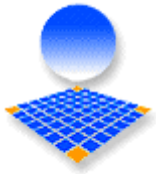
Ministère de l'Education Nationale

Université de Montpellier II

Rapport de Projet Informatique FMIN200
du Master informatique 1ère année effectué
du 31/01/2011 au 30/05/2011

Publication et interconnexion d'une ontologie de l'assemblée nationale et du jeu de données nosdeputes.fr sur le Web de données

Julien Plu



Remerciements

Je tiens à remercier M. François Scharffe pour son immense aide pour ce projet et son soutien, ainsi que M. Michel Plu pour ses avis et son regard professionnel sur le projet. Des remerciements aussi à M. Michel Leclère, Mme Marie-Laure Mugnier et Mlle Madalina Croitoru, tous trois professeurs de l'UE FMIN208 pour leur cours qui m'a été d'une très grande aide. D'autres remerciements vont à M. Christian Bizer et Richard Cyganiak pour m'avoir aidé à utiliser leurs outils. Merci aussi à Mlle Anne-Sophie Pala Massoni, étudiante en Master à l'école de sciences politique de Toulouse, pour son aide à la compréhension du fonctionnement de l'assemblée nationale. Des remerciements vont aussi aux administrateurs du site http://www.nosdeputes.fr pour leur soutien, leur conseil et leur aide. Et aussi à Thibaut Cuvelier pour ses remarques.

Table des matières

1. Qu'est-ce que le Web sémantique ? 6
1.1 Introduction 6
1.2 Histoire du Web sémantique 6
1.3 Objectifs du Web sémantique 8
1.4 Le Web sémantique et le Web 3.0 8
1.5 Le Web sémantique souvent critiqué 9
1.6 L'ontologie, outil principal du Web sémantique 9
1.6.1 Définition 9
1.6.2 Quelques ontologies déjà existantes 10
2. Introduction 10
2.1 Généralités 10
2.2 Le sujet 11
2.3 Cahier des charges 12
2.3.1 Objectifs 12
2.3.2 Public visé 12
2.3.3 Caractéristiques 12
2.3.4 Perspectives 12
2.3.5 Membre et encadrant du projet 12
2.3.6 Internationalisation 13
2.3.7 Ressources apportées 13
2.3.8 Principales tâches réalisées 13
2.3.9 Structuration du projet 13
2.3.10 Contraintes 13
3. Organisation du projet 13
3.1 Organisation du travail 14
3.2 Choix des outils de développement 14
4. Déroulement du projet 15
4.1 Étape 1 : conception de l'ontologie 15
4.2 Étape 2 : développement de l'ontologie 16
4.3 Étape 3 : publication de l'ontologie 31
4.4 Étape 4 : développement du script de mapping de la base de données 32
4.5 Étape 5 : publication du mapping de la base de données 34
4.6 Étape 6 : interconnexion 35
4.7 Étape 7 : développement d'une application 38
4.8 Installation et administration des serveurs 39
5. Manuel d'utilisation 40
6. Impressions personnelles 40
7. Perspectives et conclusions 41
7.1 Perspectives 41
7.2 Conclusions 41
7.2.1 Fonctionnement de l'application 41
7.2.2 Fonctionnement du groupe de travail 41
8. Annexe : documents d'analyse 42
8.1 Scan de l'ontologie OAN 42

1. Qu'est-ce que le Web sémantique ?

Avant d'introduire les spécifications de ce projet, je vais introduire ce que peux être le Web sémantique et une ontologie pour que tous les lecteurs puissent comprendre de quoi ce rapport parlera par la suite.

1.1 Introduction

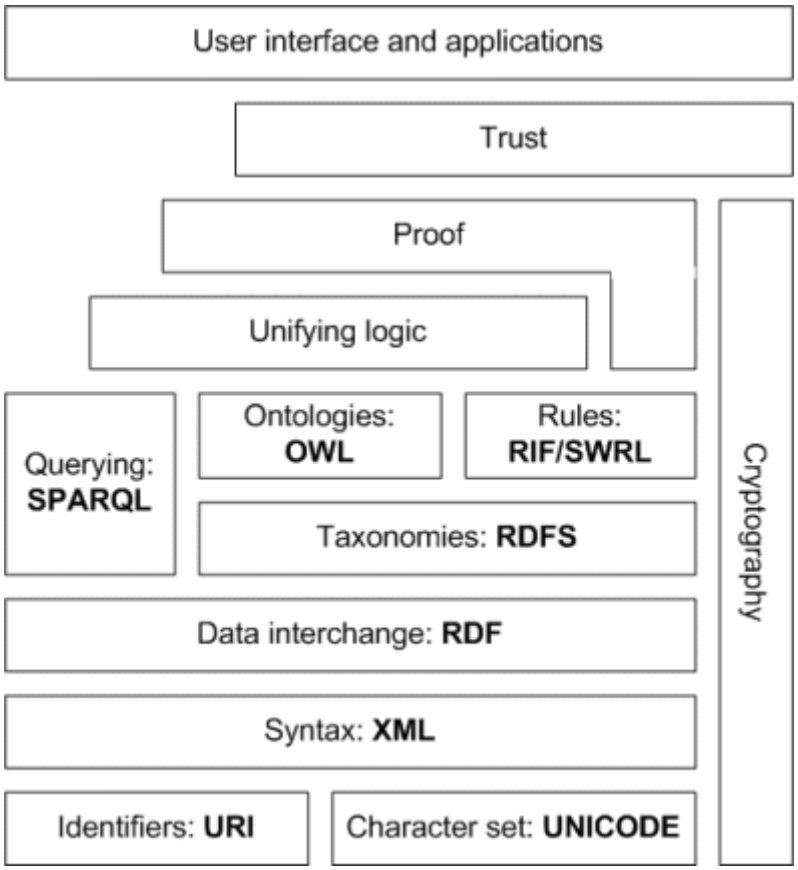
Le Web sémantique (plus techniquement appelé « le Web de données ») permet aux machines de comprendre la sémantique, la signification de l'information sur le Web. Il étend le réseau des hyperliens entre des pages Web classiques par un réseau de lien entre données structurées permettant ainsi aux agents automatisés d'accéder plus intelligemment aux différentes sources de données contenues sur le Web et, de cette manière, d'effectuer des tâches (recherche, apprentissage, etc.) plus précises pour les utilisateurs. Le terme a été inventé par Tim Berners-Lee, co-inventeur du Web et directeur du W3C, qui supervise l'élaboration des propositions de standards du Web sémantique.

La plupart du temps, lorsque l'on prononce le terme de Web sémantique, on parle des différentes technologies qui se cachent derrière. Parmi les plus connus, on peut citer RDF (Ressource Description Framework) qui correspond à un modèle d'information, des formats d'échanges de données en RDF pour communiquer entre différentes applications (RDF/XML, RDF/JSON, N3, Turtle, N-Triples et d'autres). Dans le domaine du Web sémantique, la sémantique des données est décrite par des ontologies - ce terme sera défini plus loin dans l'article - avec des langages prévus pour fournir une description formelle de concepts, termes ou relations d'un domaine quelconque. Ces langages sont RDFS (*Ressource Description Framework Schema*) et OWL (*Web Ontology Language*). Il existe aussi des langages de description des données structurées dans du XHTML afin que des outils effectuent un traitement automatique de ces différentes données. Ces langages sont RDFa et Microformat. Ensuite, pour finir avec la liste des technologies, il existe un langage de requête, au même titre que SQL pour les bases de données relationnelles, SPARQL, qui effectue des requêtes sur des triplets RDF. Il en existe d'autres (RQL et RDQL), mais ils sont bien moins utilisés.

1.2 Histoire du Web sémantique

En 1994, lors de la première conférence WWW à Genève, plus précisément au CERN, a lieu l'annonce de la création du W3C. C'est d'ailleurs à cette période que Tim Berners-Lee dresse les objectifs du W3C et montre les besoins d'ajouter de la sémantique au Web futur. Il montre alors en quoi les liens hypertextes ou, plus précisément, la façon dont on met en relation les documents sur le Web est trop limitée pour permettre aux machines de relier automatiquement les données contenues sur le Web à la réalité. Compte tenu de l'ambition d'un tel projet, cette idée suscite quelques résistances et controverses qui sont classiquement rencontrées dès qu'on aborde des problématiques liées au domaine de l'intelligence artificielle.

Après cette conférence, mise à part la mise en place des recommandations nécessaires à la construction des documents, le W3C nouvellement créé entame les premières réflexions sur la mise en place du Web sémantique. Ces réflexions aboutissent à la publication d'un premier *draft* de recommandations sur le Web sémantique en octobre 1997 et d'un second en avril 1998. Cette même année, Tim Berners-Lee publie un document sur les toutes premières moutures de ce qui sera plus tard appelé le Web sémantique. Ces moutures consistent à mettre en place les différentes technologies du Web sémantique. Dans ce document, il présente le Web sémantique comme une sorte d'extension du Web des documents, qui constitue une base de données à l'échelle mondiale, afin que toutes les machines puissent mieux lier les données du Web. Cette feuille de route se matérialise par une représentation graphique, le « layer cake », qui montre l'agencement des différentes briques technologiques du Web sémantique.



Par ailleurs, en 1999, Tim Berners-Lee publie le livre *Weaving the Web* dans lequel il dresse un portrait du Web et les pistes pour son avenir. Les idées du Web sémantique n'en sont évidemment pas absentes. C'est d'ailleurs cette même année qu'il énonça sa célèbre citation :

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines

talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

J'ai fait un rêve pour le Web [dans lequel les ordinateurs] deviennent capables d'analyser toutes les données sur le Web - le contenu, les liens et les transactions entre les personnes et les ordinateurs. Un « Web sémantique », qui devrait rendre cela possible, n'est pas encore sorti, mais, quand ce sera le cas, les mécanismes d'échanges au jour le jour, la bureaucratie et notre vie quotidienne seront traitées par des machines qui parlent à d'autres machines. Certains nous ont vanté depuis des lustres les « agents intelligents » et cela va enfin se concrétiser.

Cette citation provient de cette adresse[1].

1.3 Objectifs du Web sémantique

Un des principaux objectifs du Web sémantique est de permettre aux utilisateurs d'utiliser la totalité du potentiel du Web : ainsi, ils pourront trouver, partager et combiner des informations plus facilement. Aujourd'hui tout le monde est capable d'utiliser des forums, d'utiliser des réseaux sociaux, de chatter, de faire des recherches ou même d'acheter différents produits. Néanmoins, il serait mieux que la machine fasse tout ceci à la place de l'homme, car, actuellement, les machines, ont besoin de l'homme pour effectuer ces tâches. La raison principale est que les pages Web actuelles sont conçues pour être lisible par des êtres humains et non par des machines.

