

Contribution

Ouverture

Les articles de Wikipédia sont constitués généralement d'un texte, mais aussi de certaines informations structurées (infobox, images, liens externes, redirections entre les pages etc...) présentes sous la forme de balises Wiki. Le projet DBpedia extrait des informations structurées de Wikipédia et les transforme dans une base de connaissances riche sous forme d'un graphe avec des entités reliées.

Dans ce chapitre, nous allons vous donner une vue d'ensemble sur l'annotation des métadonnées, une analyse des besoins et la procédure d'extraction de DBpedia et les pistes de travail possibles. Puis nous présentons l'architecture globale de notre système ainsi que notre hypothèse et la structure de l'application que nous avons développée pour concrétiser notre hypothèse. La dernière partie de cette section porte sur les résultats de notre étude et la validation de notre hypothèse.

Utilité des annotations

En générale, l'annotation, c'est une étiquette qu'on ajoute à une ressource Web. Depuis la création du Web, plusieurs systèmes d'annotation sont apparus (ThirdVoice, PageSeeder, HyperNews, Nestor, etc...). Nous citons brièvement les conséquences liées à ces systèmes d'annotation telles que l'information annotée doit d'une manière ou d'une autre être structurée "utilisable" et descriptive de la ressource ou de son utilisation. De plus, la ressource en question doit exister et peut être exploitée sur le Web indépendamment des informations qui lui sont associées. La figure ci-dessous montre le système intermédiaire entre le client et le service Web permettant la communication entre ces deux entités.

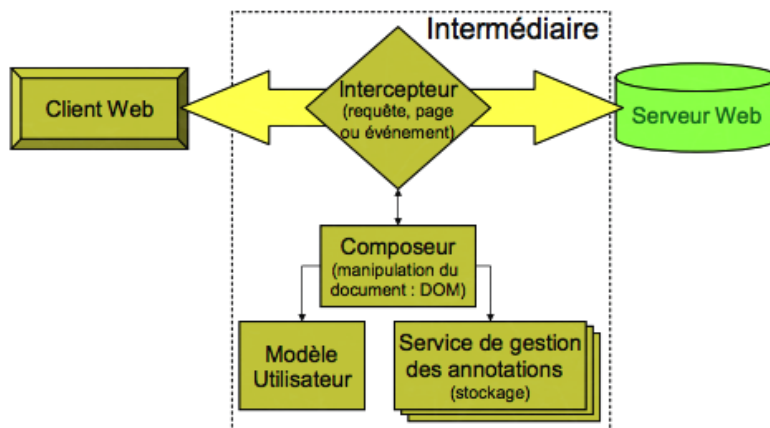


FIGURE 1 – Notation d'intermédiaire

L'annotation sémantique fait référence à plusieurs types distincts d'annotations formelles, explicites et permanentes. Il existe des outils d'annotation basés sur les ontologies *Ontology based annotation tool* et des critères relatifs aux annotations par exemple : les types de ressources concernées, la structuration des schémas de description, l'automatisation marquée de la mise en place, etc.

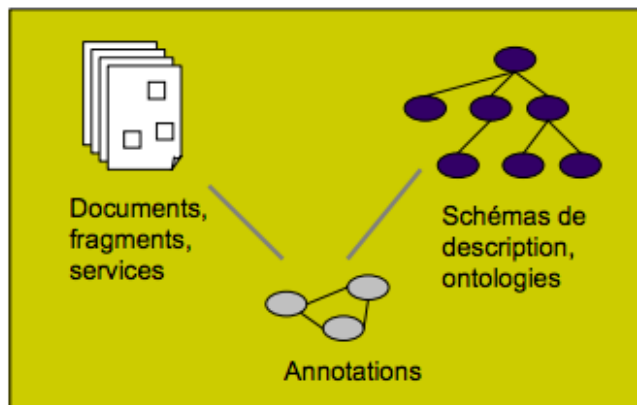


FIGURE 2 – Différents niveaux de connaissances

L'annotation d'un triplet RDF est une façon d'ajouter des metadonnées à un triplet RDF pour décrire une restriction spatiale.

Comment on utilise les annotations temporelles ? Un exemple d'utilisation est sur le site “sig.ma”¹ créée par *the digital enterprise research institute in Ireland*, la plateforme fournit un moteur de recherche par mot clé qui permet de récupérer des images et des textes accessibles par des annotations RDF, ainsi qu’une liste d’URI synonymes correspondant à la clé de recherche et des liens vers des sources Web contenant des données RDF pertinentes.

Analyse des Besoins

Le large succès de Wikipedia (qui est le 2ème site le plus visité sur internet) et le progrès des techniques d’extraction des données ont abouti à la naissance de la construction automatique de larges bases de connaissances comme DBpedia, YAGO, etc...

