Table des matières

Table des figures

Glossaire

Evidential Reasoning Service C'est un système cognitif qui identifie différentes preuves 12

Données structurées et semi-structurées Les données semi-structurées sont des informations où les informations sont structurées avec des balises avec une arborescence (comme un format XML). Les données structurées peuvent se résumer à des bases de donnés orientés objets. 17

Linked Data *Linked data* ou Web des données est un moyen d'utiliser le web afin de connecter des données qui n'étaient pas liées précédemment, ou d'utiliser le Web afin de facilier la liaison des données déjà liées avec d'autres méthodes 18

Réseau Bayésien Un réseau bayésien est utilisé en informatique et en statistique. Il fournit un modèle graphique probabilistes représentant des variables aléatoires sous la forme d'un graphe orienté acyclique. 11

Triple Un triple est une entité de donnée représentée au format Sujet-Prédicat-Objet comme "John connait Richard". 17

Visual Analytics *Visual analytics* est l'analyse de raisonnement au travers d'interfaces visuelles intéractives 12

Acronymes

OSINT *Open Source Intelligence* 12

OWL *Web Ontology Language* 17

RDF *Resource Description Network* 17

SPARQL *SPARQL Protocol and RDF Query Language* 17

URI *Uniform Resource Identifier* 17

W3C *World Wide Web Consortium* 17

Introduction

Intelligence analysis (analyse du renseignement en français) est un procédé où une ou plusieurs méthodes cognitives sont utilisées afin de peser des données et où différentes hypothèses sont testées. Ce procédé est généralement utilisé par l'armée pour récolter des informations pour répondre à des questions tactiques sur des opérations en cours ou bien pour prédire des comportements futurs.

Il peut être difficile de donner du sens à des informations contradictoires ou bien incomplètes et de peser le poids des hypothèses se contredisant pour expliquer une situation. Cela impose une certaine demande cognitive aux analystes et peu d'outils automatisés sont à disposition pour les aider dans leurs tâches.

C'est ainsi qu'avec la collaboration entre différentes universités (Université de Southampton, Université de Cardiff et l'université de St Andrews) qu'un outil est mis en place afin d'aider les analystes à obtenir, évaluer et interpréter des informations en collaborant avec d'autres analystes. Assistés par des algorithmes d'intelligence artificiel, les analystes peuvent raisonner avec différents types de preuves pour identifier, pour un contexte donnée, ce qui s'est passé et pourquoi, vérifier la crédibilité et comment obtenir des preuves supplémentaires.

Après avoir introduit l'Université de Cardiff et son département informatique, nous présenterons l'application Web et ma participation au sein de celle-ci au cours de mes douze semaines de stage. Nous passerons par les notions utilisées avant de montrer la structure du projet. Nous consacrerons une partie notamment consacrée à des tests et à la refonte du code. Un manuel d'utilisation est à disposition pour introduire de comprendre générer un rapport en langage naturel à partir d'une analyse donnée. Pour finir, nous présenterons mon rapport d'activité afin de traiter du travail réalisé, mon organisation et des difficultés rencontrées dans la poursuite du projet.

1 Présentation de l'université

Cardiff University ou bien *Prifysgol Caerdydd* en gallois, est une université dans la recherche publique se situant à Cardiff, au Pays de Galles, au Royaume-Uni.

1.1 Histoire

L'université fut fondée le 24 Octobre 1883 et fut formellement établi par la charte royale en 1884. La charte royale étant une charte accordée par le souverain avec l'accord de son conseil privée afin de donner un statut légal) des corporations telles que les cités, les sociétés, ou bien les universités. Initiallement appelée *University College of South Wales and Monmouthshire*, comparée à aujourd'hui, l'université était assez petite. Il n'y avait que :

— 13 personnels universitaires
— 12 départements
— 102 étudiants à plein temps
— 49 étudiants à mi-temps

C'est en 1893 qu'elle devint l'une des institutions fondatrices de l'University of Wales et put enfin donner ses propres diplômes. En 1972, l'université prit le nom de *University College*, Cardiff.

1.2 Fusions

Avant de s'intituler *Cardiff University*, l'université fusionna en 1988 avec *University of Wales Institute of Science and Technology*. C'est en 1999, que son nom public devint *Cardiff University*. En 2004, elle fusionna avec*University of Wales College of Medicine*. Ces deux universités ne faisaient qu'une, jusqu'en 1931, où elles se sont séparées, maintenant elles sont de nouveau réunies.

En décembre 2004, le conseil privé approuva une nouvelle charte additionnelle donnant le statut officiel d'université. Le nom légal changea pour *Cardiff University* et l'université est devenue indépendante de *University of Wales*. Avant 2005, les diplômes étaient de *University of Wales*. Depuis, les étudiants sont diplômés sous *Cardiff University*.

1.3 Armoiries

Le *College of Arms* attribua l'armoirie de l'université en 1988 après la fusion avec *University of Wales Institute of Science and Technology*.

Cette armoiries présente des partisans qui sont généralement rarement attribués aux universités. L'ange et le dragon sont dérivés des timbres des institutions fusionnées.

Le slogan *"Gwirionedd, Undod a Chytgord"* est tiré d'une fin de phrase de la prière pour l'église militante dans le Livre de Prière Commune. Il signifie *"Vérité, Unité et Harmonie"*.

FIGURE 1 – L'armoirie de *Cardiff University*

1.4 School of Computer Science and Informatics

School of Computer Science and Informatics, une des branches de *Cardiff University*, se spécialise dans l'éducation et la recherche en informatique.

1.4.1 Education

L'université propose une formation en informatique au sein de laquelle, les étudiants sont informés sur les différentes recherches en cours à l'Université. Les études se concentrent surtout sur la préparation à l'ingénierie logiciel pour futurs emplois ainsi que l'apprentissage des aspects fondamentaux en informatique tout en proposant différentes spécialités. Des possibilités de stages ou d'études à l'étranger durant un an sont également à disposition.

Avec l'obtention d'une licence, il est possible de se spécialiser en participant dans la recherche, il gratifie l'accès à différents équipements et structures. Des diplômes pluridisciplinaires sont également à disposition. Pour finir, l'accueil d'étudiants internationaux est aussi mis en place au sein de l'université.

1.4.2 La recherche

La recherche prend une grande part au sein de l'université. Elle est organisée en trois groupes :

- **Systèmes complexes** : Spécialisation dans des systèmes de grande envergure, l'analyse de données à l'échelle et les systèmes parallèles et distribués ainsi que l'optimisation à multiple critères et la modélisation mathématique.
- **Ingénierie de données et connaissances** : Spécialisation dans la représentation de connaissances, le *machine learning and data mining* et l'informatique mobile et spatiale.
- **Technologies Visuelles** : Spécialisation dans la vision d'ordinateur, *computer graphics*, *geometric computing* et le traitement d'images/vidéos.

Les domaines de recherches privilégiés sont :

- Les données d'ordre privées et la cybersécurité
- Les systèmes distribués et parallèles
- *Human factors technology*
- Représentation de conaissances et raisonnement
- *Social Computing*
- Extraction de textes et données
- Technologies quantiques et ingénierie

La majorité de ces recherches sont interdisciplinaires en collaboration avec les différents secteurs de l'université ainsi que d'autres universités. On peut trouver des collaborations à travers différents aspects d'ingénierie, la physique, le domaine biomédical, les sciences sociales ou encore l'art. Des partenariats de collaboration inclus les secteurs tels que *Human Factors Technology Centre* et *Wales Institute of Mathematical and Computational Sciences*.

Pour plus de renseignements concernant les différents disciplines et secteurs de recherche, veuillez visiter : `https://www.cardiff.ac.uk/computer-science/research`.

2 Cahier des charges

2.1 Présentation du sujet et analyse du contexte

Les processus de déplacement de données, au travers d'informations et de revendications qui peuvent se compléter ou bien se contredire, jusqu'à la compréhension et la production d'une explication cohérente d'une situation, impliquent de nombreux procédés qui peuvent demander beaucoup de temps et qui comptent sur des experts humains hautement entrainés. Un nombre de méthodes et d'outils a été proposé afin d'automatiser les traitements de données à produire (potentiellement) des informations utiles plus rapidement, ou de soutenir une interprétation d'informations plus rigoureuse où un expert humain peut juger les hypothèses en concurrence.

