Petasis *et al.* 2013] Georgios Petasis, Ralf Möller et Vangelis Karkaletsis. *BOEMIE: Reasoning-based Information Extraction*. Dans Proceedings of the 1st Workshop on Natural Language Processing and Automated Reasoning co-located with 12th

International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), volume 1044 of *CEUR Workshop Proceedings*, pages 60–75, A Corunna, Spain, September 2013. CEUR-WS.org.

[Platt 1998] J. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization.* Dans B. Schoelkopf, C. Burges et A. Smola, editeurs, Advances in Kernel Methods - Support Vector Learning. MIT Press, 1998.

[Popa *et al.* 2002] Lucian Popa, Yannis Velegrakis, Renée J. Miller, Mauricio A. Hernández et Ronald Fagin. *Translating Web Data.* Dans VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China, pages 598–609, 2002.

[Popov *et al.* 2004] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov et Angel Kirilov. *KIM – a Semantic Platform for Information Extraction and Retrieval.* Natural Language Engineering, vol. 10, no. 3-4, pages 375–392, Septembre 2004.

[Quinlan 1986] J. Ross Quinlan. *Induction of decision trees.* Journal of Machine Learning, vol. 1, pages 81–106, 1986.

[Quinlan 1993] Ross Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Ratinov *et al.* 2011] Lev Ratinov, Dan Roth, Doug Downey et Mike Anderson. *Local and Global Algorithms for Disambiguation to Wikipedia.* Dans Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 1375–1384, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Reeve 2005] Lawrence Reeve. *Survey of semantic annotation platforms.* Dans ACM Symposium on Applied Computing, pages 1634–1638. ACM Press, 2005.

[Ricci *et al.* 2011] Francesco Ricci, Lior Rokach, Bracha Shapira et Paul B. Kantor. Recommender systems handbook. Springer, New York; London, 2011.

[Ritze *et al.* 2010] Dominique Ritze, Johanna Völker, Christian Meilicke et Ondrej Svób-Zamazal. *Linguistic analysis for complex ontology matching.* Dans Proceedings of the 5th International Workshop on Ontology Matching (OM), volume 689 of *CEUR Workshop Proceedings*, Shanghai, China, 2010. CEUR-WS.org.

[Salton & Buckley 1988] Gerard Salton et Christopher Buckley. *Term-weighting approaches in automatic text retrieval.* Information Processing and Management, vol. 24, no. 5, pages 513–523, 1988.

[Salton & McGill 1986] Gerard Salton et Michael J. McGill. Introduction to modern information retrieval. McGraw-Hill, Inc., New York, NY, USA, 1986.

[Salton *et al.* 1975] Gerard Salton, Andrew K. C. Wong et Chung-Shu Yang. *A vector space model for automatic indexing.* Commun. ACM, vol. 18, no. 11, pages 613–620, Novembre 1975.

[Schafer *et al.* 2007] J. Ben Schafer, Dan Frankowski, Jon Herlocker et Shilad Sen. *Collaborative filtering recommender systems.* Dans Peter Brusilovsky, Alfred Kobsa et Wolfgang Nejdl, editeurs, The adaptive web, pages 291–324. Springer-Verlag, Berlin, Heidelberg, 2007.

[Scharffe *et al.* 2014] François Scharffe, Ondřej Zamazal et Dieter Fensel. *Ontology Alignment Design Patterns.* Knowledge and Information Systems, vol. 40, no. 1, pages 1–28, 2014.

[Scharffe 2009] François Scharffe. *Correspondence Patterns Representation.* Thèse de doctorat, Université d'Innsbrück, 2009.

[Shearer *et al.* 2008] Rob Shearer, Boris Motik et Ian Horrocks. *HermiT: A Highly-Efficient OWL Reasoner.* Dans OWLED, volume 432. CEUR-WS.org, 2008.

[Shekarpour *et al.* 2011] Saeedeh Shekarpour, Soren Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann et Claus Stadler. *Keyword-Driven SPARQL Query Generation Leveraging Background Knowledge.* Dans Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '11, pages 203–210, Washington, DC, USA, 2011. IEEE Computer Society.

[Simonic *et al.* 2013] Klemen Simonic, Jan Rupnik et Primoz Skraba. *Missing Properties in Linked Data Datasets.* Dans KDD '13: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013.

[Sirin *et al.* 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur et Yarden Katz. *Pellet: A practical OWL-DL reasoner.* Journal of Web Semantics, vol. 5, no. 2, pages 51–53, 2007.

[Sleeman & Finin 2013] Jennifer Sleeman et Tim Finin. *Type Prediction for Efficient Coreference Resolution in Heterogeneous Semantic Graphs.* Dans 2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013, pages 78–85, 2013.

[Suchanek *et al.* 2009] Fabian M. Suchanek, Mauro Sozio et Gerhard Weikum. *SOFIE: a Self-Organizing Framework for Information Extraction.* Dans World Wide Web Conference (WWW), pages 631–640, Madrid, Spain, 2009. ACM.

[Suchanek 2015] Fabian M. Suchanek. *A Hitchhiker's Guide to Ontology.* Dans Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume, page 629, 2015.

[Thiéblin *et al.* 2016] Élodie Thiéblin, Fabien Amarger, Ollivier Haemmerlé, Nathalie Hernandez et Cassia Trojahn. *Vers une approche pour la reformulation automatique de requêtes à partir d'alignements complexes.* Dans Journées Francophones d'Ingénierie des Connaissances (IC), pages 35–46, Montpellier, France, 2016.

[Toussaint & Tenier 2007] Yannick Toussaint et Sylvain Tenier. *Annotation sémantique par classification.* Dans Francky Trichet, editeur, Ingénierie des Connaissances, volume 2 of *Ingénierie des Connaissances - 18èmes journées francophones*, pages 85–96, Grenoble, France, Juillet 2007. Cépadues éditions.

[Tsarkov & Horrocks 2006] Dmitry Tsarkov et Ian Horrocks. *FaCT++ Description Logic Reasoner: System Description.* Dans Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006), volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297, Berlin, Heidelberg, 2006. Springer.

[Unger *et al.* 2012] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber et Philipp Cimiano. *Template-based Question Answering over RDF Data.* Dans Mohand-Said Hacid, Zbigniew W. Ras, Djamel A. Zighed et Yves Kodratoff, editeurs, Proceedings of the 21st International Conference on World Wide Web, WWW 2012, WWW '12, pages 639–648. ACM, 2012.

[Vargas-Vera *et al.* 2002] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt et Fabio Ciravegna. *MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup.* Dans EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, pages 379–391, London, UK, 2002. Springer-Verlag.

[Völker *et al.* 2007] Johanna Völker, Pascal Hitzler et Philipp Cimiano. *Acquisition of OWL DL Axioms from Lexical Resources.* Dans The Semantic Web: Research and Applications: 4th European Semantic Web Conference (ESWC), pages 670–685, Innsbruck, Austria, 2007. Springer.

[Völker 2009] Johanna Völker. Learning Expressive Ontologies, volume 2 of *Studies on the Semantic Web.* IOS Press, 2009.

[Yelagina & Panteleyev 2014] Nataliya Yelagina et Michail Panteleyev. *Deriving of Thematic Facts from Unstructured Texts and Background Knowledge.* Dans Knowledge Engineering and the Semantic Web: 5th International Conference (KESW), volume 468, pages 208–218. Springer, 2014.