The image shows a composite of information about Barack Obama from DBpedia. On the left, there's a snippet from the Wikipedia article in French, detailing his birth (1961), education (Columbia University), and political career. Below this is a Wikidata entry for Barack Obama, listing various identifiers and properties. On the right, a portrait of Barack Obama is shown above a structured summary of his roles and dates.

| Fonctions | |
|--|-----------------------------|
| 35^e président des États-Unis | |
| 20 janvier 1961 – 22 novembre 1963 | (2 ans, 10 mois et 2 jours) |
| Élection | 8 novembre 1960 |
| Vice-président | Lyndon B. Johnson |
| Prédécesseur | Dwight D. Eisenhower |
| Successeur | Lyndon B. Johnson |
| Sénateur du Massachusetts | |
| 3 janvier 1953 – 22 décembre 1960 | |
| Prédécesseur | Henry Cabot Lodge, Jr. |
| Successeur | Benjamin A. Smith II |

FIGURE 3 – Différentes sources d’informations dans DBpedia

Beaucoup de connaissances sont construites en se basant sur l’extraction automatique des faits relationnels dans un texte. Malheureusement, les bases de connaissances convergent sur les faits statiques et ne donnent pas une grande importance à la dimension temporelle de ces faits. Et ceci a lieu en dépit du fait que la majorité des faits évoluent avec le temps, ou n’est

1. <http://sig.ma/>

valide que dans une période temporelle précise. Ainsi, nous remarquons que le temps a une dimension significative dans ces bases de connaissances.

La dimension temporelle est particulièrement importante dans les relations binaires comme *isPresidentOf*, *isCEOof*, *isMarriedTo*, on peut être mariée à plusieurs épouses mais dans des différents intervalles de temps (On ne prend pas compte des exceptions que représentent les mariages polygames). Une base de connaissances contenant plusieurs présidents des États-Unis ne peut être consistante que lorsqu'on ajoute une dimension temporelle à ces faits. De plus l'annotation temporelle aide à faire la distinction entre les faits courants et les faits dépassés. Par exemple le fait "Kennedy est le président des États-Unis" est correct, mais n'est plus valide. Lorsqu'on attache une annotation temporelle à un fait comme celui là, il devient universellement valide.

Problématique

Lorsqu'on parcourt DBpedia, on trouve beaucoup de triplets qui décrivent des informations temporelles. Ces derniers sont généralement liés à un contexte événementiel précis. Il est plus difficile d'exploiter ces informations si elles ne possèdent pas une structure universellement valide, claire et lisible par la machine. Dans DBpedia, il se trouve que des informations liées au même contexte temporel sont exprimées de la manière suivante :

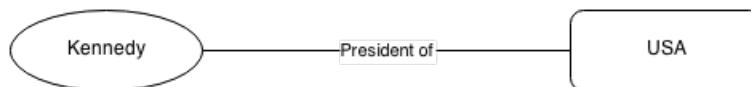


FIGURE 4 – triplet "Kennedy"

Le premier triplet n'a pas une sémantique valide que en tenant compte du triplet suivant :

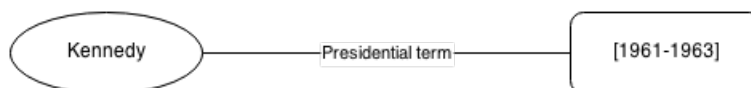


FIGURE 5 – triplet presidential term "Kennedy"

Dans cette étude, on vise plutôt à annoter les triplets (s, p, o) avec une étiquette temporelle qui indique et précise la validité de ce terme dans un cadre logique qui appartient au monde réel où en dehors de ce cadre, on peut dire que ce triplet RDF n'est pas valide et qu'on ne peut pas l'utiliser.

Étude préliminaire et approches possibles

Web Collaboratif

C'est le Web qui s'appuie sur les utilisateurs pour construire son contenu. Nous avons commencé notre travail de recherche par une étude préliminaire autour du contenu de ces plateformes collaborative ; nous avons étudié les pistes possibles pour l'exploitation des dumps de Wikipédia et Wikidata. Tout d'abord, nous avons téléchargé les fichiers des collections XML et nous avons observé la structure des informations dans ces sources d'informations. Ensuite nous avons implémenté un premier algorithme d'extraction en utilisant un parseur XML (SAX²). La figure ci-dessous représente notre schéma de modélisation dans laquelle nous avons procédé avec une modélisation qui touche directement la source principale d'informations Wikipédia.