Le Web sémantique a donc comme principal objectif que ces mêmes machines puissent réaliser seules toutes les tâches fastidieuses comme la recherche ou l'association d'informations et d'agir sur le Web lui-même.

1.4 Le Web sémantique et le Web 3.0

La communauté du Web dans son ensemble à tendance à dire que les deux termes « Web sémantique » et « Web 3.0 » représentent à peu près le même concept, si ce n'est pas totalement interchangeable. La définition continue de varier en fonction des gens avec qui on parle. L'avis général est que le Web 3.0 est très certainement la prochaine grande révolution, mais il se trouve que, pour le moment, ce ne sont que des spéculations quant à ce qu'il pourrait être. Il y aura encore de grosses améliorations, mais en gardant la plupart des propriétés du Web 2.0. Il y en a certains qui prétendent que le Web 3.0 sera plus applicatif et centrera ses efforts vers des environnements plus graphique, d'autres qui prétendent qu'il sera plus axé sur la recherche d'information géographique basés sur la géolocalisation ou encore même utiliser les nombreux progrès en intelligence artificielle.

[1] http://en.wikipedia.org/wiki/Semantic_web

1.5 Le Web sémantique souvent critiqué

Comme vous avez pu le deviner, le Web sémantique permet à tout un chacun d'en savoir plus sur les sujets qui les intéressent. Cela signifie *a contrario* que diverses institutions peuvent récolter des informations sur vous librement et donc constituer des dossiers. Ces institutions peuvent être des agences de publicité, de sécurité ou même des services de renseignements. Et ceci, tout en restant dans la légalité car, dans 90 % des cas, ces informations sont mises en lignes volontairement ou non par les utilisateurs, sans qu'ils se doutent du « danger » que peut représenter cette action.

Le Web sémantique est aussi critiqué à cause de sa lourdeur : les langages utilisés pour le Web sémantique sont très verbeux car dérivés du XML et donc souvent pénibles à utiliser. De ce fait, l'écriture d'ontologies est souvent très problématique car elle exige une spécialisation dans un domaine particulier et, lorsque l'on ne maîtrise pas ce domaine, elle devient très difficile à créer. Ainsi, certaines personnes disent qu'il est préférable d'utiliser des « word tags » (ce sont une série de mots-clés qui permettent de qualifier une ressource) à la place des ontologies.

1.6 L'ontologie, outil principal du Web sémantique

Cette partie explique de manière non technique ce qu'est une ontologie.

1.6.1 Définition

L'ontologie est la base de ce que l'on appelle la représentation des connaissances. Ce domaine est né de la volonté des chercheurs de représenter diverses connaissances humaines sous une forme utilisable par des ordinateurs pour effectuer des raisonnements. Ces connaissances sont exprimées sous formes de symboles auxquels on donne une signification : une « sémantique ».

Imaginons la problématique suivante : ayant une base de documents (textes, images...) et une requête, comment trouver les documents pertinents pour cette requête ?

En général, votre moteur de recherche préféré recherche la suite de mots de votre requête dans les textes des documents indexées (recherche *full-text*) ou dans des listes de mots associés au document (aussi appelées métadonnées).

Par exemple on tape dans notre moteur de recherche préféré les mots suivants pour des images : « automobile » puis « voiture ». On s'aperçoit que les résultats ne sont pas du tout les mêmes, alors que, vu que les deux mots représentent la même chose, on pourrait s'attendre à trouver les mêmes images.

Que se passe-t-il ? En fait, le moteur de recherche compare des mots sans prendre en compte leur sémantique. Il exécute uniquement une recherche purement syntaxique et donc sans réflexion et sans gestion des synonymes car « automobile » et « voiture »

représentent le même concept. Plus précisément, on peut dire qu'il n'y a pas de gestion de la relation de généralisation/spécialisation sur les concepts. Par exemple, « taxi » est une spécialisation du concept de « voiture » et « véhicule » est une généralisation du concept de « voiture ». Ainsi, pour raisonner, il ne faut plus se baser sur les mots mais sur les concepts. Mais que signifie raisonner ? Raisonner est l'action de produire des connaissances à partir de connaissances. C'est un terme très souvent employé en intelligence artificielle.

Ainsi, pour résoudre ce problème, on construit des bases de connaissances, constituées de :

- une **ontologie** : un ensemble de concepts et de relations entre ces concepts ;
- **règles** : une expression de contraintes sur les relations et concepts de l'ontologie ;
- **faits** : des individus de l'ontologie.

1.6.2 Quelques ontologies déjà existantes

Voici une petite liste non exhaustive des ontologies existantes :

- FOAF : elle permet de décrire des personnes et les relations qu'elles entretiennent entre elles ;
- EVENT : elle permet de décrire des évènements ;
- DC : elle permet de décrire des documents numériques ;
- VCARD : elle permet juste de décrire des personnes ;
- CREATIVE COMMONS : elle permet de décrire les différentes licences d'un document.

2. Introduction

2.1 Généralités

Ce projet est fait pour être incorporé avec d'autres projets similaires au projet Datalift[2]. Mais qu'est-ce que Datalift ? Voici une introduction et une explication du but de Datalift :

L'objectif principal de Datalift est d'amorcer l'élévation sémantique des données brutes sur le Web. Plusieurs sous-objectifs permettent d'atteindre cet objectif principal : il est tout d'abord nécessaire de fournir les ontologies qui permettront aux fournisseurs de données de décrire celles-ci. Datalift recherche et développe des méthodes et outils permettant aux fournisseurs de données de sélectionner les ontologies permettant de

[2] http://www.datalift.org

décrire leurs données. Il est aussi nécessaire de convertir les données brutes en RDF en conformité avec les ontologies sélectionnées. Le consortium de Datalift recherche et développe une suite d'outils intégrés qui facilitera le processus de conversion pour un large éventail de formats de données sources (relationnel, XML, feuilles de calcul, microformats et autres formats de métadonnées). La puissance du Web de données venant de la quantité et de la qualité des liens entre les ressources qu'il contient, Datalift va rechercher sur ce sujet avec pour but d'automatiser le processus d'interconnexion. Datalift va aussi rechercher des méthodes nouvelles pour attacher des licences aux données liées. En intégrant toutes ces technologies, la plateforme Datalift va fournir une chaine complète pour le processus d'élévation sémantique des données avec pour objectif de devenir la plateforme de référence pour ce processus. Datlift va finalement publier les données à travers un réseau de fournisseurs de données, et avec pour objectif de constituer une base de données sur le Web suffisante pour la création d'applications innovantes et ainsi d'amener d'autres fournisseurs à élever leurs données.

Maintenant que l'on a explicité le but du projet, on va passer en revue les différentes étapes qui compose la publication d'un jeu de données afin que nous puissions avoir une vue d'ensemble de ce projet :

- trouver un jeu de données ;
- créer une ontologie dans le domaine du jeu de données ;
- mapper le jeu de données avec notre ontologie afin de produire nos données sous forme de triplets RDF ;
- interconnecter les données déjà contenus sur le Web de données avec les nôtres.

On connait dorénavant les différentes étapes qui composent un projet de publication de données, on peut donc commencer l'explication en détail du projet de ce TER.

2.2 Le sujet

Le but de ce TER (travail d'étude et de recherche) consiste à publier des jeux de données dans un domaine à déterminer (musique, publications scientifiques, sciences de la vie) sur le Web de données. Le processus de publication consiste à :

- sélectionner un vocabulaire adapté et prétraiter les données ;
- les convertir ou exposer suivant l'approche choisie et les publier en utilisant un « triplet store » ;

- les interconnecter avec d'autres jeux de données existants ;
- mettre en place une interface permettant de naviguer dans les données enrichies et de démontrer ainsi les avantages du Web sémantique.

2.3 Cahier des charges

2.3.1 Objectifs

- Publier un jeu de données sur le Web de données.
- Faire découvrir le Web sémantique.
- Interconnecter notre jeu de données avec ceux déjà existant sur le Web.

2.3.2 Public visé

D'abord pour les étudiants puis pour les professeurs et enfin pour toutes les personnes désireuses de s'informer sur le Web sémantique.

2.3.3 Caractéristiques

Ce projet est un mélange de diverses technologies du Web sémantique et est surtout simple d'utilisation ainsi qu'interopérable entre les différents jeux de données existant sur le Web.

2.3.4 Perspectives

- Faire adopter le Web sémantique aux concepteurs du site http://www.nosdeputes.fr.
- Apport de données pour le projet Datalift.
- Proposer une publication des données accessible depuis le site de l'assemblée nationale.

2.3.5 Membre et encadrant du projet

- François Scharffe, scharffe@lirmm.fr, qui est l'encadrant de ce projet.
- Julien Plu, julien.plu@etud.univ-montp2.fr.

2.3.6 Internationalisation

L'utilisation est réservée à la France, car l'ontologie est propre au système politique de ce pays.

2.3.7 Ressources apportées

- Publication du projet sur un serveur du LIRMM.
- Documents sur support informatique (pdf, site internet, …).
- Ontologie pour l'assemblée nationale française.

2.3.8 Principales tâches réalisées

- Conception de l'ontologie en OWL.
- Mise en place d'une plate-forme de publication d'ontologies, Neologism.
- Mapping de la base de données avec D2R pour un faire des triplets RDF.
- Installation d'un triplet store, Sesame.
- Interconnexion des données avec Silk.
- Développement d'une application de test avec Jena et GWT.

2.3.9 Structuration du projet

- Étape 1 : conception de l'ontologie.
- Étape 2 : développement de l'ontologie.
- Étape 3 : publication de l'ontologie.
- Étape 4 : développement du script de mapping de la base de données.
- Étape 5 : publication du mapping de la base de données.
- Étape 6 : interconnexion.
- Étape 7 : développement d'une application.

2.3.10 Contraintes

- Ne pas toucher au schéma et au contenu de la base de données.
- Quatre mois de développements.

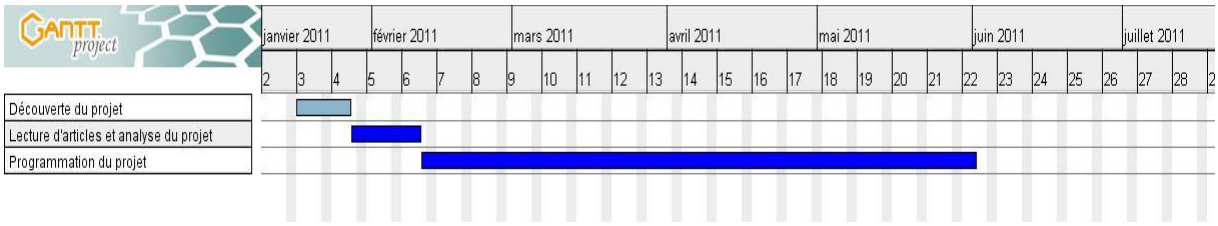
3. Organisation du projet

Après avoir découvert ce qu'était le Web sémantique et les ontologies, il est grand temps de penser au projet en lui-même.