Des outils commerciaux tels que l'*IBM i2 Analysts Notebook* and *Palantir*, par exemple, fournissent une variété de manières d'automatiser l'extraction de données depuis des sources diverses, et le classement des données visuelles soutient la découverte de relations clés au parmi l'information. Au fur et à mesure que l'analyse avance et que l'expert identifie et formule des hypothèses alternatives, il y a également des outils qui fournissent un soutien automatisé pour leur évaluation. Au sein de l'outil *Xerox PARC ACH tool*, qui sous-tend *ThinkSuite*, par exemple, l'analyste commence avec un ensemble d'informations clés, puis spécifie les hypothèses alternatives qui peuvent expliquer la situation et leurs relations avec l'information acquise. Cette identification d'hypothèses, et la structure d'informations et d'explications alternatives sont utilisées par l'outil afin d'automatiser le traitement du poids des hypothèses en concurrence. Nous réalisons ces procédés en modélisant le problème en un réseau Bayésien, qui propage le poids/certitude donnée à chaque morceau d'évidence par l'expert au travers d'hypothèses faisant preuve d'une rigueur mathématique. Cependant, ces processus dépendent de l'expert humain identifiant toutes les hypothèses possibles, et de donner un poids à toutes les pièces d'évidence.

Aucun de ces outils fournissent un moyen d'enregistrer la relation entre informations et revendications (e.g. se soutiennent, se sapent, ou se contredisent), et soutiennent le procédé de constructions d'hypothèses (i.e. informations et revendications qui sont, à travers certains standards, cohérents). Un aspect plus poussé du procédé de génération de compréhension de données qui sont pauvrement soutenus par les outils existants (voire pas du tout soutenus) est la production finale d'un rapport dans une forme qui satisfait les besoin d'un preneur de décision (*decision-maker*). Les outils offrent, par exemple, des moyens de visualiser des ensembles de données complexes, mettent en avant les liens au sein de l'information acquise, ou de résumer les hypothèses alternatives considérées et leurs probabilités calculées. La production d'un rapport qui couvre les problèmes clés comme la plausibilité, probabilité, faible probabilité d'une hypothèse alternative (justification de l'analyste) ne peut être (semi-)automatisé par des outils existants parce qu'ils ne permettent pas à l'analyste de capturer leur raisonnement.

Collaborative Intelligence Spaces (CISpaces) est un ensemble d'outils et d'algorithmes soutenant le *sensemaking* avec des preuves provenant de différentes sources, et pour la dissémination de rapports en langage naturel. Cet outil fondé sur le web est intégré

avec un moteur d'extracteur de faits, à travers lesquelles les preuves écrites extraites de sources ouvertes peuvent être demandées, et l'information retirée est utilisée au sein d'une analyse. CISpaces facilite la phase centrale du *sensemaking* au coeur du procédé de l'analyse du renseignement dans un format déclaratif. En faisant cela, nous souhaitons que le procédé itératif creuse pour trouver de l'information, le *sensemaking* à travers lequel, l'analyse de structure augmente en incrémentant une poignée d'information, des fichiers de preuves, jusqu'à l'identification et l'évaluation d'hypothèses alternatives. Nous voyons cela comme complémentaire à la fonctionnalité offerte par les outils en cours :*Visual Analystics* afin de soutenir le rassemblement de renseignement et la génération de preuve initiale ; et l'évaluation d'hypothèses via le raisonnement Bayésien. Les méthodes et algorithmes proposés ici remplissent l'écart de soutien, complétant l'expertise de l'analyste en liant les informations afin de former des hypothèses. Ce procédé est, cependant, résistant à l'automatisation dû à un effort de l'ingénierie de connaissances significatif nécessaire pour traiter les données et les modèles de raisonnement. La solution proposée est d'exploiter à la fois les capacité de raisonnements humains et automatisés à travers une coopération homme-machine.

Fondée sur une approche sur la théorie de l'argumentation , cela permet de procurer un*computational model of reasoning* proche au preneur de décision humain. La construction se réalise à partir des schémas d'argumentations, qui sont des modèles d'inférence défendable pour structurer des preuves et inférences développé au travers d'analyses de l'argumentation humaine extensive tels que la loi. De support machine au raisonnement humain est procuré à travers un ensemble de capacités *Reasoning-as-a-Service* : un service d'extraction de fait à l'intégration OSINT ; un *Evidential Reasoning Service*, fondé sur la théorie de l'argumentation, afin d'identifier des hypothèses alternatives cohérentes ; un service d'origine qui enregistre des méta-données sur la création d'analyses ; et un service d'explication pour produire des rapports concis qui peuvent être utilisé pour des briefings ou des analyses collaboratives. L'interface procure un espace pour les analystes afin de tracer les inférences sous la forme d'arguments avec des prémisses (informations ou hypothèses) et des conclusions, et met en avant les conflits. Avec cette approche, CISpaces permet l'exploration de l'expertise de l'analyste, leurs compétences d'analyses et leurs capacité de comprendre la situation dans son contexte.

Le projet se projet se focalise sur 3 éléments clés qui sous-tendent le kit d'outils CISpaces : l'extraction d'information depuis des sources ouvertes ; la structure de preuves et l'identification d'hypothèses ; et la génération de rapport automatisés.

2.2 Analyse des besoins fonctionnels

Notre système doit être conçu afin d'aider les activités de routines des analystes. CISpaces vise à délivrer les fonctionnalités suivantes :

- Production en quasi temps réel de rapports intelligents, et de données clés et extraction d'événements, maintenant l'agilité face à l'émergence de vocabulaire et noms/lieux d'intérêts.
- Analyse graphique d'hypothèses à partir d'informations liées, soutenues par une approche efficace pour l'identification automatisée d'hypothèses et d'éva-

luation.
— Génération de rapport rapide à partir de l'analyse en cours à être délivrée à des échelons plus élevés.

2.3 Analyse des besoins non-fonctionnels

Pour permettre d'interagir avec notre outil pour une collaboration proche entre homme-machine, CISpaces a été conçu à travers une analyse extensive des prototypes précédents et l'orientation des experts sur les sujets en questions. Ce processus de conception a résulté avec les besoins utilisateurs et systèmes principaux suivants :

1. Adoptabilité : la frontière à l'adoption est un fardeau dans le déploiement. CISpaces est facile d'accès à travers un une interface web standard, comme un système de *cloud* en ligne. Cela permet aussi de minimiser les coûts de maintenance et une gestion aisée de l'authentification et du contrôle d'accès.
2. Utilisabilité : L'outil maintient une conception intuitive, qui facilite la compréhension de l'architecture du système (fondée sur des approches familières de gestion de document) et la navigation de l'outil à travers une interaction en *drag-and-drop*.
3. Adaptabilité : L'outil répond efficacement face à une augmentation du nombre d'utilisateurs (facilitée avec l'architecture web), de documents et sources d'information en cours, et la taille de l'analyse.
4. Cohérence : Notre système répond aux actions de multiples utilisateurs qui peuvent résulter en conflits et devrait être capable de récupérer d'une défaillance client/moteur de recherche en un état d'analyse cohérent.
5. Ouverture : Nous visons à ce que les composants de CISpaces soient facilement accessible aux autres systèmes, à travers une sortie du projet en open source, et que par l'utilisation de standards ouverts pour l'échange de données.

3 Rapport technique

Cette partie se consacre aux différentes notions utilisées afin de pouvoir comprendre les outils utilisés en conception. Le travail se centre sur la génération du rapport à partir de l'analyse sous forme de graphe présente dans l'espace de travail. Cette partie est écrite en code Java sous le nom de NLG.

3.1 Notions utilisées

Avant de rentrer dans le vif du rapport technique, il est nécessaire d'expliquer quelques notions utilisées afin de pouvoir comprendre la représentation du graphe et également comment le rapport est généré.