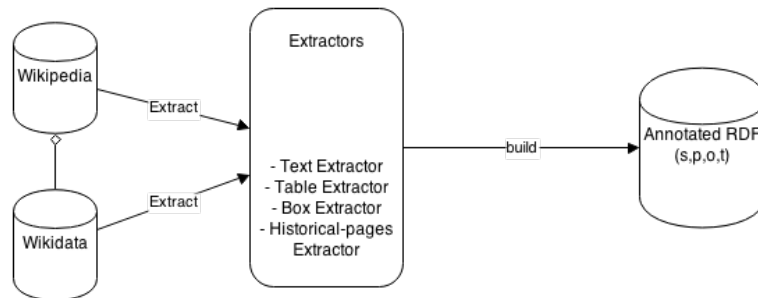


FIGURE 6 – Première approche : schéma de modélisation générale

Cette modélisation est une première rubrique d'analyse et de conception d'une solution qui touche les besoins préliminaires de notre étude. Par ailleurs, nous avons retrouvé une autre modélisation plus proche à nos besoins principaux et que nous vous détaillerons par la suite.

2. http://fr.wikipedia.org/wiki/Simple_API_for_XML

Traitement automatique des langues

Le traitement automatique de la langue (TAL) est une discipline à la frontière linguistique liée à l'intelligence artificielle. Il existait un type de TAL statistique proposant des méthodes statistiques, probabilistes ou purement statistiques pour résoudre certaines difficultés. On distingue plusieurs domaines d'application de TAL comme la traduction automatique, la génération automatique de texte, la correction orthographique, la reconnaissance de l'écriture manuscrite, etc... Mais dans ce type de traitement, il y a des problèmes qui peuvent apparaître et principalement celui de l'ambiguïté temporelle.

Ambiguïtés temporelles

C'est la propriété d'un mot ou d'une suite de mots (comme notre cas) qui peuvent avoir un ou plusieurs sens d'analyses grammaticales possibles. Dans une phrase simple ou composée, l'indicateur temporel peut avoir plusieurs sens tout dépend du contexte de la phrase.

Les informations temporelles peuvent avoir des représentations différentes :

- Un événement “Je vous propose un rendez-vous *demain* pour parler de ma plateforme PiSharing”.
- Une connaissance “Jacques Chirac est le président de la république Française” **mais quand ?**.

Le présent par exemple peut avoir plusieurs sens ou contextes : présent de narration, présent de généralité, présent qui réfère au futur proche, etc... Les signaux temporels sont ambigus par exemple dans ces expressions : il court pour rattraper le temps, tu tournes après la rivière, etc... On remarque qu'il y a des indicateurs temporels, mais ce n'est pas le temps qui est relatif à un événement qui peut nous intéresser. La plupart des expressions sont floues comme : il y a deux ans, chaque deux semaines, j'arrive dans deux secondes, etc... Certes, il n'y a pas une logique descriptive qui peuvent nous aider à mettre un lien entre l'événement et la période temporelle.

L'analyse du temps s'inscrit dans la compréhension globale des textes, et des événements auxquels on fait référence dans ce texte non pas en analysant une phrase comme suit.

Modalité : “l’équipe de France voulait gagner la coupe du monde en 2006.”
Anaphore : “..., cela pourrait avoir lieu dans les éditions suivantes.”

Les évènements décrits (et que l’on souhaite fixer temporellement) peuvent être : duratifs ou ponctuels/accomplis ou inaccomplis. De même pour les dates qui peuvent être des dates absolues “le 18 mars, c’est mon anniversaire” ; ou bien des dates relatives par rapport au moment de l’énonciation par exemple : “il y a deux ans”. Pour la durée aussi on distingue plusieurs types comme la durée absolue “durant 2 ans” et la durée relative “depuis un an”. Dans un texte, on trouve aussi un ensemble d’expressions de fréquence par exemple “tous les ans, le vendredi 13” et des expressions plus complexes comme “après la Révolution Tunisienne”.

Les textes contiennent des informations temporelles de taille massive qui sont difficilement exploitables. Nous avons essayé de donner une vue globale sur cette procédure que nous avons décidé de ne pas l’adopter parce que notre objectif est d’annoter des triplets RDF plutôt que de faire l’analyse des textes. Nous détaillerons par la suite l’architecture de DBpedia à partir de laquelle on s’inspire pour proposer notre solution.

Historique des modifications dans Wikipédia

L’historique est une page attachée à un article encyclopédique pour conserver le journal de la plupart des modifications qui ont été apportées à cet article. Cette page permet de connaître la date, l’auteur et la teneur externe de chaque modification. Dans cette encyclopédie, nous avons remarqué qu’à partir de l’historique de modifications, on peut déduire plusieurs informations liées à deux ou plusieurs contextes temporels différents. Nous souhaitons, si c’est possible, extraire ces informations temporelles et les rendre exploitables dans DBpedia.