3.1 Organisation du travail

Étant le seul membre de ce projet, la répartition des tâches n'était pas très difficile à organiser. Avec M. Scharffe, nous nous sommes vus à peu près une fois par semaine, le mercredi à 15h au secrétariat informatique plus exactement.

Durant la première réunion nous avons fait une planification du travail selon le diagramme de Gantt suivant :



Ensuite, M. Scharffe m'a donné deux articles scientifiques sur le Web sémantique à lire avant de commencer le projet. Ces articles sont :

- Linked Data - The Story So Far, Christian Bizer, Tom Heath, Tim Berners-Lee, http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf
- How to Publish Linked Data on the Web, Christian Bizer, Tom Heath, Richard Cyganiac, http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

Une fois lus, nous nous sommes revus pour faire une découpe étape par étape du projet. Ensuite, ce fut à chaque fois pour faire le point sur le travail accompli de la semaine.

3.2 Choix des outils de développement

Voici la liste des outils et des langages utilisés pour développer ce projet :

- OWL, pour le langage d'écriture de l'ontologie ;
- Protégé, pour la conception de l'ontologie ;
- D2R, pour le mapping entre la base de données et des triplets RDF, avec son langage de script D2RQ ;
- Neologism, pour la publication et la documentation de l'ontologie ;
- LODE, pour la documentation de l'ontologie ;
- Silk, pour l'interconnexion de nos triplets avec ceux de triplets store externes ;
- MySQL, pour l'hébergement de la base de données ;
- Apache, pour le serveur web afin d'avoir les données toujours accessible sur le Web ;
- Sesame, pour stocker ces données sous forme de triplets ;
- Apache Tomcat, pour le serveur d'application, hébergeant D2R et Sesame ;
- GWT, pour la conception d'une petite application de test ;
- Jena, pour le framework de manipulation de données RDF et de requêtes SPARQL.

Une petite précision sur le choix de LODE et Neologism, bien qu'ils soient similaire pour certains points. Neologism, du fait de son jeune âge, ne prend pas encore en compte toutes les possibilités offertes par OWL ; pour la même raison, sa documentation est bien inférieure à celle de LODE. Par contre, en ce qui concerne la présentation, un énorme effort a été fourni par Neologism, en comparaison de LODE. C'est pour toutes ces raisons qu'il a été décidé d'utiliser conjointement les deux, pour obtenir de chacun le meilleur dont il est capable.

4. Déroulement du projet

Le projet a été découpé en sept parties, comme précédemment indiqué. Comment en est-on arrivé à ce découpage ? En quoi consistaient ces différentes parties ?

4.1 Étape 1 : conception de l'ontologie

Après un moment d'explications préliminaires au sujet, nous avons décidé de réfléchir sur la manière de bien réaliser le projet. On a donc discuté des différentes étapes du projet pour voir si ces étapes ne devaient pas se découper aussi en différentes étapes.

La première chose qui nous est venu à l'esprit est, quelles sont les jeux de données à utiliser ? On s'est donc mis à chercher. Au bout d'un certain temps, nous avons trouvé les données du site http://www.nosdeputes.fr que nous avons décidé de choisir ce jeu de données.

Le site http://www.nosdeputes.fr est une initiative du collectif RegardsCitoyens.org, une association constituée de citoyens de tous âges et régions qui se sont rencontrés sur Internet dans un désir commun de proposer un accès simplifié au fonctionnement de nos institutions démocratiques.

De plus, les administrateurs du site étaient curieux de voir ce que cela pouvait donner si l'on mettait toutes leurs données sur le Web sémantique, puisqu'ils en avaient déjà discuté avec M. Scharffe.

Le jeu de données étant trouvé, il fallait maintenant concevoir l'ontologie de notre sujet. Étant totalement débutant dans le domaine de l'organisation de l'assemblée nationale française, j'ai donc pour cette raison fait appel à une amie qui connaît bien ce domaine puisqu'elle est étudiante en Master à l'école de Science Politique de Toulouse et à un poste à haute responsabilité dans un partie politique. Après de longues discussions avec cette personne, je parviens à comprendre le mode de fonctionnement de l'assemblée nationale française. C'est donc à ce moment précis que je commence à regarder le contenu de la base de données, histoire de voir et comprendre comment les administrateurs du site http://www.nosdeputes.fr ont conçu cette base de données.

Je me suis alors heurté à un problème. Ce problème concernait la conception de cette base de données, car il fallait déjà faire le tri entre les données utiles et celles qui ne le sont pas. Par exemple, les commentaires des internautes ne nous intéressent pas, au contraire du texte des articles de loi. Ensuite, autre problème avec la base de données, son organisation. J'ai eu beaucoup de mal à voir comment elle était organisée.

Avec ces deux problèmes, autant dire que la conception de l'ontologie ne s'annonçait pas d'une réelle simplicité. Il nous a d'ailleurs fallu nous y prendre plus d'une fois avant d'arriver à la version finale et cette difficulté nous a valu un léger retard sur le planning du projet. À ce moment-là, j'ai donc pris contact avec les administrateurs du site http://www.nosdeputes.fr pour discuter avec eux de leur conception et c'est avec plus ou moins de mal que nous avons réussi à concevoir cette ontologie. Nous avons aussi décidé de tirer parti d'ontologies déjà existantes comme FOAF et VCard pour la représentation d'un député et de Event pour la représentation d'une intervention. À la fin de cette conception nous avons obtenu une ontologie comportant 19 classes et 77 propriétés. Les schémas de l'ontologie se trouvent en annexe.

4.2 Étape 2 : développement de l'ontologie

Une fois l'ontologie sur papier faite, il a fallu passer au développement en OWL. J'ai tout d'abord commencé à développer mon ontologie avec Protégé, mais je me suis confronté à un problème : la manipulation et la compréhension de cet outil, car, non seulement, je ne connaissais pas le langage OWL, mais, en plus, je ne savais pas trop ce que je faisais avec Protégé. J'ai donc pris la décision, pour éviter de perdre encore plus de temps, de développer l'ontologie à la main avec la documentation du W3C. J'ai donc entrepris une longue lecture de celle-ci afin d'apprendre à développer cette ontologie.

Une fois l'ontologie finie, il a fallu valider cette ontologie avec le validateur de fichier RDF du W3C et le validateur OWL de l'université de Manchester. Ainsi le fichier était bien formé et ne comportait pas d'erreur de syntaxe.

J'ai nommé cette ontologie « Ontologie de l'assemblée nationale » (ou OAN). Dans le détail de cette ontologie, de quoi est-elle composée et que permet-elle ?

Tout d'abord elle se découpe en différents modules :

- Les députés : caractéristiques qu'un député peut avoir (nom, prénom, questions…)
- Les commissions : ce qui concerne une commission (organisme, rapport, intervention…)
- Les lois : comment et de quoi est constituée une loi (section, article, amendement…)

Maintenant faisons un tableau énumérant toutes les classes et propriétés contenus dans chaque module.

Tout d'abord les classes pour le module des députés :

Nom de la classe	Commentaire
oan:Depute	La classe « Depute » sert à représenter la notion de député siégeant à l'assemblée nationale. Un député est un représentant auprès de l'assemblée nationale élu par le peuple.
oan:Mandat	La classe « Mandat » représente les différents mandats qu'il est possible d'avoir pour un député.
oan:QuestionEcrite	La classe « QuestionEcrite » représente les différentes questions qui sont écrites par un député pour les ministres du gouvernement.
oan:Reponse	La classe « Reponse » représente la réponse des ministres aux différentes questions écrites émises par les députés.

Les propriétés pour le module des députés :

Nom de la propriété	Domain	Range	Commentaire
oan:aPourMandat	oan:Depute	oan:Mandat	La propriété « aPourMandat » représente le mandat d'un député.
oan:numCirco	oan:Depute	xsd:nonNegativeInteger	La propriété « numCirco » représente le numéro de la circonscription dont le député est responsable.
oan:placeHemicycle	oan:Depute	xsd:nonNegativeInteger	La propriété « placeHemicycle » représente le numéro de la place qu'occupe le député à l'assemblée nationale.

oan:nomMandat	oan:Mandat	xsd:string	La propriété « nomMandat » représente le nom du mandat que peut avoir un député.
oan:finMandat	oan:Mandat	xsd:string	La propriété « finMandat » représente la fin du mandat d'un député.
oan:debutMandat	oan:Mandat	xsd:string	La propriété « debutMandat » représente le début du mandat d'un député.
oan:appartientOrganisme	oan:Depute	oan:Organisme	La propriété « appartientOrganisme » représente l'organisme auquel appartient un député.
oan:estEcrite	oan:Depute	oan:QuestionEcrite	La propriété « estEcrite » représente le député qui a écrit cette question écrite.
oan:texteQuestion	oan:QuestionEcrite	xsd:string	La propriété « texteQuestion » représente le texte de la question écrite émise par le député pour un ministre.
oan:themeQuestion	oan:QuestionEcrite	xsd:string	La propriété « themeQuestion » représente le thème de la question écrite émise par le député pour un ministre.

oan:legislatureQuestion	oan:questionEcrite	xsd:nonNegativeInteger	La propriété « legislatureQuestion » représente la législature sous laquelle la question écrite a été émise.
oan:pourMinistere	oan:QuestionEcrite	xsd:string	La propriété « pourMinistere » représente le nom du ministère pour lequel la lettre a été écrite.
oan:reponseCorrespondante	oan:QuestionEcrite	oan:Reponse	La propriété « reponseCorrespondante » représente la réponse écrite à une question écrite émise par un député.
oan:dateEcriture	oan:QuestionEcrite	xsd:date	La propriété « dateEcriture » représente la date à laquelle a été émise la question écrite.
oan:motifRetrait	oan:QuestionEcrite	xsd:string	La propriété « motifRetrait » représente le motif pour lequel la question écrite a été supprimé.
oan:numeroQuestion	oan:QuestionEcrite	xsd:nonNegativeInteger	La propriété « numeorQuestion » représente le numéro de la question écrite.
oan:texteReponse	oan:Reponse	xsd:string	La propriété « texteReponse » représente le texte de la réponse à une question écrite émise par un ministre.

oan:dateReponse	oan:Reponse	xsd:date	La propriété « dateReponse » représente la data à laquelle a été émise la réponse à une question écrite.