[Yosef *et al.* 2011] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol et Gerhard Weikum. *AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables.* Dans Proceedings of the 37th International Conference on Very Large Databases, (VLDB 2011), pages 1450–1453, 2011.

[Yuan *et al.* 2014] Wancheng Yuan, Elena Demidova, Stefan Dietze et Xuan Zhou. *Analyzing Relative Incompleteness of Movie Descriptions in the Web of Data: A Case Study.* Dans CEUR, editeur, Proceedings of the ISWC 2014 Posters & Demonstrations Track, a track within the 13th International Semantic Web Conference (ISWC 2014), volume 1272, pages 197–200, October 2014.

[Zaveri *et al.* 2014] Amrapali Zaveri, Andrea Maurino et Laure-Berti Equille. *Web Data Quality: Current State and New Challenges*. Int. J. Semant. Web Inf. Syst., vol. 10, pages 1–6, 2014.

Annexe A

Patron de règle JAPE générique

Nous montrons ici le patron de règle JAPE qui va permettre de créer des règles JAPE qui permettront d'introduire les assertions de propriétés relatives aux propriétés dites *des documents*.

Le patron JAPE générique contient deux variables qui sont instanciées pour chaque *propriété des documents* :

- lignes 1 et 30, PROP_NAME sera automatiquement remplacé par le nom d'une propriété des documents ;
- ligne 29, RANGE_NAME sera automatiquement remplacé par le nom du co-domaine de la propriété considérée ;

```
1  Rule: AssociatedPropertiesPopulation_PROP_NAME
2  ({Lookup}):lookup
3  -->
4  :lookup{
5     //find the annotation matched by LHS
6     //we know the annotation set returned
7     //will always contain a single annotation
8     Annotation lookupAnn = lookupAnnots.iterator().next();
9
10    //find the class of the lookup
11    String type = (String)lookupAnn.getFeatures().get("type");
12    if(type.equals("instance")){
13
14       String lookupText = gate.Utils.stringFor(doc, lookupAnn);
15       if(lookupText.equals("-") || lookupText.equals("_")){
16          return;
17       }
18       boolean containDigit=false;
19       for(int i=0; i<lookupText.length(); i++){
20          if(Character.isDigit(lookupText.charAt(i))){
21             containDigit=true;
22             break;
23          }
24       }
25       if(containDigit){
26          return;
27       }
28
29       String associatedClassName="RANGE_NAME";
```

```
30          String associatedPropertyName="PROP_NAME";
31
32          String associatedIndividual = (String)lookupAnn.getFeatures().get("URI");
33          OURI uri=ontology.createOURI(associatedIndividual);
34          OInstance instanceRange=ontology.getOInstance(uri);
35          if(instanceRange == null){
36             System.err.println("Error instance \"" + uri + "\" does not exist!");
37             return;
38          }
39
40          OClass associatedClass =
                ontology.getOClass(ontology.createOURIForName(associatedClassName));
41          if(associatedClass == null){
42             System.err.println("Error class \"" + associatedClassName + "\" does not
                   exist!");
43             return;
44          }
45          //instance range type is associatedClass?
46          if(instanceRange.getOClasses(OConstants.Closure.TRANSITIVE_CLOSURE).contains(
                associatedClass)){
47             String docName = OUtils.toResourceName(doc.getName().replace(".xml",""));
48             OInstance documentInstance =
                      ontology.getOInstance(ontology.createOURIForName(docName));
49             ObjectProperty associatedProperty = ontology.getObjectProperty(
                      ontology.createOURIForName(associatedPropertyName));
50             if(documentInstance == null){
51                System.err.println("Error instance \"" + docName + "\" does not exist!");
52                return;
53             }
54             if(associatedProperty == null){
55                System.err.println("Error instance \"" + associatedPropertyName + "\"
                      does not exist!");
56                return;
57             }
58             try {
59                documentInstance.addObjectPropertyValue(associatedProperty,
                      instanceRange);
60             } catch (InvalidValueException e) {
61                // TODO Auto-generated catch block
62                e.printStackTrace();
63             }
64          }
65       }
66    }
```

Annexe B

Détails des expérimentations

B.1 Définitions obtenues et histogrammes détaillés

Cette section montre les définitions des concepts cibles telles qu'elles ont été apprises par SAUPODOC ainsi que les histogrammes détaillés de l'exactitude, la F-mesure, la précision et le rappel pour les corpus de destinations et de films. Dans les tableaux B.1 à B.4 qui présentent les définitions, la colonne "nb^+" désigne le nombre d'exemples positifs pour chacun des concepts cibles considérés. Les Figures B.1 à B.8 montrent les histogrammes détaillés par concept cible.

Rappelons que le corpus des destinations a été annoté avec une ontologie relativement détaillée (161 classes descriptives) mais qu'il était de taille limitée (80 documents) alors qu'à l'inverse le corpus de films comprenait 10 000 documents annotés avec une ontologie très simple (5 classes descriptives).

Le fait que le corpus de destinations ne comprenne que peu de documents nous a permis de vérifier manuellement la qualité des assertions de propriétés introduites aussi bien à partir des documents que de DBpedia mais aussi de l'annoter, là aussi manuellement, avec des concepts cibles variés (39) pour tester la forme des règles qui pouvaient être apprises.

En revanche, ce faible nombre de documents limite aussi pour certains concepts cibles, le nombre d'exemples positifs. Ainsi certains concepts cibles ont moins d'une dizaine d'exemples positifs en tout, ce qui signifie encore moins d'exemples positifs dans l'ensemble d'apprentissage, qui ne contenait que les 2/3 des documents du corpus.

Parmi les définitions présentées, toutes celles qui ont été obtenues à partir d'un nombre important d'exemples positifs sont intuitivement satisfaisantes ce qui n'est pas le cas de celles apprises sur un nombre d'exemples trop faible, en particulier quand elles mettent en jeu un grand nombre d'opérateurs.

n°	Nom de concept cible	Définition trouvée par SAUPODOC	nb$^+$
1	Destination pour enfants	isIdealFor some WithKids	46
2	Destination un peu culturelle	hasCulture min 3 Thing	43
3	Destination moyennement culturelle	hasCulture min 4 Thing	34
4	Destination très culturelle	hasCulture min 6 Thing	18
5	Destination d'aventure	hasActivity some Adventure	36
6	Destination animée	hasActivity some Animation	68
7	Destination baignade mi-saison été sans trop de pluie	(hasCulture some (Architecture or History)) and (hasActivity value _diving) and (hasEnvironment value _desert) and (isIdealFor value _family)	6
8	Destination baignade mi-saison été	(hasEnvironment some Environment) and (hasWeather min 4 (avgTemperatureC some double[>= "23,98335"^^double]))	20
9	Destination baignade mi-saison hiver sans trop de pluie	(hasEnvironment value _ocean) and (hasEnvironment min 2 (Cascade or Jungle))	3
10	Destination baignade mi-saison hiver	(hasEnvironment some Activity) and (hasWeather min 10 (avgTemperatureC some double[>= "23,98335"^^double]))	13
11	Destination baignade plein été sans trop de pluie	(hasEnvironment some Bathing) and (hasWeather some ((avgTemperatureC some double[>= "21,6869"^^double]) and (precipitationMm some double[<= "48,28"^^double]))) and (hasWeather some (avgTemperatureC some double[<= "21,6869"^^double]))	34
12	Destination baignade plein été	(hasWeather some (avgTemperatureC some double[>= "23,98335"^^double])) and (hasEnvironment value _beach)	47
13	Destination baignade plein hiver sans trop de pluie	(hasEnvironment some Activity) and (hasEnvironment some Cascade) and (hasEnvironment value _jungle)	6
14	Destination baignade plein hiver	(hasEnvironment some Activity) and (hasWeather min 10 (avgTemperatureC some double[>= "23,98335"^^double]))	13
15	Destination belle	hasEnvironment value _beauty	56
16	Destination côtière	hasEnvironment some Coast	67