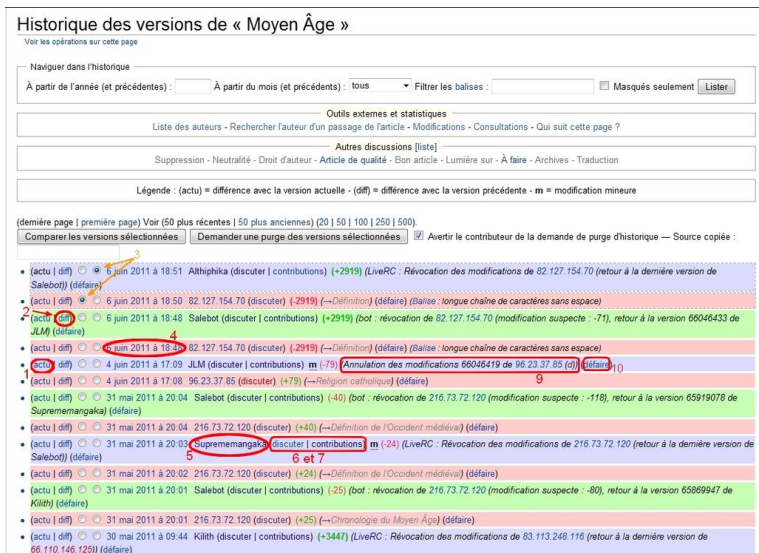


FIGURE 7 – Historique d’articles Wikipédia

Architecture d'extraction de DBpedia

Vue d'ensemble

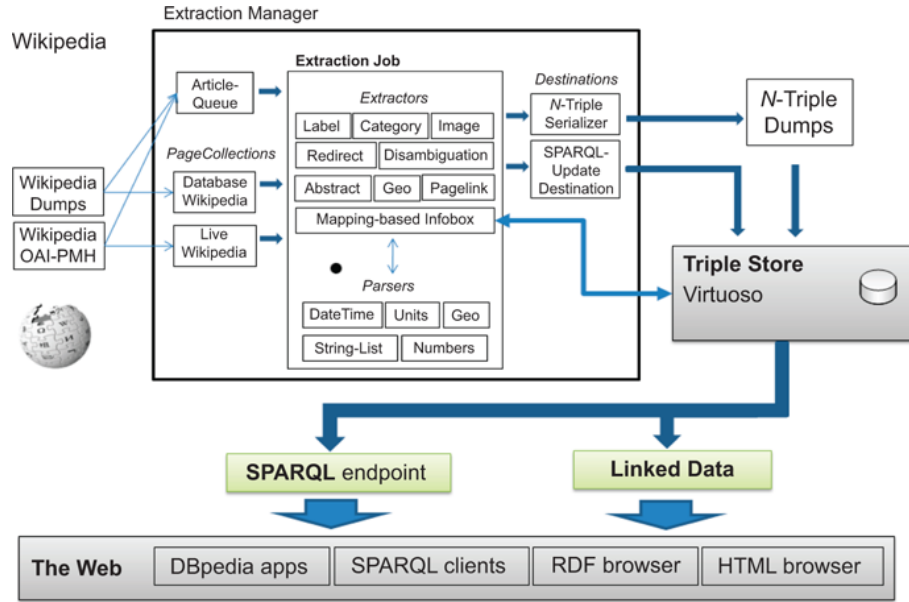


FIGURE 8 – Extracteur DBpedia

La figure ci-dessus montre l'architecture du système d'extraction des connaissances dans DBpedia. D'après Morsey et al [?] les principaux éléments du système sont les suivants : *PageCollections* est une abstraction des ressources locales ou distantes des articles de Wikipédia, *Collections* stockent ou sérialisent les triplets RDF extraites, *Extractors* qui transforme un type spécifique de la syntaxe wiki en triplet, *Parsers* soutiennent les *Extractors* en déterminant les types de données, convertit les valeurs entre différentes unités et fractionne les marqueurs dans des listes. L'*Extraction Job* regroupe une collection de pages, extracteurs et destination dans le flux de travail *workflow*. Le noyau de ce système est l'*Extraction Manager* qui gère le processus d'adoption des articles de Wikipédia sur les *Extractors* et donne les résultats à la destination. Le gestionnaire d'extraction *Extraction Manager* gère également la gestion des URI et résout les redirections entre les articles : ce système se compose de 11 extracteurs qui traitent les types des contenus de Wikipédia (*Labels*, *Abstracts*, *Interlanguage links*, *Images*, *Redirects*, *Disambiguation*, *External Links*, *Pagelinks*, *Homepages*, *Categories*, *Geo*—

coordinates). Ce framework d'extraction DBpedia est mise en place pour réaliser deux flux : extraction à partir des sources de données (*DataBaseWikipedia page collections*) et une procédure d'extraction directe (*LiveWikipedia page collections with the OAI – PMH protocol*) pour obtenir la version courante des articles.