3.1.1 Théorie de l'argumentation

La théorie de l'argumentation est l'étude interdisciplinaires sur comment des conclusions peuvent être atteintes à travers un raisonnement logique, fondé sur des revendications, des prémisses.

3.1.2 *Dung's argumentation framework*

Dung's AF est une des manières de représenter un ensemble d'arguments.

Definition 1. *Un framework d'argumentation Dung AF est représenté sous la forme d'une paire.*

$\langle A, \rightarrow \rangle$
où A est un ensemble d'arguments, et $\rightarrow$ *est une relation binaire sur A i.e.* $\rightarrow \subseteq A \times A$

Une sémantique est moyen d'identifier des ensembles d'arguments (i.e. extensions) "qui survivent à un conflit ensemble".

Voici une liste, de quelques propriétés sémantiques :

- *Conflict-freeness* : un argument attaquant et un argument attaqué ne peuvent pas rester ensemble
- *Admissibility* : une extension doit être capable de se défendre elle-même
- *Strong-Admissibility* : se défend en n'ayant aucun argument attaqué.
- *Reinstatement* : si un argument est défendu, il doit être pris.
- *I-Maximality* : aucune extension est un sous-ensemble d'une autre.
- *Directionality* : un ensemble d'arguments est affecté seulement par ses ancêtres dans une attaque.

Il existe différents types d'extensions, dans notre cas, les suivants nous intéressent :

- *Complete extension* : Ensemble d'arguments *conflict-free* et où chaque argument défendu est inclus (*Admissibility* et *Reinstatement*).
- *Grounded extension* : *Complete extension* avec le moins d'arguments. *Preferred extension* : *Complete extension* avec le plus d'arguments.

Les différents extensions vont permettre de générer le rapport. En effet, *grounded extension* mettra en avant les différents arguments où ils peuvent être estimés comme admissibles dans chaque hypothèses. *Preferred extension* permettra d'identifier les différentes hypothèses crédibles.

3.1.3 *Labellings*

Labellings ou Étiquetage permet de réprésenter la sémantique des arguments d'une autre manière :

— Un argument est *IN* si tous ses attaquants sont *OUT*
— Un argument est *OUT* si au moins un de ses attaquants est *IN*
— Sinon il est *UNDEC* (*Undecided*)

Dans nos analyses, cette sémantique permet d'identifier les différents plausibles en fonction des hypothèses. Les hypothèses crédibles possèderont des arguments qui verront leur admissibilité variée en fonction des hypothèses. Dans l'hypothèse sceptique, ces arguments qui varient correspondent à des arguments *UNDEC*.

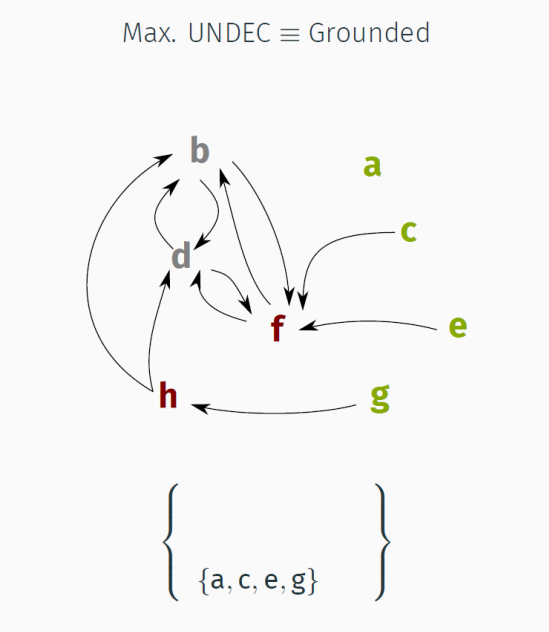

FIGURE 2 – *Grounded extension*

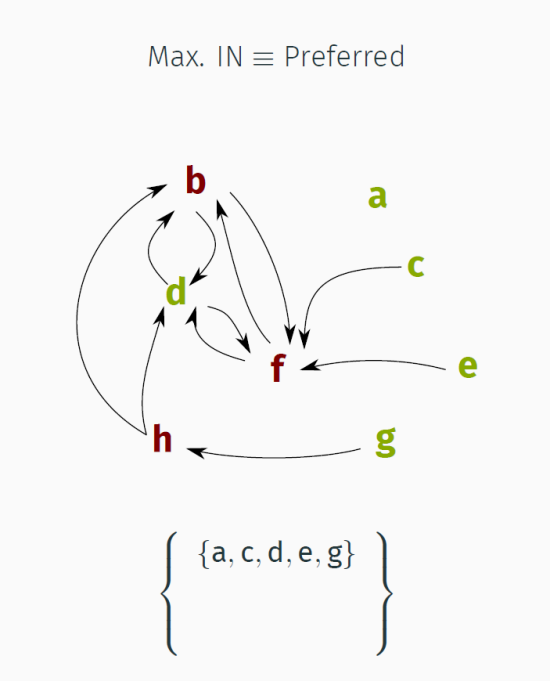

FIGURE 3 – *Preferred extension*

3.1.4 Web Sémantique

Le web sémantique, ou toile sémantique, est une extension du Web standardisée par le W3C. La standardisation encouragent l'utilisation de formats de données et de protocoles normés sur le web, le format de base étant le RDF.

RDF est un modèle standard pour l'échange d'information sur le web. Il possède des fonctionnalités qui permettent la fusions de données mêmes si les schémas sous-jacents diffèrent, et il soutient en particulier l'évolution de schémas au cours du temps sans avoir besoin que les consommateurs de données changent. RDF étend la structure des liaisons du Web en utilisant des URI pour nommer les relations entres les différents sujets,prédicats et objets (ce modèle est référé comme un *triple*). En utilisant ce modèle relativement simple, il permet à des données structurées et semi-structurées d'être mélangées, exposées, et partagées à travers différentes applications. Un *triple* peut être représenté avec un graphe de la manière suivante :

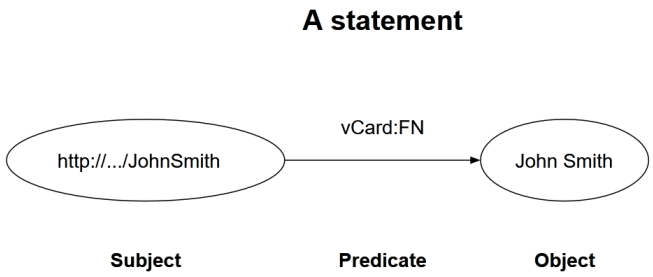

FIGURE 4 – Exemple de déclaration RDF

RDF schema permet d'étendre la représentation des connaissances. Il fournit des éléments de base pour la définition d'ontologies ou vocabulaires destinés à structurer des ressources RDF. Les RDF schemas peuvent être dans des bases relationnels et être manipulés à l'aide des requêtes en langage SPARQL. Dans un RDF Schema , des classes et des propriétés peuvent être définies. Voici les notations utilisées :

- **rdfs:Class** : Permet de déclarer une ressource RDF comme une classe pour d'autres ressources.
  Exemple : ex:Jean rdf:type foaf:Person
- **rdfs:subClassOf** : Permet de définir une hiérarchie de classes.
  Exemple : foaf:Person rdfs:subClassOf foaf:Agent
- **rdfs:domain** : Domaine de définition d'une fonction.
  Exemple : ex:travailledans rdfs:domain foaf:Person
- **rdfs:range** : Ensemble d'arrivée d'une fonction.
  Exemple : ex:travailledans rdfs:range foaf:Organization

OWL est un format qui propose encore plus de sémantique. Les classes et propriétés peuvent être beaucoup plus spécifiques. OWL permet également de définir que deux choses sont la même, cela permet de lier des données exprimées dans des

schémas différents. Par exemple, ”John Titor” sur Wikipedia peut être identifié comme le ”John Titor” sur *BBC*. De cette manière, des données provenant de multiples sites peuvent être liées. OWL permet aussi de déduire des faits comme dans l'exemple suivant :