Tableau B.1 Les concepts cibles et leurs définitions trouvées par SAUPODOC dans le domaine des destinations (1/3)

n°	Nom de concept cible	Définition trouvée par SAUPODOC	nb$^+$
17	Destination désertique	hasEnvironment some Desert	19
18	Destination randonnée	(hasCulture some (Basilica or Pagoda)) and (hasEnvironment value _nature) and (isIdealFor value _family)	5
19	Destination montagne	(hasEnvironment some Mountain) and (hasEnvironment some QualityEnvironment)	44
20	Destination naturelle	(hasEnvironment some Aquatic) and (hasEnvironment some Nature)	59
21	Destination avec vie nocturne	hasActivity some Nightlife	59
22	Destination vieille ville ou shopping	(hasActivity value _oldTown) or (hasEnvironment value _shopping)	50
23	Destination promenade	hasActivity some Promenade	41
24	Destination relaxante	hasActivity some Relaxation	65
25	Destination shopping	hasEnvironment value _shopping	28
26	Destination urbaine	hasEnvironment some Urban	34
27	Destination parcs aquatiques mi-saison été sans trop de pluie	(isIdealFor some Couple) and (hasActivity value _spectacle)	2
28	Destination parcs aquatiques mi-saison été	(hasCulture some (Architecture or Museum)) and (hasEnvironment some FlowingWater) and (hasEnvironment value _desert) and (isIdealFor value _family)	3
29	Destination parcs aquatiques plein été sans trop de pluie	(hasEnvironment some (Sea or SmallWaterBody)) and (hasActivity value _waterpark)	17
30	Destination parcs aquatiques plein été	(hasActivity some (OldTown or Tranquillity)) and (hasActivity value _waterpark)	19
31	Destination sports aquatiques mi-saison été sans trop de pluie	(hasCulture some (Architecture or History)) and (hasEnvironment value _desert) and (isIdealFor value _family)	4
32	Destination sports aquatiques mi-saison été	(hasActivity some Watersport) and (hasWeather min 4 (avgTemperatureC some double[>= "23,98335"^^double]))	12

Tableau B.2 Les concepts cibles et leurs définitions trouvées par SAUPODOC dans le domaine des destinations (2/3)

n°	Nom de concept cible	Définition trouvée par SAUPODOC	nb^+
33	Destination sports aquatiques mi-saison hiver sans trop de pluie	(hasEnvironment some Cascade) and (hasCulture value _museum) and (hasEnvironment value _rainForest)	2
34	Destination sports aquatiques mi-saison hiver	(hasWeather some ((avgTemperatureC some double[>= "26,10555"^^double]) and (precipitationMm some double[<= "102,25200000000001"^^double]))) and (hasEnvironment value _jungle) and (hasCulture min 2 Thing)	7
35	Destination sports aquatiques plein été sans trop de pluie	(hasActivity some Watersport) and (hasEnvironment some Bathing) and (hasEnvironment some QualityEnvironment) and (isIdealFor some (WithKids or WithoutKids)) and (hasActivity value _tranquillity)	26
36	Destination sports aquatiques plein été	(hasActivity some Watersport) and (hasWeather some (avgTemperatureC some double[>= "23,98335"^^double]))	33
37	Destination sports aquatiques plein hiver sans trop de pluie	(hasEnvironment some Bay) and (hasEnvironment value _jungle)	4
38	Destination sports aquatiques plein hiver	(hasWeather some ((avgTemperatureC some double[>= "26,10555"^^double]) and (precipitationMm some double[<= "102,25200000000001"^^double]))) and (hasEnvironment value _jungle) and (hasCulture min 2 Thing)	7
39	Destination sport d'hiver	(hasActivity some WinterSport) and (hasActivity min 2 Culture)	6

Tableau B.3 Les concepts cibles et leurs définitions trouvées par SAUPODOC dans le domaine des destinations (3/3)

Nom de concept cible	Définition trouvée par SAUPODOC	nb$^+$
American_films	((isFromCountry value _United_States) and (runtimeInSeconds some double[<= "6765,0"^^double])) or (isFromCountry value _United___States)	1885
American_silent_feature_films	(isFromCountry value _United___States) and (hasLanguage min 2 Thing)	202
British_films	isFromCountry value _United___Kingdom	580
English-language_films	(isFromCountry some Country) and (hasLanguage value _English)	2706
French-language_films	(isFromCountry some Language) and (hasLanguage value _French)	411
French_films	isFromCountry value _France	284
Hindi-language_films	hasLanguage value _Hindi	757
Indian_films	isFromCountry value _India	1895
Italian_films	(hasLanguage some Language) and (isFromCountry value _Italy)	339
Japanese_films	isFromCountry value _Japan	250
Spanish-language_films	(hasLanguage value _Spanish) or (hasLanguage value _Spanish_language)	294
Tamil-language_films	(hasLanguage value _Tamil) or (hasLanguage value _Tamil_language)	346

Tableau B.4 Les concepts cibles et leurs définitions trouvées par SAUPODOC dans le domaine des films