Notre proposition

En analysant les *dumps* DBpedia et en observant l'architecture de cette base de connaissance, nous avons remarqué que pour annoter temporellement les triplets RDF de DBpedia il est plus intéressant d'extraire l'ensemble des propriétés dans DBpedia, puis de trouver des faits qui ont un trait avec le temps, donner une liste de couples à partir de laquelle un expert choisie un couple et valide les résultats de notre algorithme. Par la suite, nous allons présenter en détaille notre proposition.

Modélisation

Le modèle quaternaire est un modèle qui capte la base du fait avec un indice temporel, l'exemple suivant montre le principe de ce modèle.

<politician> elected <president of US> on <date>

f1, Kennedy elected PresidentOfUSA

f2 :f1, HappenedDate

<politician> served as <politician office> from <date> to <date>

f1, Kennedy holdsPoliticalPosition PresidentOfUSA

f2 :f1, startedOnDate

f3 :f1, endedOnDate

HappenedDate est utilisée pour dire que le fait est valide que dans ce point du temps.

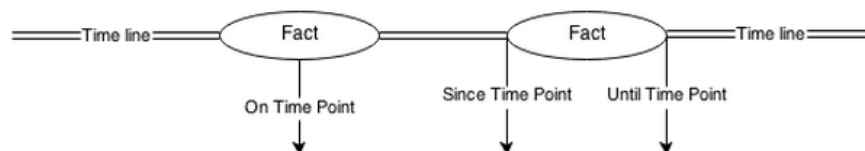


FIGURE 9 – Chronologie des événements

Ce modèle est capable d'exprimer la validité temporelle d'un triplet RDF d'une manière à la fois intelligente et lisible par la machine ; on souhaite rattacher au triplet valide que dans un point du temps ou une plage temporelle bien précise une étiquette temporelle adéquate comme le montre la figure suivante.

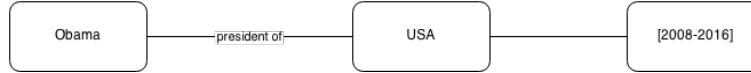


FIGURE 10 – Modélisation quadruplet

On s'intéresse particulièrement au format N-Quads comme format de sortie de notre algorithme. Les quadruplets vont être formalisés de la manière suivante :

$\langle s, p, o, t \rangle$: un sujet, prédicat, objet avec un point de temps.

$\langle s, p, o, [t1, t2] \rangle$: de même avec une intervalle de temps.

Notre hypothèse

Après une observation approfondie dans les sources de données dans DBpédia, nous avons repéré des relations logiques entre des propriétés comme (*beatifiedBy*, *beatifiedDate*). Nous avons trouvé plusieurs propriétés qui ont un lien logique entre eux, les relations temporelles ont comme objet un point du temps particulier et partagent le même sujet ou la même ressource avec une autre propriété. Durant cette étude nous avons essayé de valider cette hypothèse :

```

if (x propTemp t) and (x propWithToken z) then
    (x propWithToken z) t

```

* *propTemp* est une propriété DBpédia contenant un indice temporel (Year, Date).

propWithToken est une propriété DBpédia avec un motif rattacher.

t est l'annotation temporelle du triplet (*x,propWithToken,z*).

Nous avons présenté cette hypothèse sous forme d'une requête SPARQL. Cette requête interroge l'ensemble des ressources sur DBpédia et retourne des résultats si c'est possible. Notre hypothèse porte principalement sur le fait

d'annoté temporellement les ressources de DBpedia en utilisant en essayant de repérer deux triplets portant sur un même sujet et permettant à les relier dont le but est d'avoir un quadruplet valide. La liste des couples (*PropTemp*, *PropWithToken*) est donnée comme sortie d'une procédure d'extraction intelligente de l'ensemble des propriétés de DBpédia.