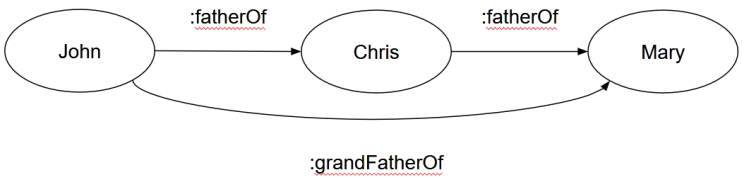

FIGURE 5 – Exemple de conclusion réalisable par transitivité avec OWL

Pour finir, SPARQL est un langage de requête et de protocole permettant la recherche, l'ajout, la modification ou bien la suppression de données RDF disponibles sur internet. Le processus de recherche est similaire au SQL mise à part que l'on accède aux données du *linked data*.

```
SELECT ?x
WHERE { ?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  "John Smith" }
```

FIGURE 6 – Exemple de requête SPARQL

3.2 Conception

L'application web est construit en grande partie en PHP/Javascript, certains algorithmes sont toutefois codés en Java. Cette partie se consacrera en particulier sur le code NLG écrit en Java permettant de récupérer le graphe JSON correspondant au graphe d'analyse pour en générer un rapport en langage naturel.

3.2.1 Choix technologiques

Apache Maven Couramment appelé Maven, Apache Maven est un outil de gestion et d'automatisation de production des projets logiciels Java en général et Java EE en particulier. Cette outil a pour objectif de produire un logiciel à partir de ses sources, en optimisant les tâches réalisées et en garantissant le bon ordre de fabrication. Mave utilise un paradigme sous le nom de *Project Object Model* afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Une des tâches pré-définies notamment utilisé est la compilation du code Java. Maven possède aussi l'aptitude à fonctionner en réseau, de fournir un moyen à synchroniser des projets indépendants.

Apache Jena Apache Jena est un framework Java gratuit et open-source permettant de de construire du web sémantique et du web des données. Ce framework est composé de différentes API interagissant entre elles afin de traiter des données RDF. Ci-dessous, se trouve l'architecture du framework.

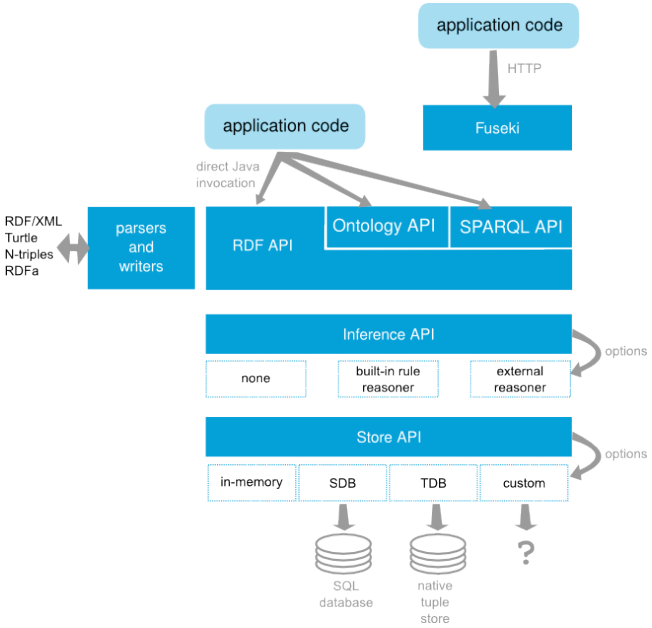

FIGURE 7 – Architecture du framework Jena

A partir d'une ontologie générée, la partie *Inference* permettra à l'utilisateur de définir un certain nombre de règles pour donner plus de sémantique à l'ontologie. En se référant à l'exemple utilisé dans la présentation du OWL, voici une règle : [rule : (:John :fatherOf :Chris)(:Chris :fatherOf :Mary) -> (:John :grandFatherOf :Mary)]

3.2.2 Architecture du programme

La figure suivante permet de visualiser comment le code NLG, derrière la génération de rapport en langage naturel à partir de l'analyse créée sous forme de graphe, est réalisée.

(1) : Sur le site web, l'analyse est représentée sous forme d'un graph. La structure du graphe ainsi que les différentes hypothèses sont conservés sur le site sous la forme d'un graphe JSON.

(2) : Pour récupérer le graphe JSON, NLG.java réalise une requête Ajax sous HTTPS. Le résultat est récupéré dans une méthode POST.

(3) : Une fois le graphe JSON à disposition, NLG créé un objet *ExtensionBulletPoints* avec le graphe sous un format *String*. *ExtensionBulletPoints* convertit le graphe JSON en une ontologie RDF à partir d'*Apache Jena*. L'ontologie permet de conserver les informations liées aux noeuds, comme leur type, la description text et types de *Labellings*. Le code NLG effectue une fonction *getText()* sur l'objet récemment créé afin de réaliser un raisonnement à partir de l'analyse sous forme d'ontologie. A partir des informations contenues dans les nœuds le texte peut être déduit.

(4) : La réponse à la requête est ensuite renvoyé par HTTPS à l'application web pour générer le texte dans l'interface.

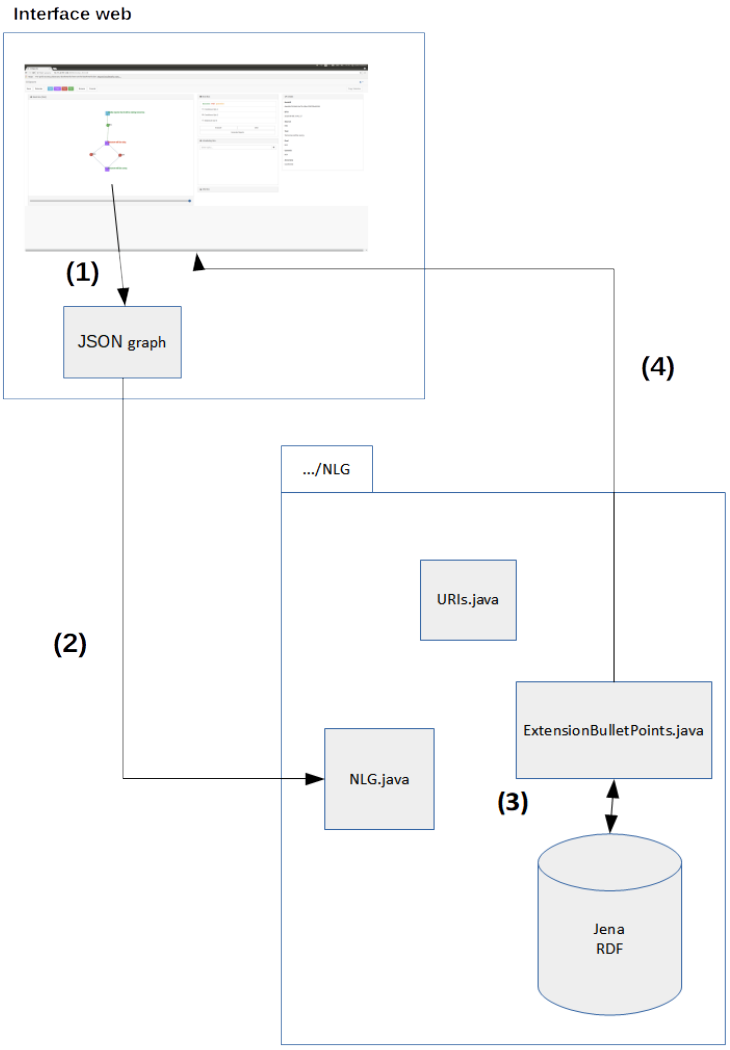

FIGURE 8 – Interactions entre l'application web et le code NLG

3.2.3 Diagramme

Le code NLG peut être représenté avec le diagramme de classe suivant :