Dans ce module, la classe « oan:Depute » est une sous-classe de la classe « foaf:Person » (http://xmlns.com/foaf/spec/#term_Person) de l'ontologie FOAF qui sert à représenter une personne. Pour les besoins du projet nous avons rajouté deux propriétés à cette classe qui sont :

Nom de la propriété	Domain	Range	Commentaire
oan:sexe	foaf:Person	{ « F » , « H » }	La propriété « sexe » représente le sexe du député. Cette propriété a été rajouté à la super classe « foaf:Person » car c'était plus convenable de procéder de cette manière
oan:Metier	foaf:Person	xsd:string	La propriété « metier » représente le métier du député. Cette propriété a été rajouté à la super classe « foaf:Person » car c'était plus convenable de procéder de cette manière.

Voyons maintenant les classes contenus dans le module des commissions :

Nom de la classe	Commentaire
oan:Organisme	La classe « Organisme » représente les différents organismes possibles auxquels un député peut appartenir.
oan:Intervention	La classe « Intervention » représente l'intervention d'un député lors d'un rassemblement à l'assemblée nationale ou bien lors d'une commission.
oan:Seance	La classe « Seance » représente les différents rassemblements auxquels un député assiste. Ces rassemblements sont généralement les sessions de l'assemblée nationale et les commissions.

oan:Rapport	La classe « Rapport » sert à représenter la notion de rapport écrit par les députés lors de commission ou d'hémicycle. Un rapport est établi dans le but de proposer des solutions à un problème.
oan:Presence	La classe « Presence » représente la présence d'un député lors d'une commission ou d'une session de l'assemblée nationale.
oan:PreuvePresence	La classe « PreuvePresence » représente les différents types de preuves qu'il est possible d'avoir afin de prouver la présence d'un député à une commission ou à une session de l'assemblée nationale.

Et voici les propriétés que l'on peut y trouver :

Nom de la propriété	Domain	Range	Commentaire
oan:typeOrganisme	oan:Organisme	{ « extra » , « groupe » , « parlementaire » }	La propriété « typeOrganisme » représente les différents types existant d'organismes.
oan:nomOrganisme	oan:Organisme	xsd:string	La propriété « nomOrganisme » représente le nom d'un organisme.
oan:nbMots	oan:Intervention	xsd:nonNegativeInteger	La propriété « nbMots » représente le nombre de mots que contient l'intervention d'un député.

oan:typeIntervention	oan:Intervention	{ « loi » , « question » }	La propriété « typeIntervention » représente les différents types d'intervention possible que peuvent faire les députés lors de divers séances d'un organisme quelconque.
oan:deputeIntervenant	oan:Intervention	oan:Depute	La propriété « deputeIntervenant » représente le député qui est l'auteur de l'intervention.
oan:intervientLors	oan:Intervention	oan:Seance	La classe « Seance » représente les différents rassemblements auxquels un député assiste. Ces rassemblements sont généralement les sessions de l'assemblée nationale et les commissions.
oan:dateIntervention	oan:Intervention	xsd:date	La propriété « dateIntervention » représente la date à laquelle est faite l'intervention.
oan:fonctionIntervenant	oan:Intervention	xsd:string	La propriété « fonctionIntervenant » représente la fonction du député qui intervient durant une séance d'un quelconque organisme.

oan:texteIntervention	oan:Intervention	xsd:string	La propriété « texteIntervention » représente le texte de l'intervention.
oan:typeSeance	oan:Seance	{ « commission » , « hémicycle »	La propriété « typeSeance » représente les différents types existant de séances.
oan:dateSeance	oan:Seance	xsd:date	La propriété « dateSeance » représente la date à laquelle à lieu une séance d'un quelconque organisme.
oan:numeroSemaineSeance	oan:Seance	xsd:nonNegativeInteger	La propriété « numeroSemaineSeance » représente le numéro de la semaine à laquelle a lieu la séance.
oan:heureDebutSeance	oan:Seance	xsd:string	La propriété « heureDebutSeance » représente l'heure à laquelle débute une séance.
oan:seanceAppartient	oan:Seance	oan:Organisme	la propriété « seanceAppartient » représente la séance d'un organisme.
oan:seanceProduit	oan:Seance	oan:Rapport	La propriété « seanceProduit » représente les différents rapports qui sont produit lors d'une séance d'un organisme quelconque.

oan:numeroRapport	oan:Rapport	xsd:nonNegativeInteger	La propriété « numeroRapport » représente le numéro donné au rapport par les différents organismes qu'il concerne.
oan:dateRapport	oan:Rapport	xsd:date	La propriété « dateRapport » représente la date à laquelle le rapport a été créé.
oan:auteurRapport	oan:Rapport	oan:Depute	La propriété « auteurRapport » représente le député qui est l'auteur du rapport.
oan:seanceCorrespondante	oan:Presence	oan:Seance	La propriété « seanceCorrespondante » représente la présence d'un député correspondant à une séance d'un organisme quelconque.
oan:datePresence	oan:Presence	xsd:date	La propriété « datePresence » représente la date à laquelle a été faite la présence.
oan:deputeCorrespondant	oan:Presence	oan:Depute	La propriété « deputecorrespondant » représente la présence d'un député.
oan:nbPreuvesPresence	oan:Presence	xsd:nonNegativeInteger	La propriété « nbPreuvesPresence » représente le nombre de preuves d'une présence que peut avoir un député.

oan:presenceCorrespondante	oan:PreuvePresence	oan:Presence	La propriété « presenceCorrespondante » représente la preuve d'une présence d'un député à une séance à la présence à cette séance.
oan:typePreuve	oan:PreuvePresence	{ « intervention » }	La propriété « typePreuve » représente le type d'une preuve de présence d'un député à une séance d'un organisme.

Dans ce module, la classe « oan:Intervention » est une sous classe, de la classe « Event:event » (http://motools.sourceforge.net/event/event.html#term_Event) de l'ontologie Event qui sert à représenter un évènement.

Voyons maintenant le contenu du dernier module, celui des lois avec tout d'abord une liste de ses classes :

Nom de la classe	Commentaire
oan:Article	La classe « Article » représente les différents articles qui peuvent être contenus dans un texte de loi.
oan:Alinea	La classe « Alinea » représente les différents alinéas qui sont contenus dans un article.
oan:PositionArticle	La classe « PositionArticle » indique dans quelle partie du texte de loi est contenu l'article correspondant.
oan:Loi	La classe « Loi » sert à représenter la notion de loi établi par les députés lors de différents hémicycles. Une loi est une règle juridique définie lors de différents rassemblements de l'assemblée nationale par les députés et les membres du gouvernement. Un projet de loi émane d'un membre du gouvernement et une proposition de loi émane d'un parlementaire.
oan:TexteLoi	La classe « TexteLoi » représente le texte contenu dans une loi. Pour plus de détails voir la classe « Loi ».
oan:Chapitre	La classe « Chapitre » représente les différents chapitres contenus dans un texte de loi.
oan:Section	La classe « Section » représente les différentes sections contenus dans un texte de loi.

oan:SousSection	La classe « SousSection » représente les sous sections contenus dans un texte de loi.
oan:Amendement	La classe « Amendement » sert à représenter la notion d'amendement. Un amendement est une demande de modification d'une loi existante.

Pour finir, ses propriétés :

Nom de la propriété	Domain	Range	Commentaire
oan:alineaNumero	oan:Alinea	xsd:nonNegativeInteger	La propriété « alineaNumero » représente le numéro d'un alinéa.
oan:texteAlinea	oan:Alinea	xsd:string	La propriété « texteAlinea » représente le texte d'un alinéa.
oan:articleCorrespondant	oan:Alinea	oan:Article	La propriété « articleCorrespondant » représente l'article correspondant à l'alinéa.
oan:texteLoiCorrespondantAlinea	oan:Alinea	oan:TexteLoi	La propriété « texteLoiCorrespondantAlinea » représente le texte de loi correspondant à l'alinéa.
oan:loiCorrespondante	oan:Alinea	oan:Loi	La propriété « loiCorrespondante » indique à quelle loi est rattaché l'alinéa.

oan:positionDans	oan:Article	oan:PositionArticle	La propriété « positionDans » représente la position de l'article dans un texte de loi, c'est-à-dire si il est dans une sous-section, dans une section, dans un chapitre ou dans le texte tout court.
oan:exposeArticle	oan:Article	xsd:string	La propriété « exposeArticle » représente un petit résumé de l'article.
oan:texteLoiCorrespondantArticle	oan:Article	oan:TexteLoi	La propriété « texteLoiCorrespondantArticle » représente l'article correspondant au texte de loi.
oan:articleCorrespond	oan:Article	oan:Loi	La propriété "articleCorrespond " représente l'article correspondant à une loi.
oan:numeroArticle	oan:Article	xsd:nonNegativeInteger	La propriété « numeroArticle » représente le numéro d'un article.
oan:titreArticle	oan:Article	xsd:string	La propriété « titreArticle » représente le titre d'un article.

oan:dansTexte	oan:PositionArticle	oan:TexteLoi	La propriété « dansTexte » indique dans quelle partie du texte de loi se trouve l'article. Car il est possible qu'un article ne soit contenu ni dans un chapitre, ni dans une section et ni dans une sous-section, il est alors en général en tout début du texte de loi.
oan:dansChapitre	oan:PositionArticle	oan:Chapitre	La propriété « dansChapitre » indique dans quel chapitre du texte de loi se trouve l'article.
oan:dansSection	oan:PositionArticle	oan:Section	La propriété « dansSection » indique dans quel section du texte de loi se trouve l'article.
oan:dansSousSection	oan:PositionArticle	oan:SousSection	La propriété « dansSousSection » indique dans quel sous-section du texte de loi se trouve l'article.
oan:aPourTexte	oan:Loi	oan:TexteLoi	La propriété « aPourTexte » représente la loi correspondant au texte de loi.
oan:texteContient	oan:TexteLoi	oan:Chapitre	La propriété « texteContient » représente les différents chapitres contenu dans un texte de loi.

oan:titreChapitre	oan:Chapitre	xsd:string	La propriété « titrechapitre » représente le titre d'un chapitre contenu dans un texte de loi.
oan:chapitreContient	oan:Chapitre	oan:Section	La propriété « chapitreContient » représente les différentes sections qui sont contenus dans un chapitre.
oan:nbIntervention	oan:Section	xsd:nonNegativeInteger	La propriété « nbIntervention » représente le nombre d'interventions qu'il y a eu pour une section.
oan:titreSection	oan:Section	xsd:string	La propriété « titreSection » représente le titre d'une section contenu dans un chapitre d'un texte de loi.
oan:sectionContient	oan:Section	oan:SousSection	La propriété « sectionContient » représente les différentes sous sections qui sont contenus dans une section.
oan:titreSousSection	oan:SousSection	xsd:string	La propriété « titreSousSection » représente le titre d'une sous-section contenu dans une section d'un chapitre d'un texte de loi.

oan:amendementConcerne	oan:Amendement	oan:Article	La propriété « amendementConcerne » représente l'article pour lequel a été créé l'amendement.
oan:statutAmendement	oan:Amendement	{ « 1ère lecture » , « 2ème lecture » , « adopté » , « projet » }	La propriété « statutAmendement » représente les status que peut prendre un amendement durant sa période de discussion.
oan:auteurAmendement	oan:Amendement	oan:Depute	La propriété « auteurAmendement » représente le député qui est l'origine de l'amendement.
oan:amendementNbFoisRectifier	oan:Amendement	xsd:nonNegativeInteger	La propriété « amendementNbFoisRectifier » représente le nombre de fois qu'un amendement a été rectifié par l'assemblée nationale.
oan:dateCreationAmendement	oan:Amendement	xsd:date	La propriété « dateCreationAmendement » représente la date à laquelle est créé l'amendement.
oan:legislatureAmendement	oan:Amendement	xsd:nonNegativeInteger	La propriété « legislatureAmendement » représente la législature sous laquelle a été créé l'amendement.

oan:texteAmendement	oan:Amendement	xsd:string	La propriété « texteAmendement » représente le texte de l'amendement.
oan:exposeAmendement	oan:Amendement	xsd:string	La propriété « exposeAmendement » représente un petit texte résumé de cet amendement.
oan:numeroAmendement	oan:Amendement	xsd:nonNegativeInteger	La propriété « numeroAmendement » représente le numéro donné à un amendement.
oan:amendementCorrespond	oan:Amendement	oan:TexteLoi	La propriété « amendementCorrespond » représente le texte de loi correspondant à un amendement.