Notre choix

Architecture du système

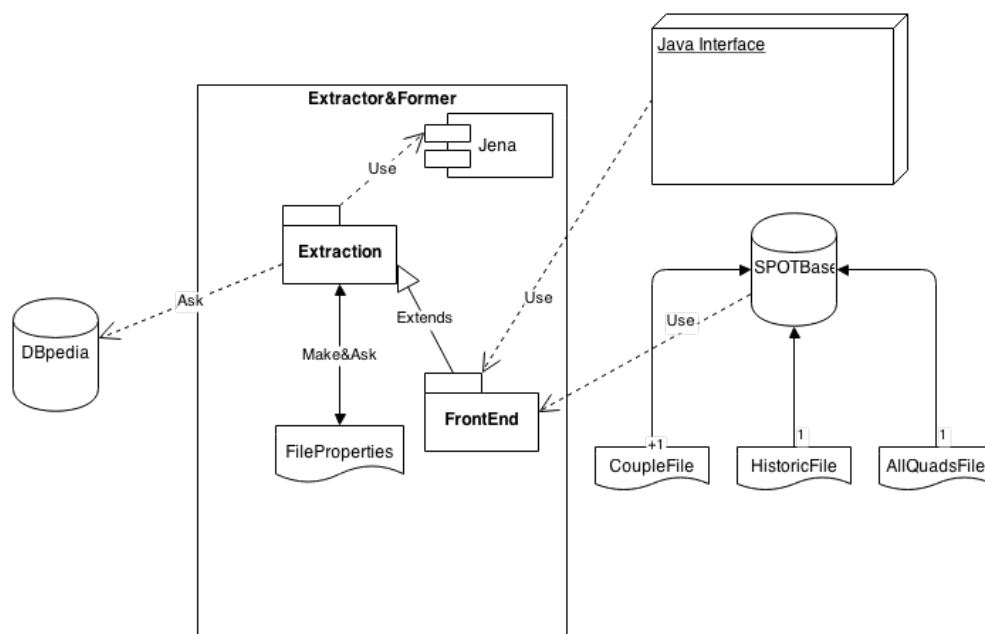


FIGURE 11 – Architecture de l'application

L'architecture de notre application se repose principalement sur celle de DBpédia. En premier lieu, nous interrogeons DBpédia pour avoir une liste de propriétés. En effet, nous pouvons prendre la liste de toutes les propriétés en interrogeant *Virtuoso SPARQL Query Editor*³ avec la requête

```
select distinct ?P where {?S ?P ?O}
```

mais on se limite aux propriétés de DBpedia dans qui ont la forme suivante

3. <http://dbpedia.org/sparql>

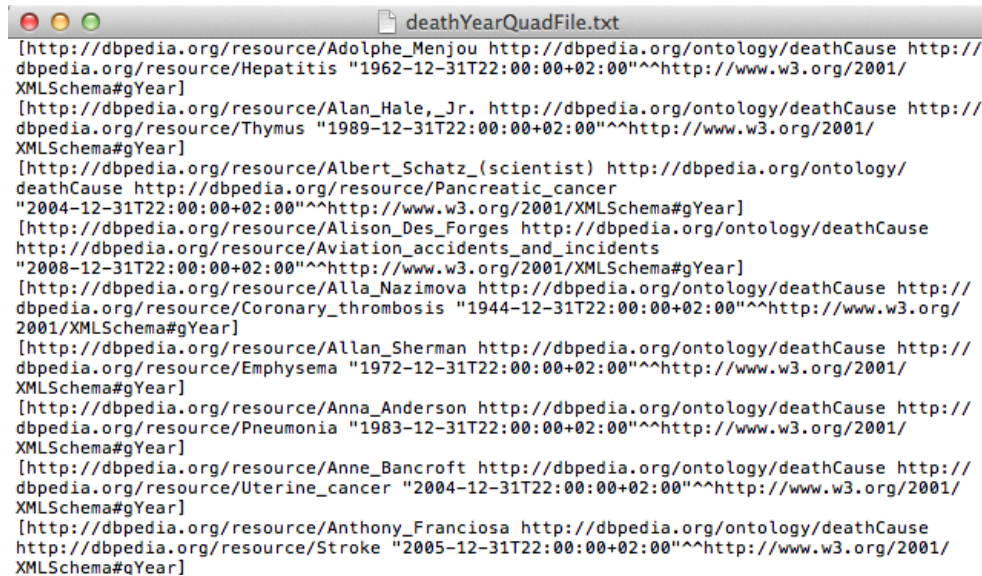
`?S rdfs:domain ?O`

rdfs : domain est une instance de *rdfs : Property* qui est utilisé pour indiquer que toute ressource qui possède une propriété donnée est une instance d'une ou plusieurs classes. Le triplet précédant indique que, *S* est une instance de la classe *rdf : Property*, *O* est une instance de la classe *rdfs : Class* et les ressources désignées par les sujets des triplets dont le prédicat est *S* sont des instances de la classe *O*. Lorsque une propriété *S* a plus d'une propriété *rdfs : domain*, les ressources indiquées par les sujets des triplets avec prédicat *S* sont des instances de toutes les classes indiquées par les propriétés *rdfs : domain*. *rdfs : domain* peut être appliqués à lui même. *rdfs : domain* de *rdfs : domain* est la classe *rdfs : Property*. Cela veut dire que toute ressource avec une propriété *rdfs : domain* est une instance de *rdf : Property*.