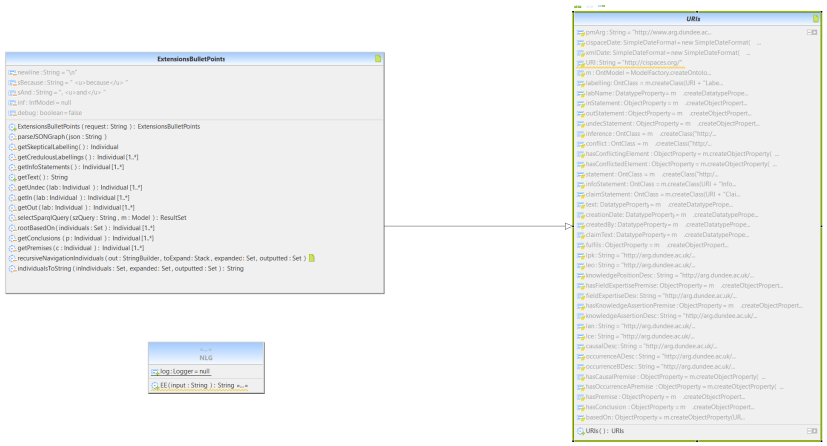

FIGURE 9 – Diagramme de classe du code NLG

Dans ce diagramme, URIs contient toutes les URI nécessaires pour concevoir l'ontologie de l'analyse. La classe NLG récupère le graphe JSON à partir d'une méthode POST dans le NLG pour créer un objet *ExtensionBulletPoints* où est appliqué une méthode getText() qui récupère le rapport en langage naturel généré à partir de l'analyse. *ExtensionBulletPoints* hérite de URIs afin de pouvoir manipuler les URI pour créer une ontologie à partir de l'un graphe JSON.

4 Manuel d'utilisation

Le manuel d'utilisation aura pour objectif de présenter l'utilisation général du site, de la construction d'une nouvelle analyse jusqu'à la génération de rapport. L'extraction de faits ne sera pas traité au sein de cette partie.

4.1 Aperçu général

Une fois le site web utilisable en local, une page d'authentification est affichée. Après avoir rentrés les identifiants, vous pouvez accéder à la page principale proposant les différentes analyses de l'utilisateur. Il est également possible de créer une nouvelle analyse ou d'importer un graphe à travers les boutons situés sur la gauche. Une analyse déja créée peut être visualiser, *Check-out* ou bien supprimée. *Check-out* permet de modifier directement l'analyse. La fonctionnalité *Export* permet d'exporter le fichier au format .cis par le site web. Si on utilise la fonctionnalité *Import*, on pourra récupérer une analyse générée au format .cis.

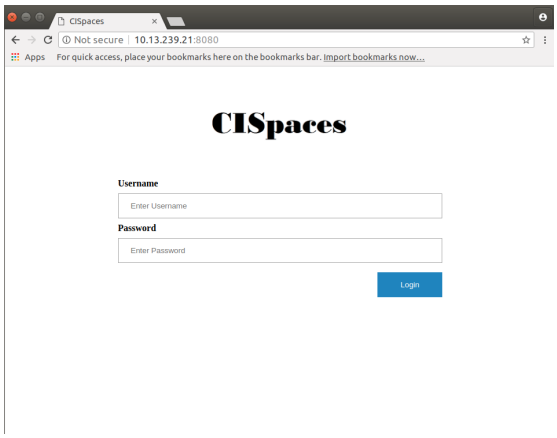

FIGURE 10 – Page d'authentification

4.2 Interface de l'analyse

Lorsqu'une nouvelle analyse est créée, l'interface suivante s'affiche :

La bande se situant en haut de la page sous le titre CISpaces propose différentes fonctionnalités. Cette bande propose la possibilité de sauvegarder l'analyse, *Relocate* (étend la *Work Bow* afin d'y ajouter plus d'éléments), voir toutes les différentes analyse (à travers *Browse*, on retourne à la page principale du site) et la fonctionnalité *Commit*. Une fois l'analyse *commit*, on ne peut plus modifier le graphe, on peut le modifier de nouveau en appuyant sur le bouton *Check-out*. Les boites de couleurs permettent de générer les différents nœuds, leur utilisation sera vu plus en détails dans la partie Générer un graphe. L'espace de travail de l'analyse se partage ensuite en quatre parties :

— **Work Box :** La partie contenant le graphe.

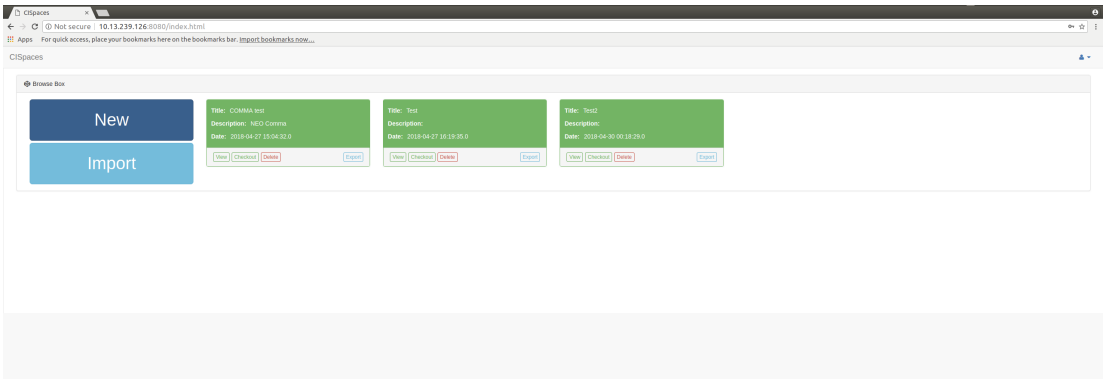

FIGURE 11 – Page Utilisateur

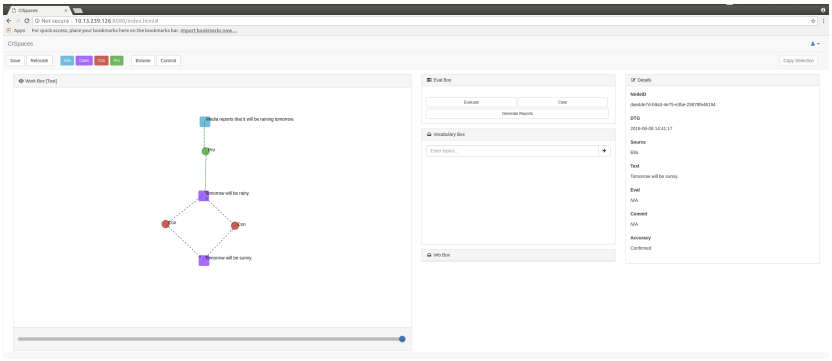

FIGURE 12 – Interface d'analyse

— **Eval Box :** Cette partie permet l'accès aux deux algorithmes principaux pour trouver les différentes hypothèses à partir de l'analyse générée dans la partie *Work Box* ainsi que la possibilité de générer le rapport de l'analyse en langage naturel.
— **Vocabulary and Info Box :** Donne les outils nécessaires pour faire des requêtes et visualiser/importer des résultats à partir de l'extracteurs de faits depuis les réseaux sociaux.
— **Details :** Permet de visualiser les informations liées au nœud sélectionné dans le graphe.

4.3 Générer un graphe

Après la visite de l'espace de travail, la génération de graphe peut débuter.

4.3.1 Placer des noeuds

Comme précédemment présentée dans la partie précédente, des carrés de couleurs dans la barre se situant sous le titre CISpaces, représentent les différents nœuds. Voici à quoi correspond chaque noeud.

— **Info :** Noeud qui correspond à une information.
— **Claim :** Noeud qui correspond à une revendication.
— **Pro :** Noeud qui permet de défendre un argument.
— **Con :** Noeud qui permet d'attaquer un argument.

Les noeuds peuvent être placés dans le *Work Box* à travers un système de drag-and-drop. Une fois un nœud placé, en double-cliquant, nous pouvons changer les informations liées à ce nœud tel que le titre et la description comme ci-dessous.