4.3 Étape 3 : publication de l'ontologie

Maintenant que l'ontologie est conçue et développée, il faut la publier. Deux outils permettent de faire ce travail : Neologism[3] et LODE[4].

En ce qui concerne LODE, il n'y a strictement aucune installation, puisque c'est un service en ligne. Il suffit juste d'indiquer les informations nécessaires (auteur, date création, résumé, etc.) et commenter correctement son ontologie et LODE affiche le tout dans une interface très lisible. Cet outil ne sert qu'à l'affichage de la documentation de l'ontologie pour voir sa structure et ce qui la compose. Malgré sa simplicité d'utilisation, on a tout de même rencontré un petit souci avec LODE, puisque celui-ci ne fonctionnait pas avec l'ontologie. C'est d'ailleurs pour cette raison que M. Scharffe a envoyé un mail au développeur de cette application afin de savoir pourquoi notre ontologie générait une exception. Le développeur nous a répondu que c'était un problème dans son application et a très vite résolu le problème, ce qui a permis d'utiliser correctement LODE avec notre ontologie. Que vous pouvez trouver à cette adresse[5].

[3] http://neologism.deri.ie/
[4] http://lode.sourceforge.net/
[5] http://speroni.web.cs.unibo.it/cgi-bin/lode/req.py?req=http://data.lirmm.fr/ontologies/files/oan/

L'autre outil, Neologism, lui, est différent. Il nécessite au préalable une installation sur un serveur Web puisque c'est un dérivé du CMS Drupal. C'est d'ailleurs une des raisons pour lesquelles M.Scharffe a ouvert un serveur à l'adresse http://data.lirmm.fr, pour publier des ontologies sous Neologism. D'ailleurs le site de publication d'ontologie est accessible à l'adresse : http://data.lirmm.fr/ontologies. Avant toute chose, l'installation de Neologism nécessite l'installation de :

- Apache, avec le module « URL rewriting » ;
- PHP, en version 5.2 minimum ;
- un SGBD : soit MySQL, soit PostgreSQL.

On a donc installé ces applications sur le serveur, le SGBD choisi étant MySQL[6]. Contrairement à ce que l'on pourrait croire, l'installation de Neologism ne s'est pas déroulée sans mal. En effet, j'ai dû envoyer un mail sur la mailing list de Neologism, l'installation s'arrêtant en plein milieu, disant que le script prenait trop de temps à s'exécuter. La réponse fut que cela venait d'un bogue du script d'installation et qu'il fallait commenter quelques lignes. C'est ce que l'on a fait et Neologism s'est parfaitement installé. Il restait ensuite à importer l'ontologie, tout se fit automatiquement. Comme vous pouvez le voir à cette adresse[7].

4.4 Étape 4 : développement du script de mapping de la base de données

Maintenant que l'ontologie est terminée et publiée sur le Web, il faut faire le lien avec la base de données. C'est dans ce but que l'outil D2R[8] (Database To RDF) et son langage de configuration ont été utilisés. Leur fonctionnalité principale est de transformer notre base de données en triplet RDF en fonction d'ontologies qu'on lui aura indiqué.

Il faut maintenant développer ce fichier de configuration. Il a fallu lire la documentation de D2R pour comprendre comment cela fonctionne. J'ai eu du mal avec cet outil car encore imparfait. En effet, il ne permettait pas de faire tout ce dont on avait envie et besoin. C'est pour cette raison que j'ai beaucoup discuté sur la mailing list avec les auteurs de cet outil. Par exemple, D2R ne permettait pas de faire une requête SQL en spécifiant la valeur d'un champ d'une l'instance courante, mais seulement la requête générale. Pour être plus précis, admettons que l'on veuille afficher le nom d'un député à la place de son « id » mais qu'on est dans une autre table que celle qui liste les députés, celle des présences à titre d'exemple. La table des députés s'appelle « parlementaire » et la table des présences se nomme « presence », la requête général qui permet d'avoir la liste des noms de tous les députés par présence est :

```
SELECT parlementaire.nom FROM parlementaire, presence WHERE
parlementaire.id=presence.parlementaire_id;
```

[6] http://dev.mysql.com/
[7] http://data.lirmm.fr/ontologies/oan/
[8] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

Mais si on veut seulement le nom du député à qui appartient une présence donnée (donc de l'instance courante puisqu'on veut une présence en particulier) et si, dans cette instance, le champ « presence.parlementaire_id » vaut « 347 », la requête précédente se transforme donc en :

```
SELECT parlementaire.nom FROM parlementaire, presence WHERE
parlementaire.id=347;
```

Maintenant on a bien le nom du député correspondant à l'instance courante de présence. Or, dans D2R, il n'y a aucun moyen de remplacer « presence.parlementaire_id » par « 347 ». Par voie de conséquence, comme la première requête renvoie plusieurs lignes, l'application produit une erreur en disant qu'elle reçoit plusieurs lignes et que ce n'est pas normal.

On était donc un peu limité dans les possibilités d'interconnexion des données entre elles. Un autre souci a été rencontré avec cet outil : le fait de pouvoir passer par plusieurs propriétés pour arriver à la donnée que l'on souhaite tout en restant sur la même table. Voici un exemple pour éclaircir le sujet. Dans la table « question_ecrite », un champ, « reponse », qui contient le texte de la réponse à la question. Or, dans l'ontologie, on a une classe « QuestionEcrite » et une classe « Reponse » avec une propriété « reponseCorrespondante » qui les relie. Ce qu'il fallait arriver à faire, c'était de faire correspondre la propriété « texteReponse » de la classe « Reponse » avec le champ « reponse » de la table « question_ecrite », sachant que la table « question_ecrite » correspond à la classe « QuestionEcrite ». J'ai donc posé la question sur la mailing list et l'on m'a répondu qu'il fallait tout simplement créer la classe « Reponse » dans le fichier de mapping qui correspondait à la classe « Reponse » dans l'ontologie et définir une propriété dans le fichier de mapping désignant que ces deux classes comme reliées par la propriété « reponseCorrespondante ». L'application faisait le reste des correspondances toute seule, ce qui a permis de réaliser au mieux ce fichier de mapping.

Un autre souci, anecdotique, correspond au format de certaines données contenues dans la base de données, inexploitables sans passer par des expressions régulières dans un script. Par exemple, le champ « adresses » de la table « parlementaire », qui peut ressembler à :

```
a:3:{i:0;s:64:"Assemblée nationale 126 rue de l'Université 75355 Paris 07
SP";i:1;s:138:"Mairie des 9e et 10e arrondissements 150 Boulevard Paul Claudel 13009
Marseille Téléphone : 04 91 14 63 11 Télécopie : 04 91 14 63 44 ";i:2;s:0:"";}
```

Malheureusement, il est impossible de passer par script puisque D2R établit une liaison directe avec la base de données. Donc deux solutions, soit changer le format, soit manipuler ce format dans une application quelconque.

Après avoir vu l'utilité du mapping, le côté technique. Plus précisément, comment fait-on pour écrire des règles D2RQ ? Il faut savoir que, *grosso modo*, une table correspond à une classe et un champ correspond à une propriété. On peut ensuite choisir les valeurs des propriétés comme étant soit une URI la valeur contenue dans le champ correspondant à la propriété. Maintenant, un petit exemple. Prenons une table « personne » et une classe

« Personne ». La table contient un champ « nom » qui est la clé primaire de la table et qui correspond à la propriété « aCommeNom », dont la valeur une chaîne de caractères (type string). Cela donne :

```
map:personne a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "personne/@@personne.nom@@";
    d2rq:class :Personne;
    d2rq:classDefinitionLabel "Personne";
    .
```

- La première ligne annonce que l'on va mapper la table « personne ».
- La seconde dit que l'on travaille sur l'instance de la base de données mise dans « map:database » ; avant tout, il faut déclarer au début du fichier de mapping à quelle base de données on se connecte.
- La troisième définit une URI qui sera utilisée pour identifier les instances du mapping de cette table, c'est en général la clé primaire de la table.
- La quatrième ligne dit que l'on mappe la table avec la classe « Personne ».
- La cinquième ligne quant à elle annonce le nom de la classe qui s'affichera.

Maintenant, les propriétés :

```
map:personne_nom a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:personne;
    d2rq:property :aCommeNom;
    d2rq:column "personne.nom";
    .
```

- La première ligne dit que l'on crée une propriété qui correspond au champ « nom » de la table « personne » ;
- La seconde ligne dit que la propriété que l'on crée appartient à l'instance de « map:personne » ;
- La troisième dit que le champ « nom » correspond à la propriété « aCommeNom » ;
- La quatrième ligne dit que la propriété aura pour valeur le contenu du champ « nom ».