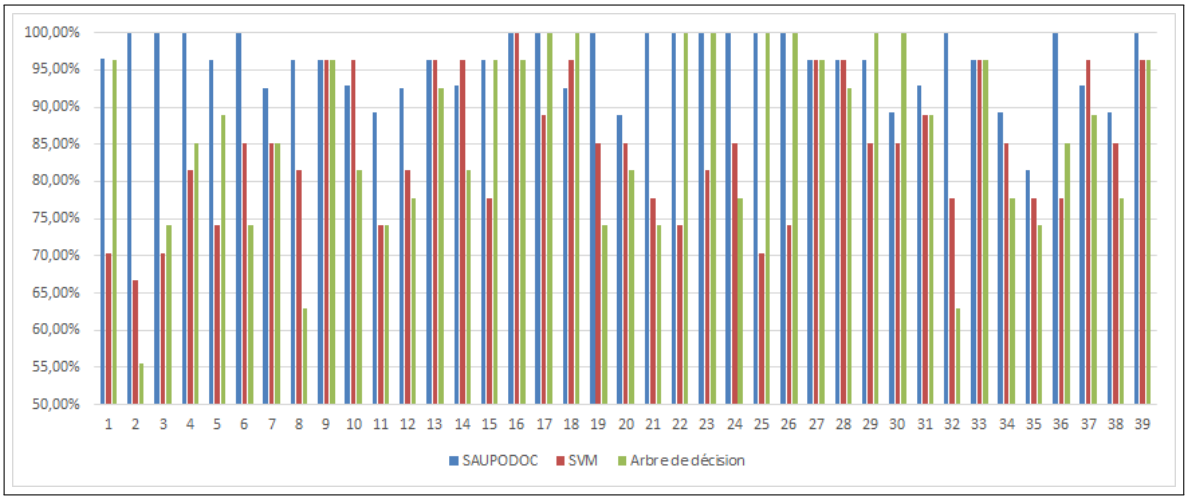


Fig. B.1 Exactitude des 39 concepts cibles du domaine des destinations

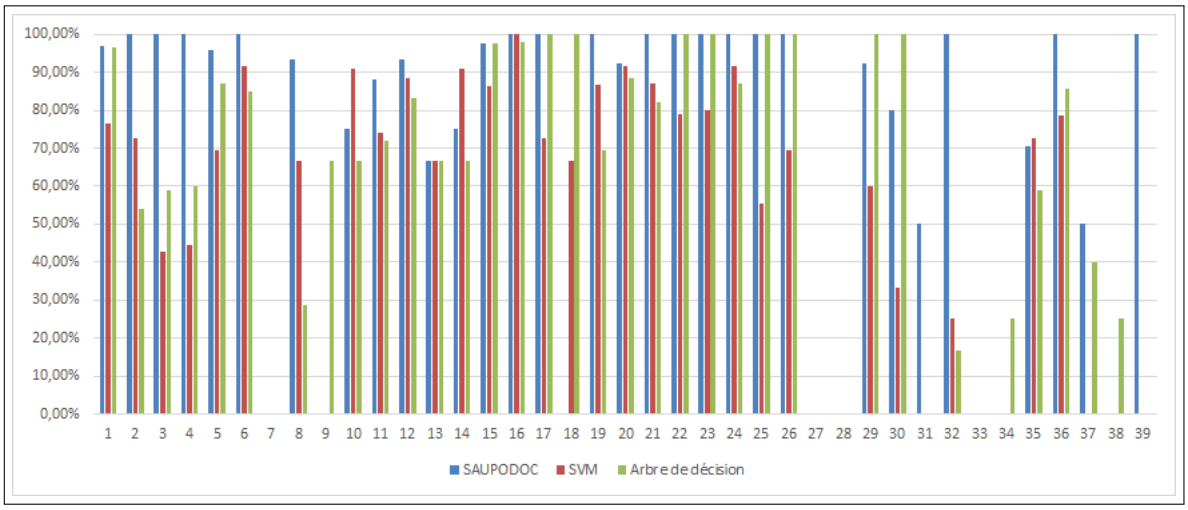


Fig. B.2 F-mesure des 39 concepts cibles du domaine des destinations

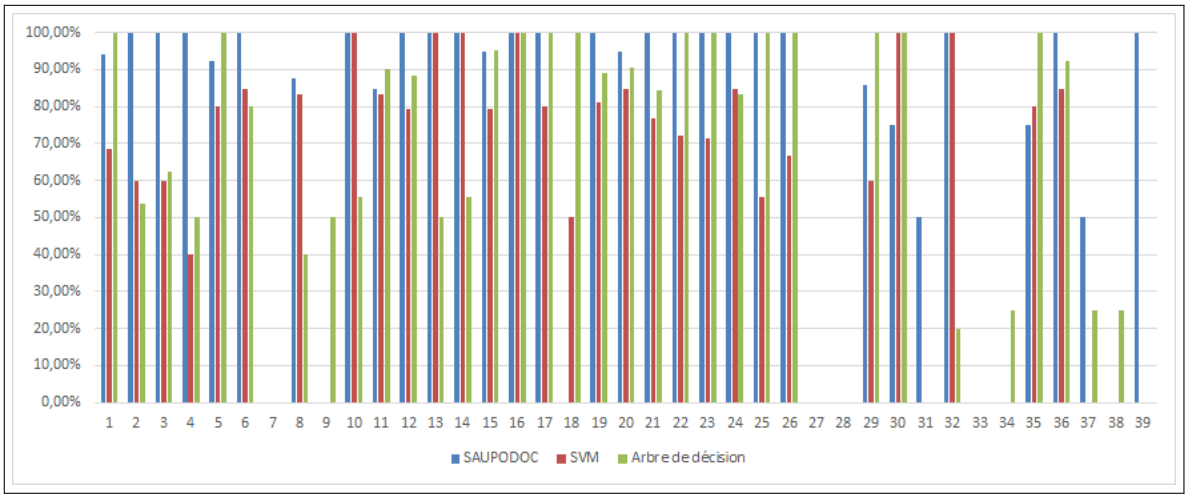


Fig. B.3 Précision des 39 concepts cibles du domaine des destinations

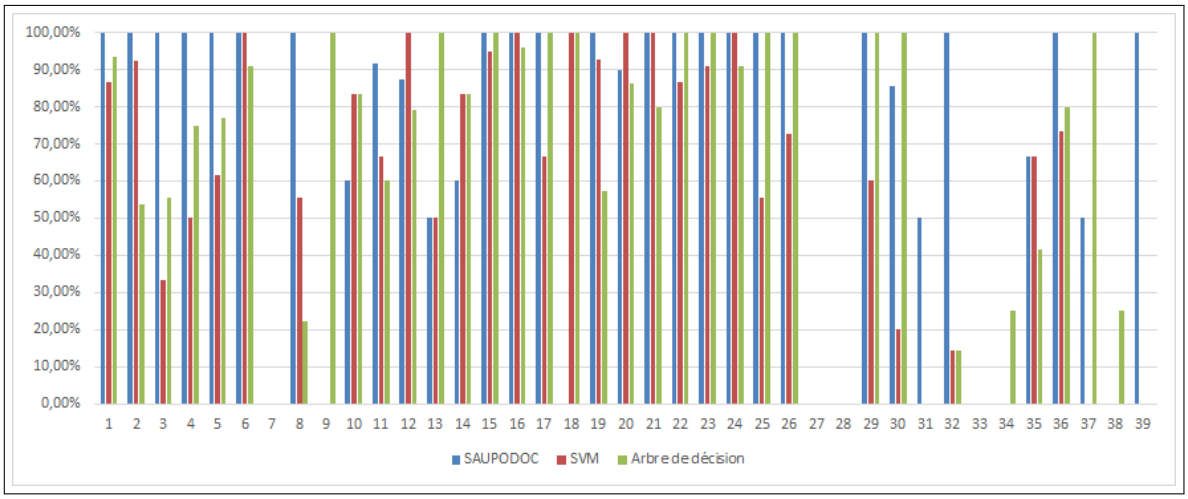


Fig. B.4 Rappel des 39 concepts cibles du domaine des destinations

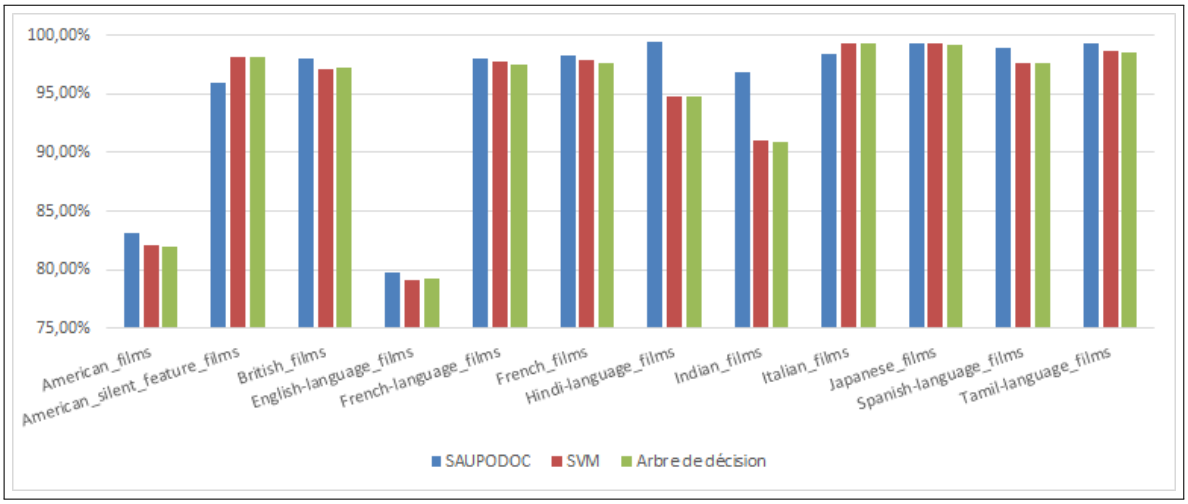


Fig. B.5 Exactitude des 12 concepts cibles du domaine des films

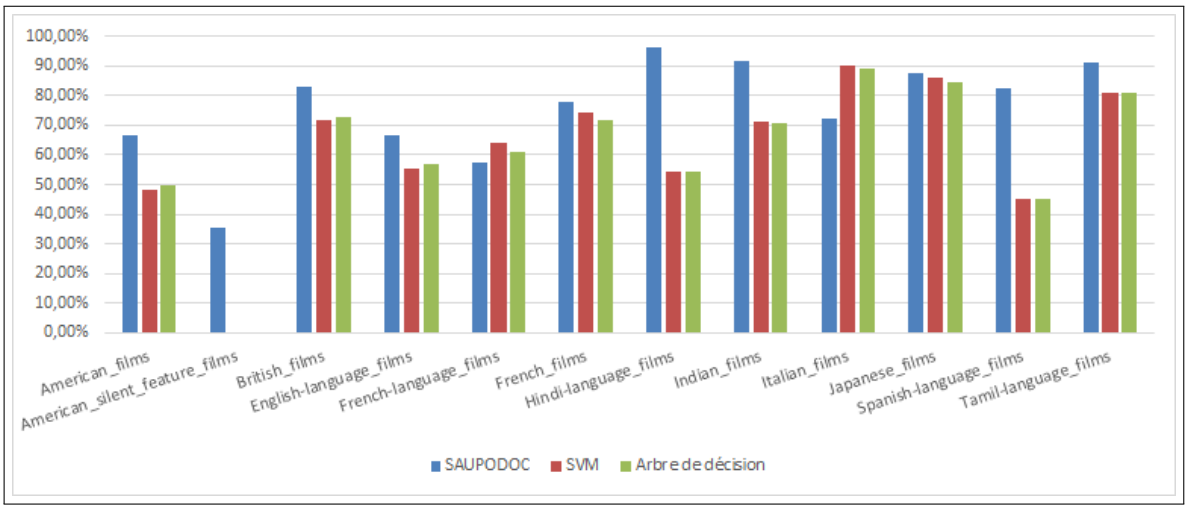


Fig. B.6 F-mesure des 12 concepts cibles du domaine des films

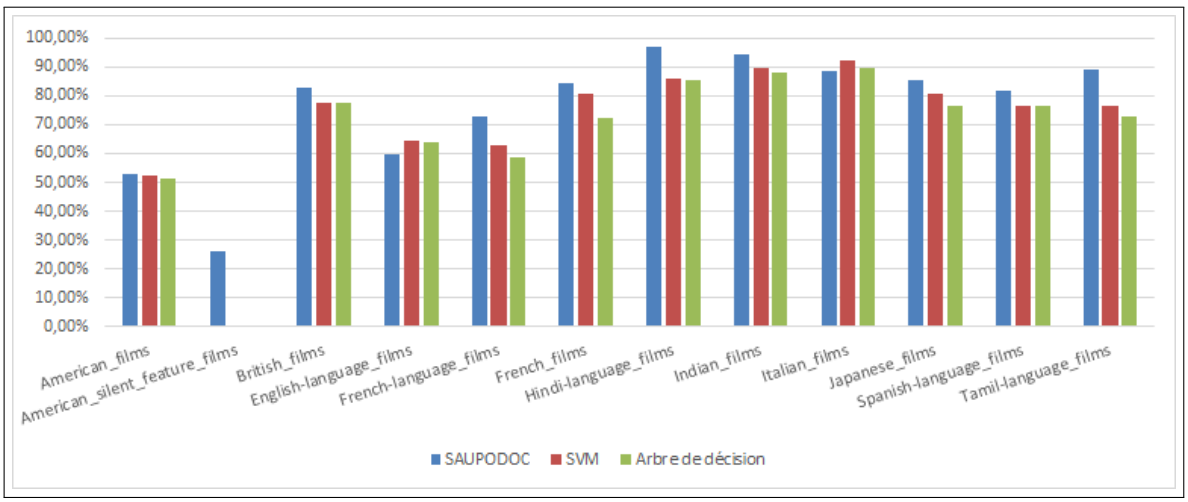


Fig. B.7 Précision des 12 concepts cibles du domaine des films

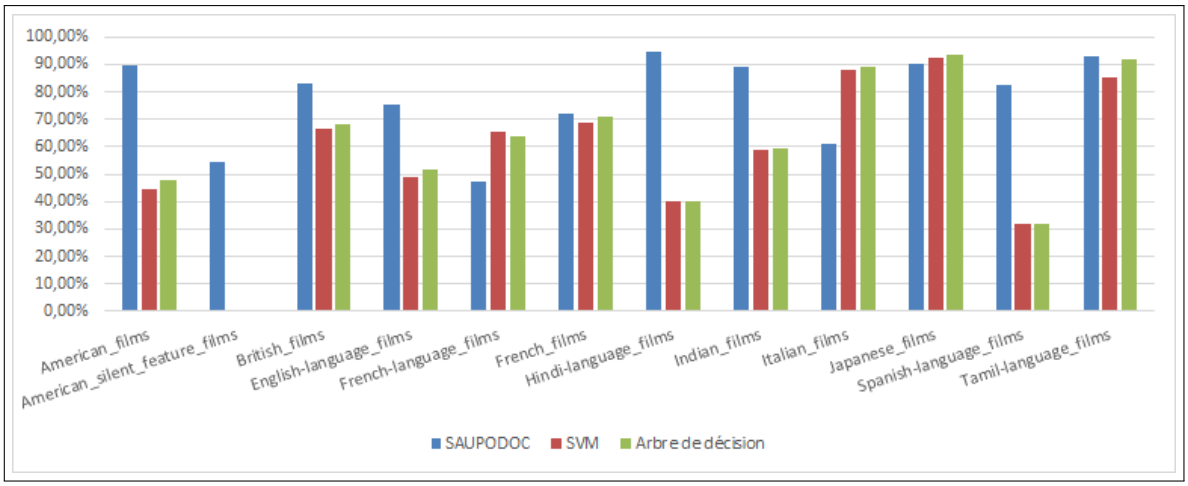


Fig. B.8 Rappel des 12 concepts cibles du domaine des films

B.2 Pertinence moyenne des annotations négatives

L'utilisation de SAUPODOC dans le cadre de la collaboration avec Wepingo est très tournée sur les annotations positives puisque ce sont celles-ci qui vont permettre à Wepingo de proposer des produits. Néanmoins, si nous considérons une vue plus générale du problème d'étiquetage des documents, nous pouvons nous intéresser aussi à la pertinence moyenne des annotations négatives. Pour ce faire, nous nous basons sur la valeur prédictive négative, que nous nommerons "précision des négatifs" ainsi que sur le taux de vrais négatifs connu aussi sous le nom de spécificité, que nous nommerons "rappel des négatifs". Enfin, nous avons aussi calculer la F-mesure des négatifs en fonction de ces deux valeurs.

$$\textit{Précision des négatifs} = \frac{VN}{VN+FN} \qquad \textit{Rappel des négatifs} = \frac{VN}{VN+FP}$$

$$\textit{F-mesure des négatifs} = \frac{2 \times \textit{précision des négatifs} \times \textit{rappel des négatifs}}{\textit{précision des négatifs} + \textit{rappel des négatifs}}$$

Les diagrammes ci-après montrent les résultats moyens obtenus sur ces mesures pour SAUPODOC, SVM et arbre de décision. Ces résultats sont satisfaisants : pour les destinations, SAUPODOC surpasse les classifieurs sur les trois mesures ; pour les films, SAUPODOC surpasse les classifieurs en terme de précision et de F-mesure.