4.3.2 Créer les arêtes

Une fois, les nœuds placés, des liens peuvent être liés entre eux. Un nœud *Info* ou *Claim* ne peuvent être liés qu'à un un nœud *Pro* ou *Con* et vice-versa. Pour lier deux nœuds entre eux, il faut dans un premier temps réaliser un clic droit sur le premier noeud et sélectionner *Link From...*. Le nœud de départ est enregistré, il faut maintenant sélectionner le nœud d'arrivée. Réalisez ceci sur ce dernier en sélectionnant l'option *Link To...*. Une flèche, plus concrètement, une arête est générée entre les deux noeuds.

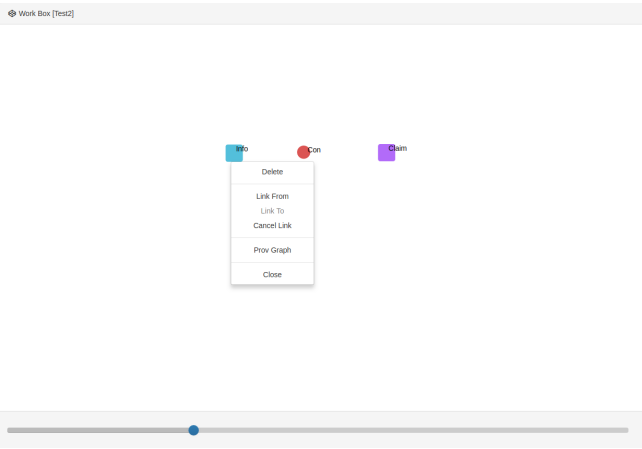

FIGURE 13 – Création du lien sur le nœud de départ

4.4 Evaluer

Si l'analyse en cours peut être évaluée, c'est-à-dire que différentes hypothèses peuvent être générées. Dans la l'*Eval Box*, le bouton *Evaluate* peut être utilisé afin de générer les hypothèses alternatives. Une fois sélectionné, différentes hypothèses s'offrent à disposition de l'utilisateur, elles peuvent être visualisées en sélectionnant

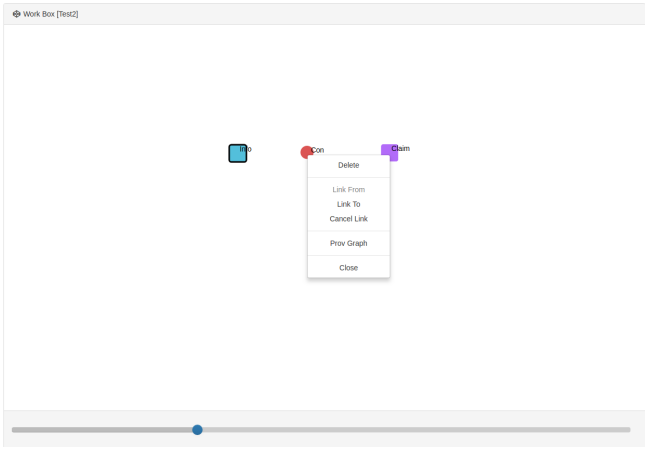

FIGURE 14 – Finir le lien entre les nœuds

les différents boutons radios. Dans une hypothèse , un nœud en vert indique qu'il est admis comme vrai, s'il est rouge, il est admis comme faux, et pour finir, s'il est en jaune (dans l'hypothèse sceptique), c'est un nœud qui peut être soit vrai ou faux au sein d'une analyse. Si l'on souhaite revenir à la vue précédente, il suffit de cliquer sur le bouton *Clear*.

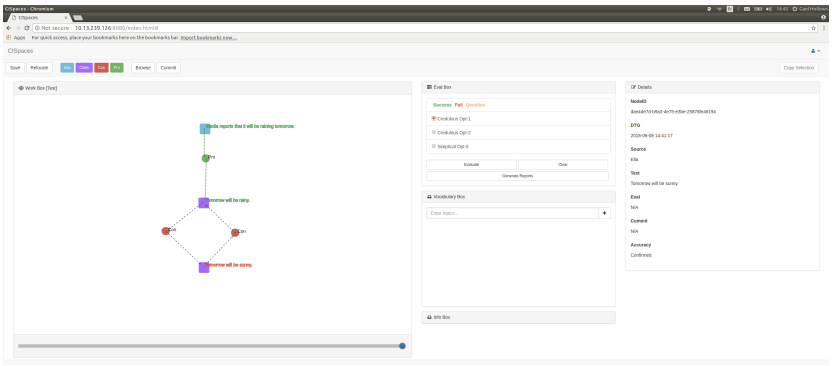

FIGURE 15 – Une hypothèse crédible

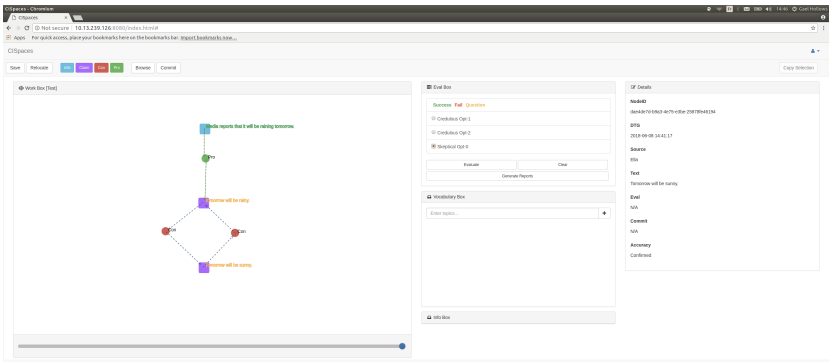

FIGURE 16 – Une hypothèse sceptique

4.5 Générer un rapport

Une autre option se trouve également à disposition dans l'*Eval Box*. *Generate Report* permet de générer un rapport de l'analyse en langage naturel au centre de la page comme ci-dessous.

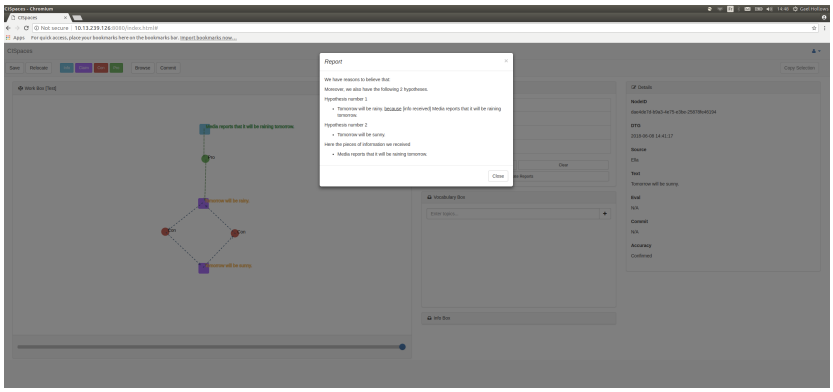

FIGURE 17 – Génération d'un rapport

5 Tests et Validation

Cette partie permet de visualiser la méthodologie et le raisonnement mis en avant dans le développement du projet afin de trouver des bugs potentiels.

5.1 Examen des résultats

Les tests vont porter sur la procédure de génération de rapport à partir du graphe JSON générée. Avant de passer par des tests unitaires, il est nécessaire de procéder à quelques tests fonctionnels pour vérifier la possible présence de bugs.

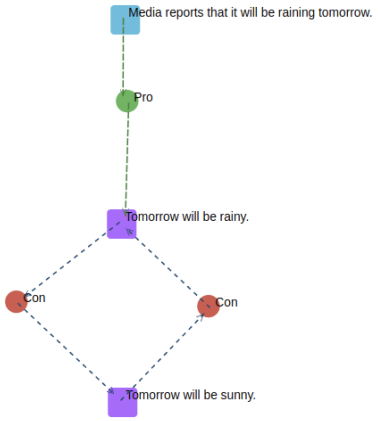

FIGURE 18 – Analyse d'une prévision météo

Premier exemple Dans un premier temps, l'exemple prit sera relativement simple. Le graphe visible dans le *Work Box* présente différents noeuds avec une information "Les médias disent qu'il va pleuvoir demain" et deux revendications se contredisant. Par extension, deux hypothèses sont possibles : "Il va pleuvoir demain" ou "Il fera soleil demain". La première hypothèse est soutenu par l'information qu'il va pleuvoir d'après les médias. Lors de la génération du rapport, les 2 hypothèses sont retrouvées ainsi que la liste d'informations à dispositions (dans le cas présent, une seule). Jusqu'à présent, aucune erreur visible.