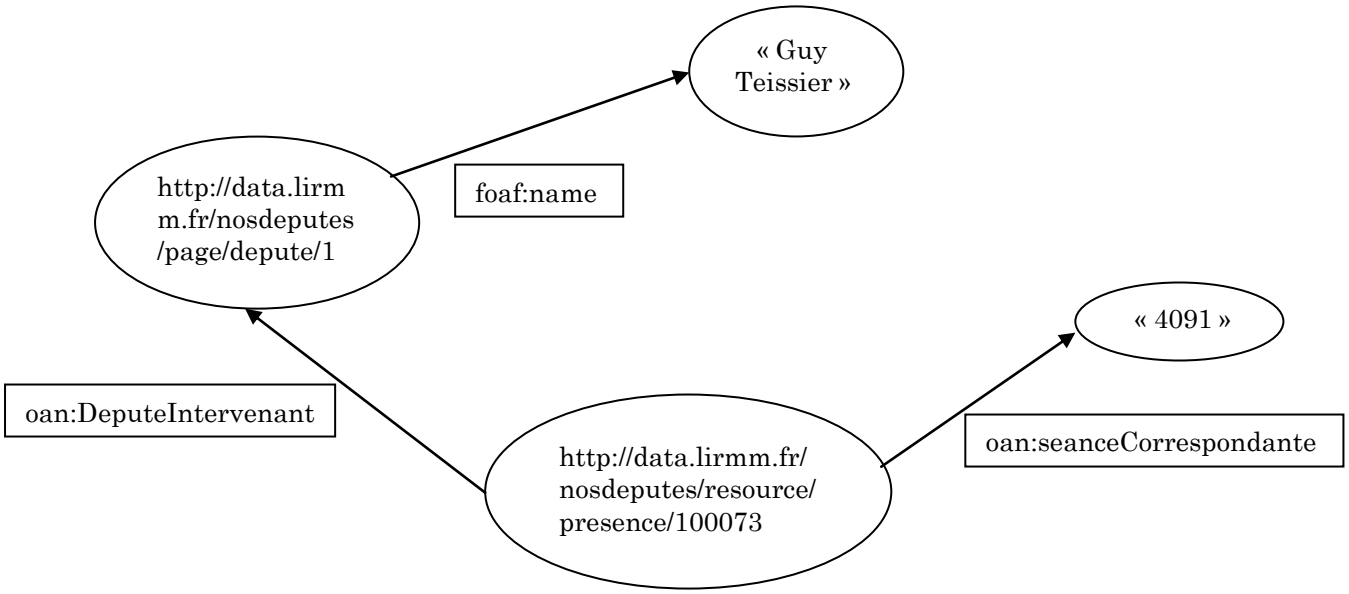
Pour le projet il y a eu une douzaines de règles de constructions pour faire correspondre les tables aux classes. Ainsi qu'une cinquantaines de règles de constructions pour faire correspondre les champs aux propriétés.

4.5 Étape 5 : publication du mapping de la base de données

Maintenant que nous avons notre fichier de configuration, il est temps de l'utiliser pour publier notre base de données en RDF. Mais avant de faire quoi que ce soit avec ce fichier, il

faut tout d'abord installer le serveur D2R sur http://data.lirmm.fr. Pour le serveur, on doit tout d'abord installer un serveur Apache Tomcat, qui écoute sur le port 8080 par défaut. Or, sur le serveur, seul le port 80 est ouvert. Pour éviter de perdre du temps à demander l'ouverture d'un autre port, on a dû se rabattre sur le 80, en interconnectant Apache et Apache Tomcat. Cette fonctionnalité installée, on a pu accéder à D2R sur le serveur du LIRMM[9]. Cette application permet en plus d'offrir un accès pour des requêtes SPARQL avec le protocole http (ce que l'on appelle souvent un SPARQL *endpoint*).

Maintenant que nos données sont accessibles en format RDF, voici un petit exemple d'un graphe RDF afin de montrer comment ces données sont représentées :



4.6 Étape 6 : interconnexion

Cette étape consiste à interconnecter les données, c'est-à-dire à accéder aux données disponibles à d'autres endroit sur le Web de données. Voyez plutôt le schéma se trouvant à cette adresse[10]. Eh bien, Les données sont contenues dans les bulles, les flèches représentant les interconnexions entre ces données. Ainsi, sans interconnecter les données, on ne fait qu'ajouter des bulles au schéma, complètement isolées, on ne peut donc pas accéder aux données contenues dans les autres sources et vice-versa. Pour notre projet nous avons interconnecté nos données avec celles de DBPedia[11]. DBPedia est la version Web sémantique de Wikipédia. Pourquoi DBPedia ? Tout simplement car c'est la source de données la plus conséquente actuellement disponible et que toutes les autres sources de données y sont connectés. Par conséquent, on aura aussi accès à ces autres données.

[9] http://www.pipm.fr/nosdeputes
[10] http://richard.cyganiak.de/2007/10/lod/
[11] http://www.dbpedia.org

Cette étape est celle qui a posé le plus de problèmes. Pourtant, le but est simple, mais, malheureusement, simple ne veut pas forcément dire facile. L'outil utilisé est Silk[12], il permet d'effectuer ces interconnexions. Il sera expliqué plus en détail un peu plus tard. Mais quels ont donc été ces différents problèmes ?

En premier lieu, l'exécution n'était pas possible sous Windows, il a fallu passer sous un système Linux.

Un autre problème l'a suivi : le SPARQL *endpoint* de DBPedia renvoyait toujours des « timeout » aux tentatives de connexions de Silk. Ne sachant pas quoi faire, j'ai fait part de ce problème sur la mailing list de Silk, on m'a répondu que cela venait d'un surplus de connexion au SPARQL *endpoint* de DBPedia et qu'il fallait essayer en boucle jusqu'à ce que cela fonctionne. J'ai donc transféré la réponse à M. Scharffe et il a pris les devants en trouvant une solution. Il a exécuté la requête à la main sur le SPARQL *endpoint* de DBPedia et a importé les résultats sur notre « triplet store » que l'on avait installé spécialement pour accueillir les données retournées par Silk, vous pourrez le trouver à cette adresse[13]. Ainsi, plus besoin d'interroger directement le SPARQL *endpoint* de DBPedia via Silk, puisqu'on peut obtenir le même résultat en interrogeant directement notre « triplet store ». Restait alors à savoir comment on pouvait connaître la correspondance entre les députés de DBPedia et les nôtres, à moins de de le faire à la main, ce qui n'était pas à envisager car beaucoup trop long à réaliser.

Pour cela, Silk a été utilisé. On a lancé l'outil à la fois sur les députés et sur les départements dirigés par les députés. Aussi, vu que DBPedia ne prend en compte que les données qui ont un mapping entre l'ontologie de DBPedia et les *infobox* des pages Wikipédia en anglais, il manquait donc plus de 75 % des députés, puisque rares sont nos députés qui ont une page Wikipédia en anglais. C'est pour cette raison que j'ai procédé moi-même sur le Wiki de DBPedia au mapping entre l'ontologie de DBPedia et le vocabulaire de l'*infobox* des politiciens français. Malheureusement, une fois fait, les outils d'extraction de ces informations n'ont pas fonctionné, il fallait donc attendre qu'une nouvelle release de la base DBPedia soit faite pour que le mapping soit pris en compte. Ainsi, pour le moment, ne sont disponibles que les informations sur les députés qui ont une page Wikipédia en anglais, c'est-à-dire, vraiment pas beaucoup.

Voyons maintenant comment fonctionne et à quoi ressemble la configuration de Silk. Il effectue une requête SPARQL sur chaque SPARQL *endpoint* des stores que l'on veut interconnecter ; une fois les résultats de ces requêtes récupérés sous forme de triplets RDF, Silk effectue une comparaison de triplets en fonction de propriétés que l'on aura définies et met le résultat de ces comparaisons dans un fichier. Il faut savoir que les comparaisons peuvent être plus ou moins strictes, c'est-à-dire que l'on peut donner une valeur pour représenter l'exactitude des résultats, les comparaisons qui ne sont pas suffisamment exactes sont placées dans un fichier différent afin qu'ils soient vérifiés par l'utilisateur. Voici donc un des fichiers de configuration que j'ai utilisé afin d'illustrer ces explications :

[12] http://www4.wiwiss.fu-berlin.de/bizer/silk/
[13] http://data.lirmm.fr/openrdf-sesame

```
<Silk>
 <Prefixes>
  <Prefix  id="rdf"  namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
/>
  <Prefix id="rdfs" namespace="http://www.w3.org/2000/01/rdf-schema#" />
  <Prefix id="owl" namespace="http://www.w3.org/2002/07/owl#" />
  <Prefix id="dbpedia-owl" namespace="http://dbpedia.org/ontology/" />
  <Prefix id="dbpedia-prop" namespace="http://dbpedia.org/property/" />
  <Prefix id="oan" namespace="http://data.lirmm.fr/ontologies/oan/" />
  <Prefix id="foaf" namespace="http://xmlns.com/foaf/0.1/" />
 </Prefixes>
 <DataSources>
  <DataSource id="nosdeputes" type="sparqlEndpoint">
  <Param  name="endpointURI"  value="http://data.lirmm.fr/nosdeputes/sparql"
/>
  </DataSource>
  <DataSource id="dbpedia" type="sparqlEndpoint">
   <Param name="endpointURI" value="http://dbpedia.org/sparql" />
  </DataSource>
 </DataSources>
 <Interlinks>
  <Interlink id="deputes">
   <LinkType>owl:sameAs</LinkType>
   <SourceDataset dataSource="nosdeputes" var="a">
    <RestrictTo>
     ?a rdf:type oan:Depute
    </RestrictTo>
   </SourceDataset>
   <TargetDataset dataSource="dbpedia" var="b">
    <RestrictTo>
     ?b rdf:type foaf:Person
    </RestrictTo>
   </TargetDataset>
   <LinkCondition>
    <Compare metric="levenshtein">
     <TransformInput function="lowerCase">
      <Input path="?a/foaf:name" />
     </TransformInput>
     <TransformInput function="lowerCase">
      <Input path="?b/foaf:name[@lang='en']" />
     </TransformInput>
    </Compare>
   </LinkCondition>

   <Filter threshold="0.2" />

   <Outputs>
    <Output maxConfidence="0.9" type="file" >
```

```
    <Param name="file" value="nosdeputes_dbpedia_dep_verify_links.xml"/>
    <Param name="format" value="alignment"/>
   </Output>
   <Output minConfidence="0.9" type="file">
    <Param name="file" value="nosdeputes_dbpedia_dep_accepted_links.xml"/>
    <Param name="format" value="ntriples"/>
   </Output>
  </Outputs>
 </Interlink>
</Interlinks>
</Silk>
```

- La balise « Prefixes » contient tous les préfixes des ontologies utiles pour les requêtes SPARQL.
- La balise « DataSources » contient les adresses des SPARQL endpoint qui vont être interrogés.
- La balise « Interlinks » contient la balise « LinkType », qui sert à savoir quelle propriété correspond à chaque résultat ; ensuite, « SourceDataset » et « TargetDataset » servent tout deux à écrire la requête SPARQL qui sera envoyé aux SPARQL *endpoint* contenu dans la balise « DataSources », puis « LinkCondition » sert à savoir quelles sont les propriétés qui seront comparées entre elles dans le résultat des requêtes ; quant à la balise « Filter », elle sert à faire des comparaisons plus ou moins strictes : plus la valeur se rapproche de 1 et plus la comparaison sera stricte ; finalement, la balise « Outputs » contient les fichiers qui contiendront les résultats, le premier pour les résultats sûrs et le second pour les résultats hypothétiques.