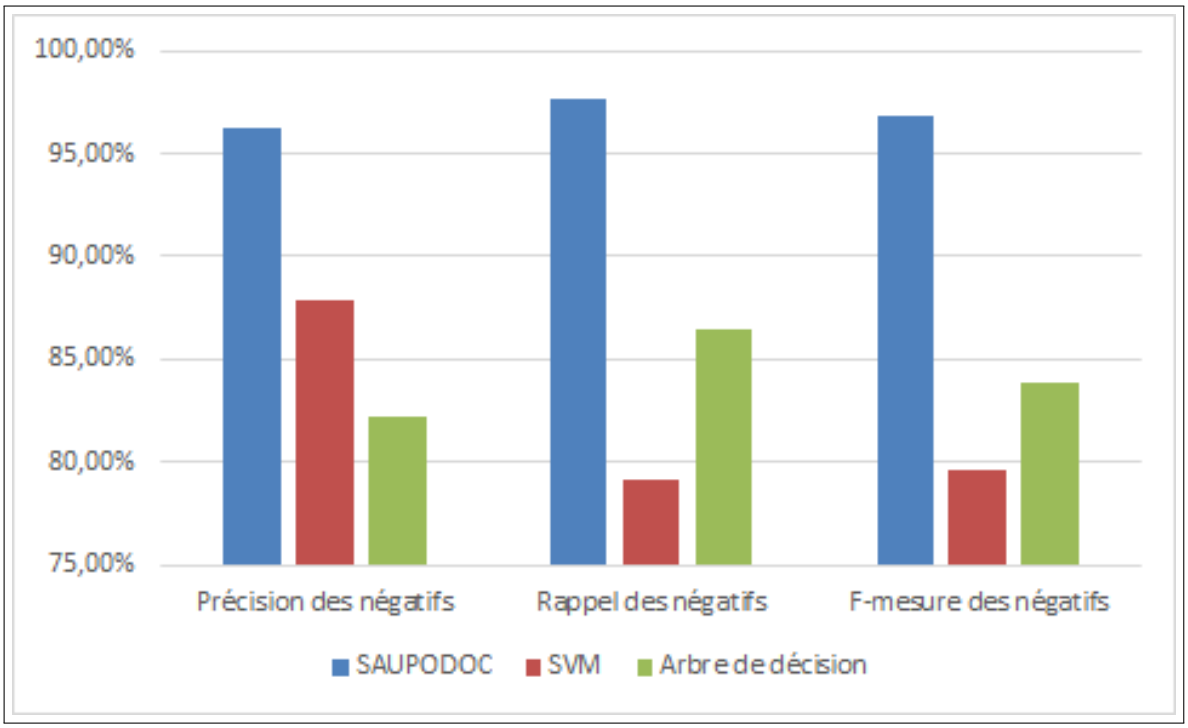


Fig. B.9 Mesures concernant les annotations négatives dans le domaine des destinations

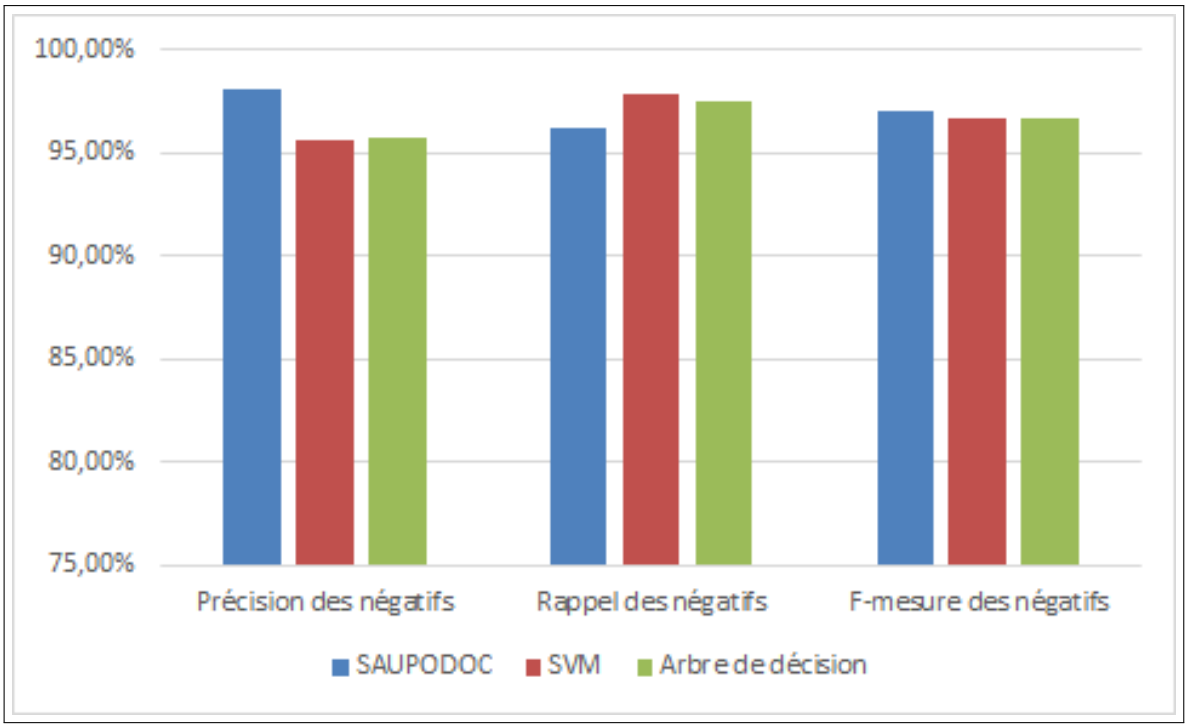


Fig. B.10 Mesures concernant les annotations négatives dans le domaine des films

Annexe C

Expérimentations détaillées sur la tâche de complétion via DBpedia

C.1 Expérimentations de DBpedia Spotlight

Pour effectuer la tâche de complétion des assertions de propriété avec des données de DBpedia (tâche 1.b), nous avons besoin de trouver la page DBpedia relative à chaque document traité. DBpedia Spotlight permet de faire cela.

Nous avons testé DBpedia Spotlight dans le domaine des destinations de voyage. Le Gold Standard a été construit manuellement : il regroupe toutes les pages de DBpedia correspondant aux destinations (au nombre de 80) décrites dans le corpus.