Second exemple Dans le cadre des tests, un exemple comprenant plus de noeuds et de relations sera utilisé. Pour rapidement mettre dans le contexte de l'exemple, un virus est en train de se propager dans une ville, cela peut donc amener différentes conséquences telles que des évacuations ou des émeutes face au décisions prises par la ville. Dans la ville, il s'y trouve un port et un aéroport. Afin de parvenir à l'aéroport il est nécéssaire de passer par un tunnel. Le fait est, les émeutes ou les évacuations

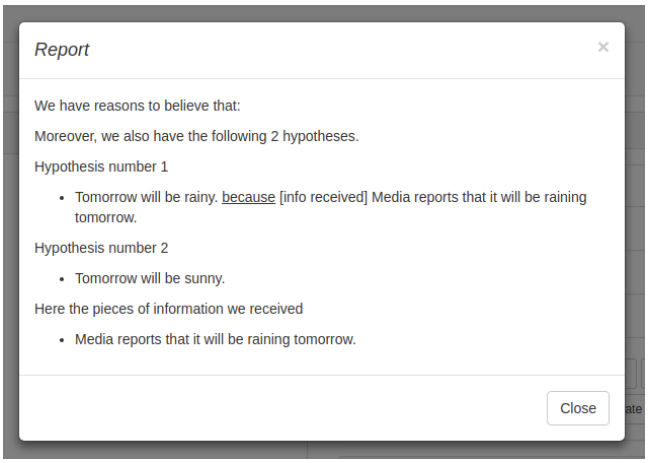

FIGURE 19 – Analyse d'une prévision météo : rapport

peuvent provoquer le bloquage de ce passage. Ci-dessous se trouve le graphe représentant la situation.

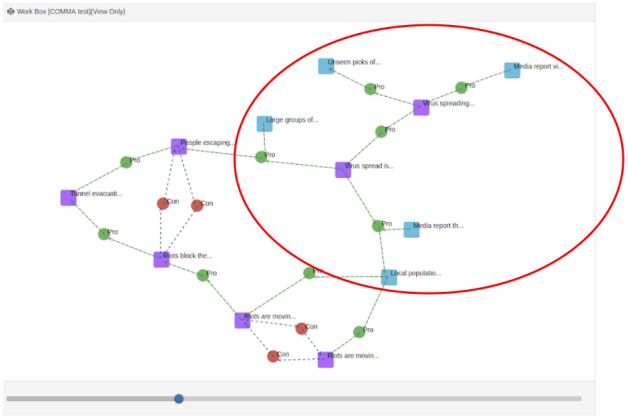

FIGURE 20 – Analyse de l'évacuation de Squirrel City

La partie entouré représente la *grounded extension*, c'est-à-dire les arguments admissibles comme étant vrais pour toute hypothèse.

Voici l'exemple de rapport qu'il souhaité d'avoir et celui reçu à partir du code actuel :

Des erreurs se sont donc produites lorsque nous comparons les parties *"We have reasons to believe that"*. Des parties supplémentaires ont été générées. Des informations en tant que conclusions ont été générées ainsi qu'un duplicat de conclusion s'est produit. Une information peut être soutenu par un argument, mais celle-ci ne nécessite pas de soutien contrairement à une revendication. Une information est déclarée,

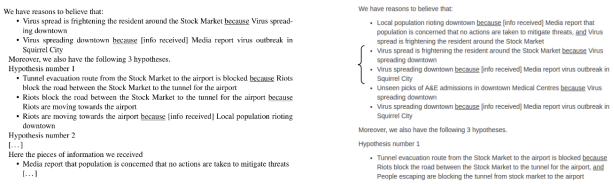

FIGURE 21 – Rapport sur Squirrel City : Résultat attendu et généré

elle est admise comme vraie, tandis qu'une revendication doit être soutenu par un argument afin de pouvoir l'admettre comme vraie ou fausse.

Maintenant que des erreurs ont été trouvées, il faut réaliser des tests unitaires pour trouver la source du problème. Avant toute chose, une des sources des bugs doit se situer dans la génération des conclusions à partir du graphe. Dans un premier temps, la transition depuis le graphe du *Work Box* au graphe JSON doit être vérifiée. En utilisant différents logs, le graphe JSON pourra être récolté. Une fois, le graphe JSON en main, afin de faciliter sa lecture, des outils sur internet tel que `https://jsonformatter.org/` permettent l'indentation d'un graphe JSON. En comparant les deux graphes, aucune erreur semble apparaître. Maintenant, il faut vérifier la transition entre le graphe JSON et l'ontologie générée. Pour permettre une meilleure lecture de l'ontologie générée, Protégé sera utilisée afin de mieux visualiser la construction de l'ontologie.

Dans la construction de l'ontologie, une seule règle est utilisée pour générer des inférences.

La règle indique que si notre nœud *Conflict* ou Pro possède une prémisse et une conclusion, une relation *basedOn* est établie. Cette règle permet de vérifier par quelles informations ou revendications, un argument est-il soutenu. Les limites de cette règle indiquent que l'argument soutenu n'est pas déterminé, est-ce une information ou une revendication soutenue. Dans le cas présent, il faudrait avoir une revendication en tant que conclusion. Il est possible qu'un changement au niveau des règles doit être établi.

Avec l'ajout d'une règle qui complète la première règle, celle-ci vérifie en plus de la règle *basedOn* que la conclusion soit une revendication. Maintenant, il faut analyser les différentes erreurs possibles lors de l'extraction à partir de l'ontologie. La fonction *rootBasedOn()* par exemple, doit récupérer les conclusions à partir des *preferred extensions*. C'est ici qu'un bug semble se produire, les individus de l'ontologie récupérés permettent de voir qu'il ne s'agit pas des conclusions recherchées. Dans le test effectué, des conclusions en tant qu'informations ont été relevées. Dans la fonction *recursiveNavigationIndividuals*, à partir des conclusions de *rootBasedOn*, les prémisses sont déduites. Ceci explique la génération d'informations en tant que conclusions dans le rapport générée. A travers les logs, la duplication de conclusions semblent être causée par la gestion de pile des prémisses lors des appels récursifs.

5.2 Suites possibles de développement

A partir des erreurs détectées dans les logs grâce aux différents tests, il est possible d'envisager différentes refontes dans le code. La mise en place d'une nouvelle règle pour récupérer des conclusions en tant que revendications devrait corriger quelques erreurs. Après refonte de celle-ci, la fonction *rootBasedOn* doit être réécrite afin de collecter les conclusions correspondant à la *preferred extension*. Aux dernières modifications, ces erreurs ont été corrigées. Lors des tests fonctionnels, de nouvelles erreurs semblent apparaitre dans la génération d'hypothèses. La refonte doit maintenant être focalisé sur la fonction *recursiveNavigationIndividuals* afin que l'extraction de prémisses et d'hypothèses se déroulent correctement.

```
@include <OWL>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix cis: <http://cispaces.org/> .
@prefix dund: <http://arg.dundee.ac.uk/aif#>.
[rule1: (?ra dund:hasPremise ?p) (?ra dund:hasConclusion ?c) -> (?c cis:basedOn ?p)]

[rule2: (?c cis:basedOn ?p) (?c rdf:type cis:claimStatement) -> (?c cis:reasonTo ?p)]
```

FIGURE 22 – Nouvelle règle pour éviter une information en tant que conclusion

FIGURE 23 – Rapport généré après refonte d'une partie du code

6 Rapport d'activité

Cette partie se concentre sur la gestion du projet durant la période de stage. Elle retracera également les différentes étapes de développement ainsi que son organisation
.