4.7 Étape 7 : développement d'une application

Maintenant que l'infrastructure adéquate est mise en place, il est maintenant utile de développer une application afin de montrer tout ce que l'on peut faire avec ces informations. Cette application n'a pour but qu'une petite démonstration.

Pour cette application, deux outils :

- GWT[14] 2.3.1 ;
- Jena[15] 2.6.4.

Le choix de GWT (Google Web Toolkit) s'est porté sur plusieurs éléments : je ne le connaissais pas, c'est aussi un outil pour lequel le seul prérequis est de connaître Java, contrairement aux outils similaires, basés sur JEE et JSP (comme Struts 2, par exemple). GWT transforme le code Java côté client en Javascript. Le seul problème : la communication client-serveur pour les traitements non client est lente. GWT est simple à prendre en main

[14] http://code.google.com/intl/fr-FR/webtoolkit/
[15] http://openjena.org/

et s'utilise à la manière de Swing, avec des composants graphiques. De plus, il intègre des API pour Google Maps, Google Docs et d'autres services Google.

Ensuite, Jena pour le côté Web sémantique, car c'est un framework qui a fait ses preuves et qui est très mature, il est énormément utilisée et donc fort documenté.

Ainsi, une fois l'application réalisée, il est possible de lister les députés et d'obtenir toutes les informations se trouvant dans la base de données et dans le « triplet store », sur ce député. On peut aussi visualiser avec Google Maps son département.

D'ailleurs, au sujet de cette visualisation, les informations utilisées ont été obtenues grâce à Geonames[16], une base de données géographiques. Pour l'utiliser, plusieurs solutions : utiliser un dump de leur base de données très conséquente ou, plus simplement et sans prendre autant place, utiliser leur API Java. Il suffit de l'utiliser dans l'application et un résultat RDF nous est retourné à chaque requête, informations que l'on peut manipuler avec Jena.

4.8 Installation et administration des serveurs

M. Scharffe ayant installé un serveur au LIRMM spécialement pour le projet (et pour d'autres projets futurs ou parallèles). Il m'a demandé d'installer les différentes applications nécessaires et de les administrer. Cette manœuvre était chronophage mais formative. Les applications installées leur version sur le serveur du LIRMM sont :

- Apache Tomcat[17] 6.0.28 ;
- Apache[18] 2.2.16 ;
- MySQL 5.1.49 ;
- PHP[19] 5.3.3.

Les applications hébergées par Apache Tomcat :

- Sesame[20] 2.3.1 ;
- D2R 0.7.1.

Précisons aussi que le serveur Apache héberge le site de publication de nos ontologies basé sur Neologism.

[16] http://www.geonames.org
[17] http://tomcat.apache.org/
[18] http://httpd.apache.org/
[19] http://www.php.net/
[20] http://www.openrdf.org/

5. Manuel d'utilisation

L'application étant très simple d'utilisation et très intuitive, aidée en cela par le nombre très restreint de fonctionnalités, il est inutile d'écrire un manuel d'utilisation. Il suffit de s'y connecter[21].

6. Impressions personnelles

Ce projet était pour moi très intéressant et surtout enrichissant en expériences. En effet, je suis pour ainsi dire parti de zéro, car je ne connaissais quasiment rien à ce domaine. Le seul lien entre ce projet et les cours de 1ère année de Master d'ingénierie en intelligence artificielle est l'UE (unité d'enseignement) FMIN208 représentation des connaissances, qui parle de la conception d'ontologies mais principalement au niveau conceptuel. Dans cette UE, nous n'avons pas eu l'occasion d'utiliser un des langages prévus les ontologies du Web sémantique.

Ce projet s'avérait donc particulièrement difficile, mais j'étais extrêmement motivé et je me savais capable d'y arriver. Ce projet a duré environ quatre mois, ce qui laissait peu de temps pour en faire un maximum. Il a donc fallut se mettre tout de suite au travail et, ayant quatre autres projets en cours ce fut encore moins facile, mais pas infaisable.

Pour l'organisation, nous y sommes donc allés par étapes pour faciliter la tâche et arriver progressivement à un bon résultat. Avec M. Scharffe, nous nous sommes vus environ toutes les semaines le mercredi après-midi pour discuter de l'avancement du projet et pour m'aider dans les cas où j'avais du mal pour exécuter une tâche.

Néanmoins, durant ce projet, je me suis inscrit à quelques mailing-list de divers outils que j'ai utilisés pour poser des questions et résoudre mes problèmes. La plupart du temps, les réponses venaient des développeurs de ces outils, cela m'a donc permis d'engager un certain dialogue avec eux et d'ainsi avoir un minimum de soutien de leur part. Il faut aussi dire que l'anglais fut omniprésent, que cela soit au niveau de la documentation des outils ou du dialogue avec des non-francophones. Malheureusement, par moment, mes questions sont restées sans réponses et il a fallu que je me débrouille seul, avec plus ou moins de succès, mais toujours soutenu par M. Scharffe.

Les objectifs de ce projet ont été atteints pour la plupart avec plus ou moins de mal. Il a fallu apprendre différents langages et outils comme OWL, RDF, D2R, Silk, Protégé, Neologism... Durant le projet, j'ai dû aussi m'occuper de l'administration et de l'installation de plusieurs outils sur notre serveur au LIRMM. J'ai donc dû aussi apprendre à utiliser tous ces outils comme Apache, Apache Tomcat et MySQL, ce qui me prenait beaucoup de temps, temps que je ne pouvais mettre dans le but réel du projet, mais cela m'a permis de me cultiver et surtout qui a servi à d'autres personnes.

[21] http://www.data.lirmm.fr/semdeputes

7. Perspectives et conclusions

7.1 Perspectives

Ce projet a de nombreuses perspectives et peut aussi contribuer à améliorer des projets existants. Voici une liste non exhaustive des perspectives possibles :

- modifier certains formats de la base de données (comme les adresses (vu plus haut), les mandats et les villes administrées) ;
- modifier le schéma de la base de données ;
- ajouter des fonctionnalités à l'application GWT (réseaux social, commentaire des utilisateurs, anecdotes des députés, etc.) ;
- intégrer ce projet au site de l'assemblée nationale ;
- améliorer l'ontologie de départ pour l'étendre à d'autres domaines de la politique que l'assemblée nationale et détailler plus certaines classes ;
- pourquoi pas, développer une application mobile.

7.2 Conclusions

7.2.1 Fonctionnement de l'application

Dans l'ensemble, cela fonctionne plutôt bien et comme cela était prévu, malgré les nombreuses difficultés dans la base de données choisie (son schéma et certains formats de ses données). Malgré tout, le résultat est très satisfaisant, sachant que rien n'était sûr au départ, vu que je n'avais jamais touché à ce domaine.

Les outils que nous avons utilisés ont été très bien choisis et nous ont permis de faire ce que l'on souhaitait, si ce n'est le petit problème avec D2R et les requêtes SQL. Mais ceci a été pallié grâce à l'application en GWT. On peut aussi dire que notre analyse de départ (l'ontologie) a été bien réalisée, puisqu'elle est assez générique pour que l'on puisse y ajouter des concepts.

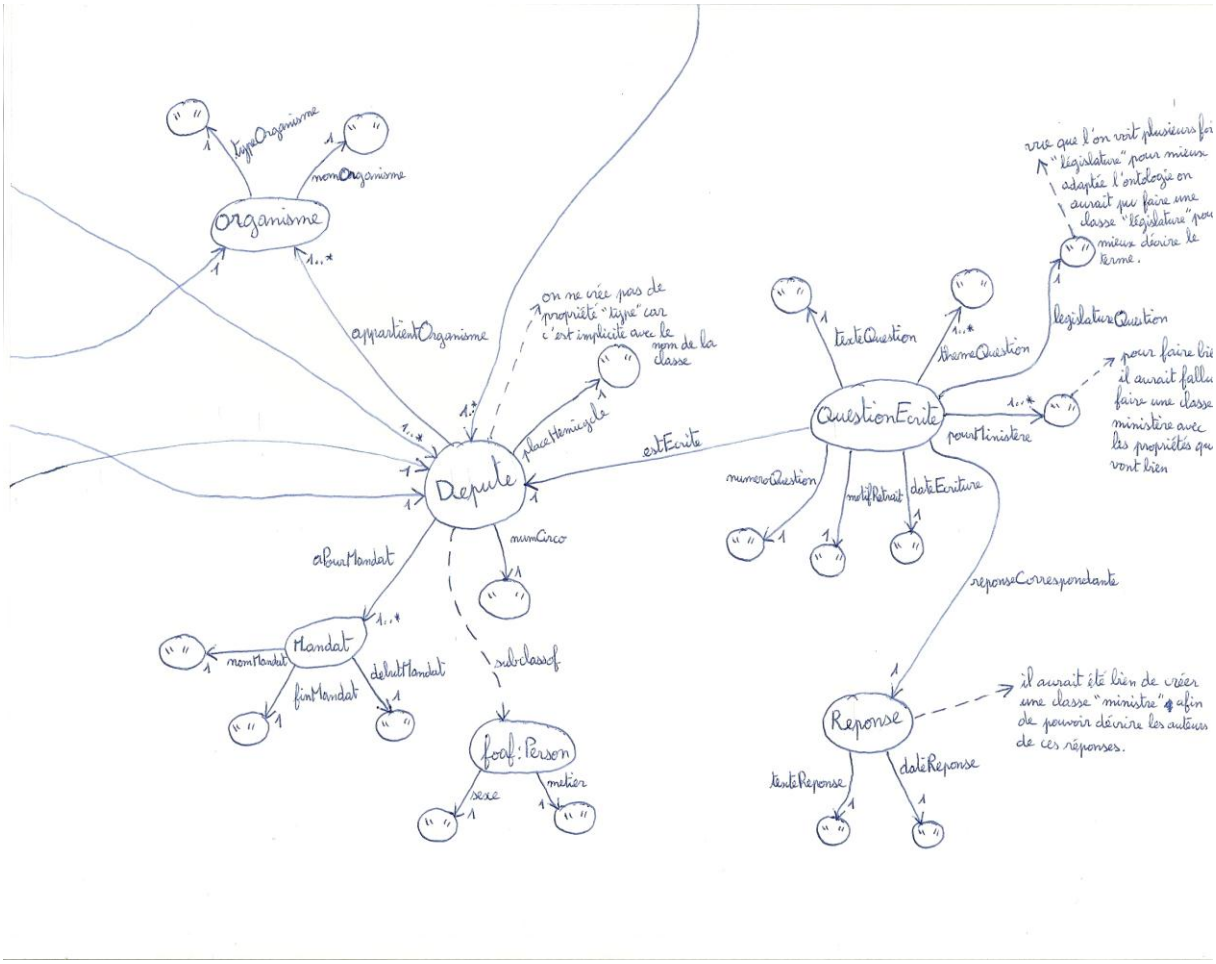
7.2.2 Fonctionnement du groupe de travail