Ensuite, nous avons pensé à stocker l'ensemble de propriétés dans un fichier pour ne pas avoir des contraintes de mémoire. La procédure peut être faite une seule fois l'hors du lancement de l'application, mais elle ne sera plus nécessaire après, car il suffit de spécifier le nom du fichier. Ensuite, à partir de notre source de propriétés DBpédia, nous avons implémenté un algorithme d'extraction à comme sortie une liste de couples (PropriétéTemporelle,PropriétéReliée). Dans l'application, nous avons choisi de prendre l'avis d'un expert pour valider les résultats de notre algorithme à travers une liste labellisée de quadruplets sous forme d'un *outputTextarea*. L'objectif de cette procédure est de permettre à l'expert de valider ou ne pas valider la logique de la représentation des quadruplets. Enfin, la validation des résultats permet de stocker l'ensemble des résultats dans un fichier portant les labels du couple, un fichier contenant tous les quadruplets validés, par la suite, notre algorithme fait automatiquement l'appel à un fichier CSV d'historique qui a forme suivante :

(*attribute*,*TempAttribute*,*boolean*,*date_Exploration*,*keep*,*file*) que nous le détaillerons plus tard. L'ensemble de fichiers seront stockés dans un dossier forment une base de données de sortie que nous avons appelé SPOT-Base.

Avec certains couples, nous avons eu des très bons résultats, par exemple avec le couple (*deathCause*,*deathYear*) nous avons réussi à formé 2766 quads. La figure ci-dessous montre le format de la sortie de notre algorithme.



```

[http://dbpedia.org/resource/Adolphe_Menjou http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Hepatitis "1962-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]
[http://dbpedia.org/resource/Alan_Hale,_Jr. http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Thymus "1989-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]
[http://dbpedia.org/resource/Albert_Schatz_(scientist) http://dbpedia.org/ontology/
deathCause http://dbpedia.org/resource/Pancreatic_cancer
"2004-12-31T22:00:00+02:00"^^http://www.w3.org/2001/XMLSchema#Year]
[http://dbpedia.org/resource/Alison_Des_Forges http://dbpedia.org/ontology/deathCause
http://dbpedia.org/resource/Aviation_accidents_and_incidents
"2008-12-31T22:00:00+02:00"^^http://www.w3.org/2001/XMLSchema#Year]
[http://dbpedia.org/resource/Alla_Nazimova http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Coronary_thrombosis "1944-12-31T22:00:00+02:00"^^http://www.w3.org/
2001/XMLSchema#Year]
[http://dbpedia.org/resource/Allan_Sherman http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Emphysema "1972-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]
[http://dbpedia.org/resource/Anna_Anderson http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Pneumonia "1983-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]
[http://dbpedia.org/resource/Anne_Bancroft http://dbpedia.org/ontology/deathCause http://
dbpedia.org/resource/Uterine_cancer "2004-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]
[http://dbpedia.org/resource/Anthony_Franciosa http://dbpedia.org/ontology/deathCause
http://dbpedia.org/resource/Stroke "2005-12-31T22:00:00+02:00"^^http://www.w3.org/2001/
XMLSchema#Year]

```

FIGURE 12 – Fichier de Quadruplet DeathYear et DeathCause

Il se trouve aussi qu'il y a des couples qui valide notre hypothèse mais qui ne donnent pas des résultats. Il se peut que les deux triplets n'ont pas le même sujet comme $\{(wineRegion, wineYear), (whaDraft, whaDraftYear), (areaCode, areaDate), \text{etc...}\}$

Nous avons réussi à former 305 couples de propriétés, mais nous pouvons encore restreindre ce nombre. Dans notre méthode, nous avons choisi d'extraire même les propriétés reliées aux propriétés temporelles qui contiennent un motif similaire et non pas seulement qui sont identique au suffixe d'une propriété temporelle. Cela a été dans la mesure d'augmenter le nombre de nos données de test pour avoir une vision globale sur les propriétés DBpédia et analyser par la suite les différents résultats/possibilités.

Outils

Application Java

Le choix de développer le logiciel sous forme d'une application *Java* était un choix personnel et qui s'explique de nombreuses manières. Premièrement, la maîtrise de ce langage de programmation me permet d'utiliser des dif-

férentes structures de données et d'explorer la documentation de certaines méthodes plus facilement. De plus, il existe plusieurs sources de documentation sur le Web. En outre une large communauté aide à répondre aux questions et résoudre le problème. Il existe une version Java de DBpédia et cela permet plus facilement d'intégrer mon travail disponible en version open source sur mon compte⁴ git. Enfin, nous avons utilisé des librairies *Jena* qui sont aussi disponibles et écrites en Java.

Jena

Jena⁵ est un *Framework* open source écrit en Java pour construire des applications dans les domaines du *LinkedData* et le Web sémantique. Ce *Framework* est composé de plusieurs programmes différents qui interagissent entre eux pour traiter des données écrites en RDF. Jena fournit un support pour le langage de définition d'ontologies (OWL). Ce *Framework* se compose des différentes API RDF, Ontology et SPARQL. Une couche interface d'application et une troisième couche pour le stockage. La figure ci-dessous présente en détaille les différentes composantes de Jena.