DBpedia Spotlight peut être lancé soit sur une balise spécifique (la balise *name* dans le cadre des destinations de vacances, contenant le nom de la destination, par exemple "Cambodia" dans la Figure C.1), soit sur le document entier. La Figure C.1 montre la différence dans les résultats obtenus, sur un document décrivant le Cambodge. Avec le texte entier, le document est annoté par Cambodian_cuisine. Avec la balise *name*, il est annoté par Cambodia. Les paragraphes suivants montrent que pour le cas des destinations de vacances, les annotations, i.e., les pages retournées, sont plus pertinentes avec seulement la balise *name*.

Fig. C.1 Résultats de DBpedia Spotlight sur un document décrivant le Cambodge en utilisant soit le texte entier (gauche) soit la balise *name* contenant "Cambodia" (droite)

Balise *name*	Page du gold standard	Page trouvée	Différence moyenne pour			
			lat	long	temp	prec
Egypt-Nile	Egypt	null				
Las Vegas	Las_Vegas	Las_Vegas_Valley				
New York	New_York_City	New_York	2,285	0,997	2,551	6,975
Rhodes	Rhodes	null				

Tableau C.1 Les 4 pages erronées données par DBpedia Spotlight en utilisant la balise *name* (lat = latitude, long = longitude, temp = température, prec = précipitation)

Balise *name*	Page du gold standard	Page trouvée	Différence moyenne pour			
			lat	long	temp	prec
Banff	Banff	Banff _National_Park	0,322	0,428	0,279	1,960
Cambodia	Cambodia	Cambodian_Cuisine				
Egypt-Nile	Egypt	null				
Las Vegas	Las_Vegas	Las_Vegas_Strip	0,054	0,036		
New York	New_York_City	New_York _(magazine)				
Rhodes	Rhodes	null				
Rome	Rome	Ancient_Rome			3,856	54,074

Tableau C.2 Les 7 pages erronées en utilisant le document textuel entier

Lorsque DBpedia Spotlight est exécuté uniquement sur la balise *name*, 76 documents sur 80 (95%) sont associés à une page correcte et 4 ne le sont pas (cf. Tableau C.1). Parmi les 4 documents, deux ne sont associés à aucune page. Un document est associé à une page fausse qui ne permet d'obtenir aucune information. Le 4ème document qui décrit la ville de New York est associé à la page sur l'état de New York. Cette page nous permet d'obtenir des données proches de celles recherchées. Pour quantifier l'erreur, nous prenons la valeur absolue de la différence pour la latitude (resp. longitude) entre la valeur sur la page trouvée et celle qui aurait dû être trouvée (gold standard). Pour la température et la précipitation, nous prenons la différence moyenne par mois. Par exemple, nous trouvons que la température mensuelle à New-York est 2.55°C plus basse ou haute comparée à la température du Gold standard de New-York.

Quand on exécute DBpedia Spotlight sur tout le document (et non plus uniquement sur la balise *name*), 73 pages sont correctes (91,25%) et 7 pages sont fausses (cf. Tableau C.2). Les pages non trouvées précédemment ne le sont pas non plus, mais pas toujours pour la même raison. Il y a aussi davantage d'erreurs. Nous en déduisons qu'il est préférable de faire exécuter DBpedia Spotlight uniquement sur la balise *name*. Le seul meilleur résultat est celui de Las Vegas qui a presque la bonne latitude et longitude.

C.2 Évaluation du processus d'extraction des assertions de propriétés du LOD

Nous avons testé notre approche basée sur les chemins d'accès alternatifs en cas de valeurs manquantes dans DBpedia. Pour rappel, une liste ordonnée de chemins (partie switch) est donnée pour chaque correspondance (PE_s,PE_t). Dans cette annexe, nous reprenons l'exemple de partie switch donné page 106 :

```
1 case 1 info ← i_t.PE_t.getVal(); break;                          /* accès direct */
2 case 2 info ← i_t.capital.PE_t.getVal(); break;                      /* ordre 1 */
3 case 3 info ← i_t.largestCity.PE_t.getVal(); break;                  /* ordre 1 */
4 case 4 info ← i_t.subparts.PE_t.getVal(); break;                     /* ordre 1 */
5 case 5 info ← i_t.country^-1.PE_t.getVal(); break;                   /* ordre 1 */
6 case 6 info ← i_t.?prop.PE_t.getVal() ∪ i_t.?prop^-.PE_t.getVal(); break;
                                                                       /* ordre 1 */
7 case 7 info ← i_t.country.capital.PE_t.getVal(); break;              /* ordre 2 */
8 case 8 info ← i_t.?prop^-1.subparts.PE_t.getVal(); break;            /* ordre 2 */
```

La partie switch pour la géolocalisation (latitude et longitude) est plus simple que la partie switch relative aux données météorologiques données précédemment. Deux propriétés de DBpedia correspondent à la latitude (resp. longitude). Si

les valeurs de ces deux propriétés sont manquantes, on peut obtenir leurs valeurs via la capitale de la destination. La partie switch se limite donc aux cas 1 et 2 de l'exemple. Les données de géolocalisation se trouvent directement sur les pages en accès direct pour 79 destinations sur 80, et sur la page de la capitale pour 1 cas. Pour les données météorologiques, le mode d'accès est résumé dans le Tableau C.3.

Numéro du cas dans la partie switch dans userAlgo	1	2	3	4	5	6	7	8
Nombre de pages trouvées contenant des données météorologiques	29	26	2	5	5	7	4	2

Tableau C.3 Données météorologiques trouvées par l'algorithme sur les 80 destinations

L'algorithme est assez intuitif sauf lorsqu'il s'agit de pages liées (cas 6 et 8). Pour être sûr de la pertinence, nous avons analysé les données des 7 destinations trouvées via ces pages liées (cf. Tableau C.4 pour le cas 6) et celles trouvées via des sous-parties de pages liées (cas 8 cf. Tableau C.5). Dans chaque tableau, chaque page retournée représente un endroit proche de la destination initiale. Il n'y a qu'une seule exception : London au Kentucky est retourné à la place de London en Angleterre. Les données météo pour Londres sont des moyennes calculées à partir des valeurs provenant des 8 pages données dans le tableau, cette erreur n'a donc que peu d'impact. Les données moyennes obtenues ne diffèrent en fait que de 1.12°C pour la température et de 5.25 mm pour les précipitations par rapport aux données de la City à Londres en Angleterre.

Destinations	Ressources (pages liées)
Boston	Quincy,_Massachusetts
Crete	Chania, Heraklion
Fuerteventura	Canary_Islands
Lanzarote	Canary_Islands, Gran_Canaria
London	Wellingborough, City_of_London, Kettering, London,_Kentucky, Milton_Keynes, Bromley, England, Malvern,_Worcestershire
Miami	Miami_Beach,_Florida
Santorini	Oia,_Greece

Tableau C.4 Ressources avec données météorologiques extraites en utilisant le cas 6

Notons aussi que notre algorithme permet de résoudre des problèmes de transtypage auxquels nous sommes confrontés lorsque des données n'ont pas d'unité. Cela arrive par exemple quand des valeurs de propriétés sont des nombres négatifs. Le changement d'unité au moment où on pose la requête SPARQL n'est pas possible. L'avantage de notre algorithme est qu'il va rechercher une autre valeur de propriété. Par exemple, dans le cas de Prague, Figure C.2, deux valeurs, -5,4 et -2,0 n'ont pas d'unités. L'extraction se fera alors à partir d'une page liée, la page de la République Tchèque, ce qui est très bien.