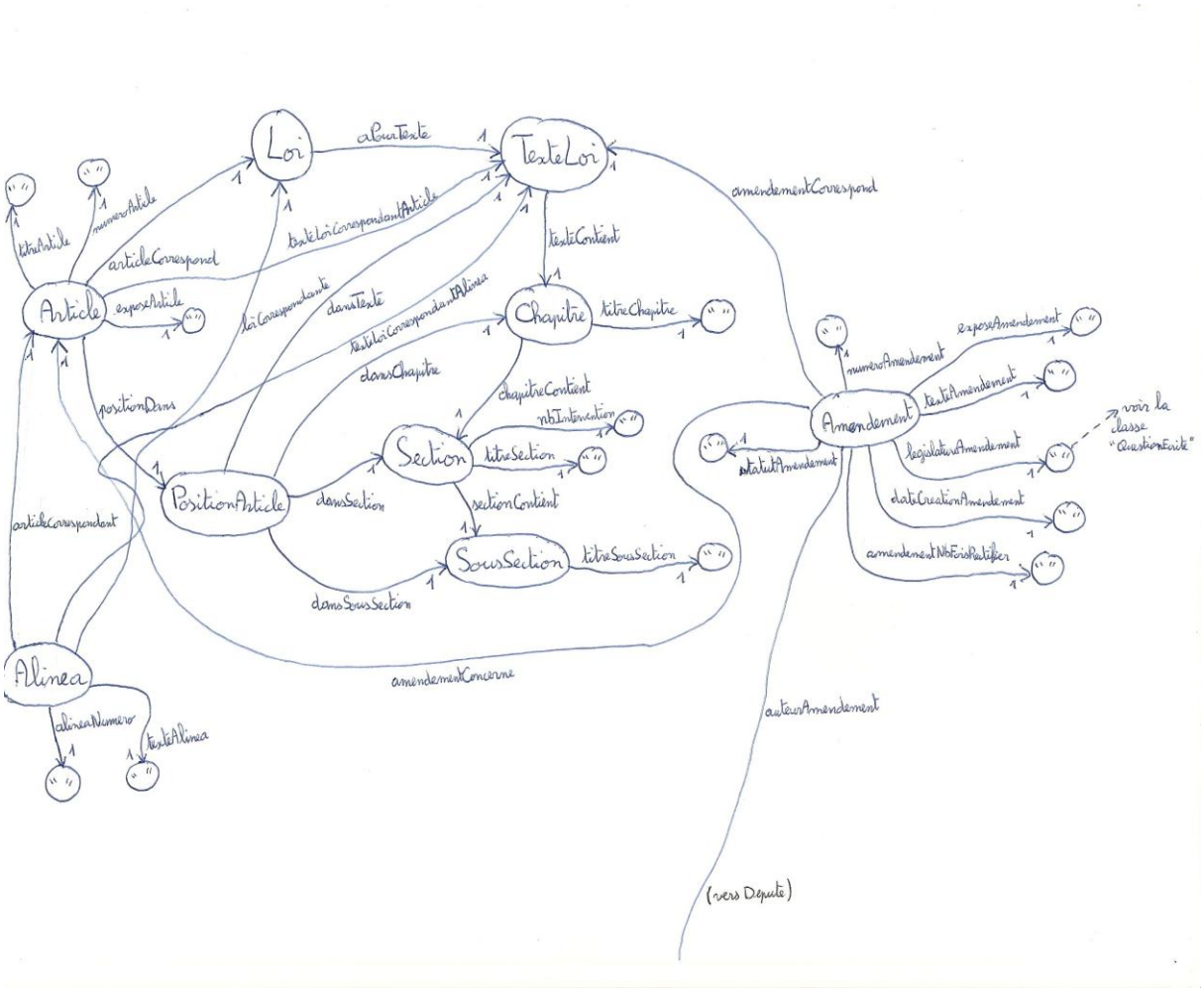
Étant seul sur ce projet, je dois dire que M. Scharffe a toujours été présent pour m'apporter son aide et me supporter. Il faut dire que ce n'était pas facile à chaque fois et j'ai beaucoup appris durant ce projet à discuter par mail autant avec mon encadrant qu'avec des étrangers qui ne parlent pas français. Il aurait été bien d'avoir un peu plus de temps pour ce projet afin de finir ce que l'on avait réellement en tête, comme plus d'interconnexions avec d'autres sources de données.

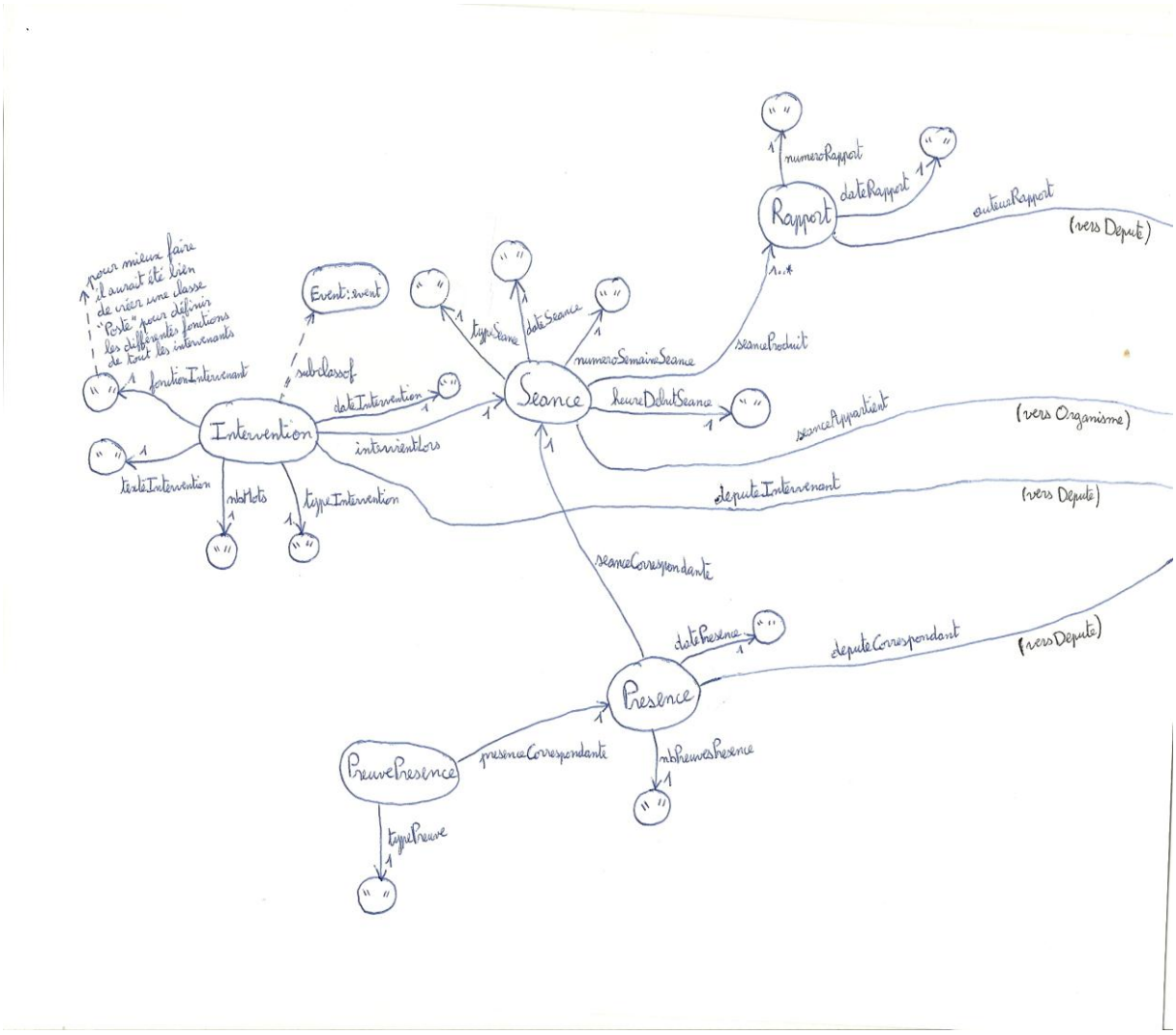
Je dois bien avouer aussi que ce projet nous a causé beaucoup de difficultés, aucune n'était insurmontable : il y avait toujours moyen de faire autrement, même si il est vrai que cela faisait perdre un peu de temps. Je peux donc dire que j'ai très bien su travailler avec mon encadrant et tout seul pour arriver à ce travail bien satisfaisant.

8. Annexe : documents d'analyse

8.1 Scan de l'ontologie OAN




Loi
aPourTexte
TexteLoi
amendementCorrespond
numeroArticle
texteArticle
articleCorrespond
texteLoiCorrespondantArticle
texteContient
Article
exposeArticle
loiCorrespondante
dansTexte
texteLoiCorrespondantAlinea
Chapitre
titreChapitre
dansChapitre
chapitreContient
positionDans
nbIntervention
Section
titreSection
PositionArticle
dansSection
sectionContient
SousSection
titreSousSection
dansSousSection
articleCorrespondant
Alinea
amendementConcerne
alineaNumero
texteAlinea
Amendement
numeroAmendement
exposeAmendement
texteAmendement
legislatureAmendement
voir la classe "QuestionEcrite"
dateCreationAmendement
amendementNbFoisRectifier
statutAmendement
auteurAmendement
(vers Depute)


pour mieux faire
il aurait été bien
de créer une classe
"Poste" pour définir
les différents fonctions
de tout les intervenants
Event:event
subClassOf
fonctionIntervenant
Intervention
dateIntervention
intervientLors
texteIntervention
nbMots
typeIntervention
typeSeance
dateSeance
numeroSemaineSeance
Seance
heureDebutSeance
seanceProduit
1..*
Rapport
numeroRapport
dateRapport
auteursRapport
(vers Depute)
seanceAppartient
(vers Organisme)
deputeIntervenant
(vers Depute)
seanceCorrespondante
datePresence
deputeCorrespondant
(vers Depute)
Presence
PreuvePresence
presenceCorrespondante
nbPreuvesPresence
typePreuve