4. <https://github.com/metanote/Extraction>

5. <https://jena.apache.org/>

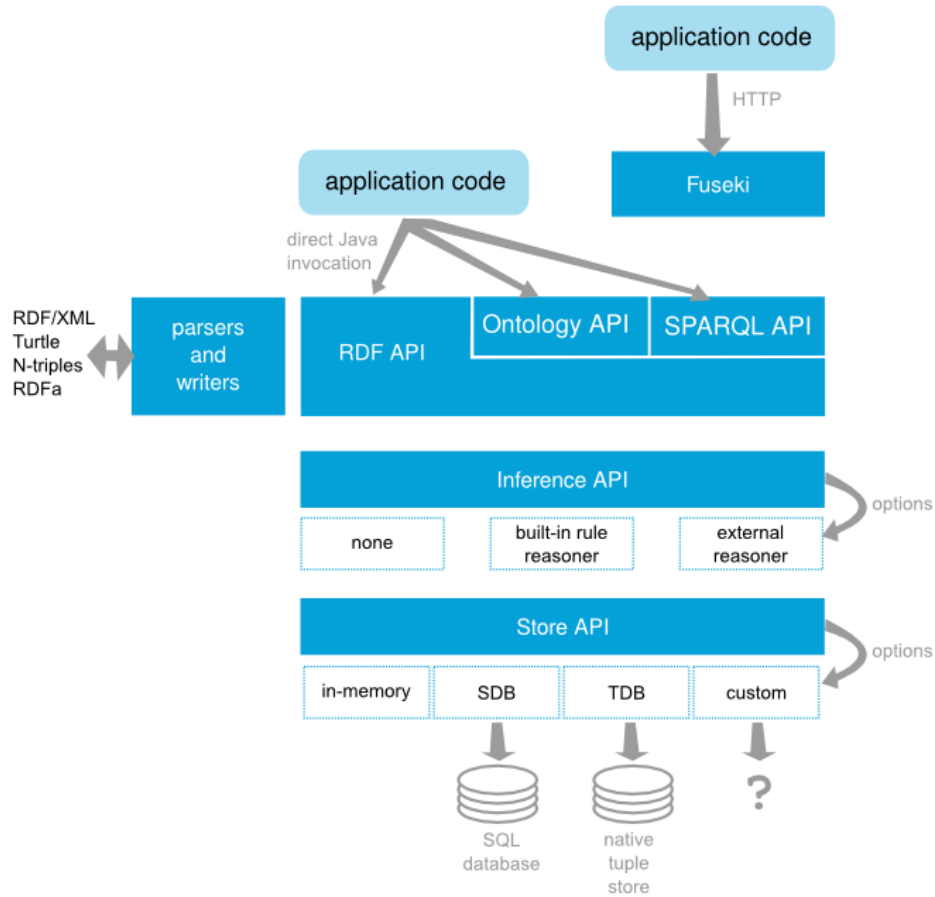


FIGURE 13 – Jena Interaction entre les différents API

Résultats et Validation

Durant cette étude, nous avons essayé d’annoter temporellement des triplets DBpedia. Nous avons réussi à formet automatiquement un nombre important de quadruplet à partir de couple de propriétés qui valide notre hypothèse. Pour certaines propriétés, notre programme prend une quinzaine de minutes des fois pour donner des résultats. Sachant qu’on utilise une machine sous OS x avec un processeur 2 GHz Intel Core i7 er 8 GO de mémoire. Cela est dû au nombre gigantesque de triplets qu’on interroge.

Nous avons stocké plus que 106998 quads que nous avons réussi à former dans le fichier *AllQaudsFile*. Nous avons remarqué que certains couples valident bien notre hypothèse et donne des excellents résultats. Certes, il y a d'autres couples ne donnent aucun résultat. Le problème, c'est que non pas toutes les propriétés dans DBpédia suivent la même logique de représentation. Si tous les triplets de DBpédia suivent la même logique de représentation, nous pouvons avoir beaucoup plus de résultats. Nous avons remarqué que dans le Web sémantique il est très important de mettre des conventions pour la représentation des données. Cela ne permet pas seulement d'utiliser les données existantes, mais aussi de mettre des hypothèses permettant de construire des travaux au dessus de ce qui existait déjà.

Résumé

Dans cette section, nous avons présenté notre approche à travers un système d'annotation temporelle des triplets dans DBpédia que nous avons déjà implémenté. Dans un premier lieu, nous avons effectué une étude préliminaire en analysant les besoins et en étudiant l'architecture de la base de connaissance. Par la suite, nous avons donné une deuxième solution plus intéressante et que nous avons développé sous forme d'un prototype fonctionnel.

Dans la prochaine section, nous allons continuer avec une procédure de fouille de données sur notre base de données de sortie SPOTBase tout en appliquant les techniques de la fouille pour analyser les résultats de notre travail. Ensuite nous présenterons les pistes d'amélioration possibles pour notre logiciel. Enfin, nous donnerons un récapitulatif qui résume la richesse de cette étude dans le domaine de l'ingénierie et la recherche.