Destinations	Pages liées	Ressources (sous-parties de pages liées)
Dubai	United_Arab_Emirates, Abu_Dhabi_(emirate)	Abu_Dhabi
	Sharjah_(emirate)	Sharjah_(city)
Caribbean	Grand_Cayman	George_Town,_Cayman_Islands
	Grand_Turk_Island	Cockburn_Town

Tableau C.5 Ressources avec données météorologiques extraites en utilisant le cas 8

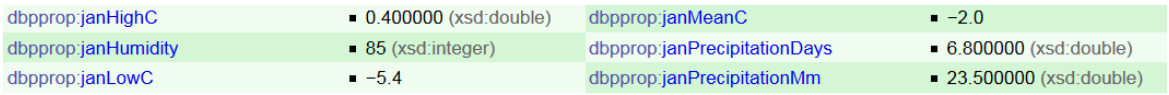


Fig. C.2 Quelques données dans la page décrivant Prague

Annexe D

Avoir des définitions explicites : un moyen pour détecter les erreurs humaines

Dans le cadre de l'approche SAUPODOC, nous générons des définitions explicites, compréhensibles par un humain. Ainsi, d'éventuelles erreurs données en entrée sont plus facilement détectables. Le concepteur peut juger une définition incohérente et ainsi mener une analyse plus poussée pour comprendre comment a été apprise cette définition. Avec un classifieur, le concepteur peut passer à côté d'un tel problème.

Par exemple, nous avons cherché à introduire un nouveau concept cible concernant les films : *Films_based_on_novels*, pour lequel nous obtenons une définition qui ne semble pas avoir de sens : (isFromCountry some Country) and (runtimeInSeconds some double[>= "4590.0"]) and (runtimeInSeconds some double[<= "9750.0"]). De plus, l'exactitude moyenne de cette définition (sur l'ensemble de test) n'est que de 51,74%.

Une analyse des données correspondantes a été menée. Celle-ci a montré que la plupart du temps, les descriptions des films mentionnent que le film est basé sur une nouvelle tandis que nos annotations données en entrée sont négatives. Rappelons que les annotations ont été faites automatiquement dans cette expérimentation en annotant positivement les films dont la page est liée à la catégorie *Films_based_on_novels* par la propriété dcterms:subject. En inspectant DBpedia, nous nous sommes rendues compte que les films considérés sont en fait associés à une catégorie plus spécifique, par exemple Films_based_on_thriller_novels. Or, dans DBpedia, Films_based_on_novels ne subsume pas ce type de catégorie. En conséquence, quand nous avons annoté notre corpus automatiquement avec le concept cible Films_based_on_novels en utilisant DBpedia, tous les films liés à une catégorie plus spécifique ont été annotés comme exemples négatifs.

Le problème se situait donc dans l'annotation fournie en entrée de l'approche. Avec SAUPODOC, nous avons pu déceler la présence d'un problème dans ce cas précis

tandis que cela aurait été beaucoup plus difficile à déceler avec les classifieurs car leur définitions ne sont pas explicites et l'exactitude obtenue pour ce concept cible était de 97,66% (pour les deux classifieurs).

Dans le cadre d'une approche comme la nôtre, qui nécessite en entrée des annotations manuelles potentiellement sujettes à erreurs, il est important de pouvoir détecter facilement la présence d'erreurs quand elles sont en grand nombre, afin de les corriger au plus vite. L'exemple montré dans cette annexe prouve que ces erreurs sont plus facilement détectables via notre approche qu'avec des classifieurs.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Enrichissement et peuplement d'ontologie à partir de textes et de données du LOD : Application à l'annotation automatique de documents

Mots clefs : Annotation sémantique, Peuplement d'ontologie, Annotations orientées application

Résumé : Cette thèse traite d'une approche, guidée par une ontologie, conçue pour annoter les documents d'un corpus où chaque document décrit une entité de même type. Dans notre contexte, l'ensemble des documents doit être annoté avec des concepts qui sont en général trop spécifiques pour être explicitement mentionnés dans les textes. De plus, les concepts d'annotation ne sont représentés au départ que par leur nom, sans qu'aucune information sémantique ne leur soit reliée. Enfin, les caractéristiques des entités décrites dans les documents sont incomplètes. Pour accomplir ce processus particulier d'annotation de documents, nous proposons une approche nommée SAUPODOC (Semantic Annotation Using Population of Ontology and Definitions of Concepts) qui combine plusieurs tâches pour (1) peupler et (2) enrichir une ontologie de domaine. La phase de peuplement (1) ajoute dans l'ontologie des informations provenant des documents du corpus mais aussi du Web des données (Linked Open Data ou LOD). Le LOD représente aujourd'hui une source prometteuse pour de très nombreuses applications du Web sémantique à condition toutefois de développer des techniques adaptées d'acquisition de données. Dans le cadre de SAUPODOC, le peuplement de l'ontologie doit tenir compte de la diversité des données présentes dans le LOD : propriétés multiples, équivalentes, multi-valuées ou absentes. Les correspondances à établir, entre le vocabulaire de l'ontologie à peupler et celui du LOD, étant complexes, nous proposons un modèle pour faciliter leur spécification. Puis, nous montrons comment ce modèle est utilisé pour générer automatiquement des requêtes SPARQL et ainsi faciliter l'interrogation du LOD et le peuplement de l'ontologie. Celle-ci, une fois peuplée, est ensuite enrichie (2) avec les concepts d'annotation et leurs définitions qui sont apprises grâce à des exemples de documents annotés. Un raisonnement sur ces définitions permet enfin d'obtenir les annotations souhaitées. Des expérimentations ont été menées dans deux domaines d'application, et les résultats, comparés aux annotations obtenues avec des classifieurs, montrent l'intérêt de l'approche.

Title : Ontology enrichment and population from texts and data from LOD: Application to the automatic annotation of documents

Keywords : Semantic annotation, Ontology population, Application-driven annotations

Abstract : This thesis deals with an approach, guided by an ontology, designed to annotate documents from a corpus where each document describes an entity of the same type. In our context, all documents have to be annotated with concepts that are usually too specific to be explicitly mentioned in the texts. In addition, the annotation concepts are represented initially only by their name, without any semantic information connected to them. Finally, the characteristics of the entities described in the documents are incomplete. To accomplish this particular process of annotation of documents, we propose an approach called SAUPODOC (Semantic Annotation of Population Using Ontology and Definitions of Concepts) which combines several tasks to (1) populate and (2) enrich a domain ontology. The population step (1) adds to the ontology information from the documents in the corpus but also from the Web of Data (Linked Open Data or LOD). The LOD represents today a promising source for many applications of the Semantic Web, provided that appropriate techniques of data acquisition are developed. In the settings of SAUPODOC, the ontology population has to take into account the diversity of the data in the LOD: multiple, equivalent, multi-valued or absent properties. The correspondences to be established, between the vocabulary of the ontology to be populated and that of the LOD, are complex, thus we propose a model to facilitate their specification. Then, we show how this model is used to automatically generate SPARQL queries and facilitate the interrogation of the LOD and the population of the ontology. The latter, once populated, is then enriched (2) with the annotation concepts and definitions that are learned through examples of annotated documents. Reasoning on these definitions finally provides the desired annotations. Experiments have been conducted in two areas of application, and the results, compared with the annotations obtained with classifiers, show the interest of the approach.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France