6.1 La planification, les méthodes de développement et de travail

Le projet étant assez complexe à comprendre dans son intégralité. Seule une portion, nous intéresse dans le développement. Mon travail consistant surtout à appliquer mes connaissances en tests et écriture de code en Java. Des connaissances mathématiques ont dû être appliquées. Les graphes et l'algèbre de Boole sont nécessaires afin de comprendre les notions d'argumentations utilisés. Le travail demandé consiste à appliquer des méthodes d'ingénierie logiciel afin de tester et faire une refonte le code Java permettant la génération du rapport. Des notions autour du web sémantique sont aussi nécessaires pour gérer des données en *triple store* au sein d'une ontologie. Il faut donc prendre en compte une phase d'apprentissage avant le commencement des tests et de la refonte du code pour corriger les erreurs.

Pour résumer, voici les objectifs principaux du stage :
— Refonte du code NLG pour CISpaces.org
— Compréhension général de la théorie de l'argumentation avec la représentation de systèmes de raisonnement.
— Méthodes d'enquêtes de recherche à appréhender

6.2 Planification

Pour pouvoir gérer du développement durant douze semaines, un planning a été établi :

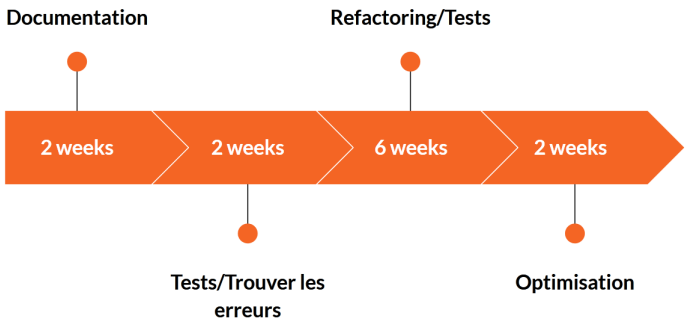

FIGURE 24 – Planning estimé

Ce planning permet d'avoir un aperçu sur l'estimation du développement du projet et n'est pas définitif.

6.3 Méthodes et outils de travail

6.3.1 Organisation et communication

Le planning servant de base au développement, des méthodes agiles peuvent toujours être mises en place. Afin de suivre le développement du projet, des rendez-vous hebdomadaires ont été instaurés. Au cours de ceux-ci, un certain nombre de tâches est défini. Ceci permet de définir des sprints d'une semaine où des tâches doivent être réalisées. Au rendez-vous, un briefing sur les tâches réalisées où des difficultés rencontrées ainsi que des questions sont émises afin de permettre un meilleur avancement du projet. Une fois le briefing réalisée, le prochain sprint peut être organisé. Pour permettre également un meilleur suivi du projet ainsi que de sa compréhension, je peux contacter mon tuteur avant le *weekly meeting* pour des informations complémentaire ou d'une aide sur un problème qui empêche l'avancement d'une ou plusieurs tâches. Au cours du début de mois de juin, une présentation a été réalisée afin de mettre en avant mon travail au sein du projet devant différents personnes intéressées dans son développement et son intérêt.

6.3.2 Outils de travails

L'application web a été développée en Java sous Unix. Avec les connaissances nécessaires pour développer sur ce système ainsi que l'expérience en Programmation Orienté Objet, le développement nécessitait majoritairement la familiarisation avec le framework Jena.

Dans le développement, git sera utilisée afin de créer une nouvelle branche au projet pour permettre le développement sans toucher à la branche principale. Une fois, le développement terminé, les branches pourront être fusionnées pour obtenir une version améliorée à l'intégralité du projet CISpaces.

Le code a été visualisé et écrit sous *Eclipse* étant déjà été utilisé au sein de l'IUT. L'espace de travail étant familier, il a été possible de développer plus rapidement.

Pour suivre les tests réalisés, des logs sont générés dans le code pour permettre de réaliser différents tests unitaires. Quand on récupère des extraits d'ontologies, la lecture de ces extraits peut s'avérer complexe et long. Pour faciliter la lecture d'une ontologie, Protégé permet un meilleure visualisation de l'ontologie, sur les classes, les propriétes et les différents individus.

6.3.3 Bilan critique

Ayant estimé la lecture de documentation à deux semaines, les aspects de la théorie de l'argumentation et de web sémantique ont été appréhendés. Néanmoins, c'est au travers des rendez-vous avec le tuteur et lors du développement des tests que les connaissances et leur appréhension se sont approfondies. Beaucoup de travail a été réalisé en auto-didactisme et avec les méthodes apprises à l'IUT. Avec l'assistance de mon tuteur de stage, j'ai pu gérer mes méthodes de travails comme faciliter la réalisation des tests au travers des logs. J'ai donc pu apprendre différentes façons de gérer les logs afin d'effectuer mes tests. En comparant le planning avec l'avancement du projet, de nouvelles erreurs et le temps effectué sur une partie prend plus de temps que prévu. La refonte nécessite plus de temps car même si une partie des tests unitaires fonctionnent, lors des tests fonctionnels d'autres erreurs peuvent surgir, il faut donc faire preuve de rigueur face à l'impact d'une refonte sur une partie de code.

Conclusion

Au cours de ces 9 semaines de stages passées, j'ai dû me familiariser avec un certain nombre de connaissances et de nouveaux outils. J'ai eu l'opportunité de réaliser un stage sur un sujet qui m'a éveillé une part de curiosité. De nouvelles notions ont pu êtres apprises, j'ai pu approfondir des connaissances déjà acquises dans le développement, en particulier la partie test. L'une des plus grosses difficultés a été de plonger dans un gros projet où de multiples fonctionnalités ont déjà été implémentées. Cela nécessite une lecture du code rigoureuse afin de le comprendre mais il faut aussi admettre que certaines parties de l'application sur lesquelles je ne travaille pas, marchent. De plus l'appréhension de nouvelles connaissances ou la lecture des logs ont pu être longs mais l'aboutissement à des résultats/changements procure une certaine satisfaction. Le changement de cadre, plus professionnel, et également le fait que le stage se déroule au Royaume-Uni, m'a permis de m'investir beaucoup plus dans un projet tout en améliorant mon anglais. Pour finir, ce projet m'a redonner envie de m'investir dans l'informatique, là où l'IUT ne me procurait pas autant de curiosité et spécialisation.

Bibliographie et Sitographie

Références

[1] Supporting Reasoning with Different Types of Evidence in Intelligence Analysis *Un premier rapport présentant CISpaces*, Alice Tonilo, Timothy J.Norman, Nir Oren, Anthony Etuk, Federico Cerutti, Timothy Dropps, John A .Allen, Robin Wentao Ouyang, Mani Srivastava, Paul Sullivan.

[2] Argumentation-based sens-making exploiting open-sources, *Second rapport une version plus élaborée de CISpaces* Documentation pas encore ouverte au public, Timothy J. Norman, Federico Cerutti, Stuart E. Middleton, and Alice Toniolo.

[3] Cardiff University, *Site internet de l'université de Cardiff*, https://www.cardiff.ac.uk/.

[4] Cardiff School of Computer science and Informatics, *Site internet du département informatique de l'Université de Cardiff*, http://www.cardiff.ac.uk/computer-science.

[5] Abstract Argumentation Semantics, *Présentation de différentes sémantiques et propriétés* Federico Cerutti, https://cardiff.cloud.panopto.eu/Panopto/Pages/Viewer.aspx ?id=f90d9d0d-c0a2-42af-af74-52b1683f367e.

[6] Apache Jena, *Introduction au Framework Jena*, https://jena.apache.org/.

[7] Collaborative Intelligence Spaces, *Site internet sur l'application web CISpaces* https://cispaces.org/.

[8] A Practical Guide To Building OWL Ontologies Using Prot´eg´e 4 and CO-ODE Tools Edition 1.3, *Tutoriel sur l'utilisation de Protégé*, University of Manchester, 24 Mars 2011, http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/.

[9] Apache Maven, *Introduction à Apache Maven*, https://maven.apache.org/index.html

[10] Efficient sat-based algorithms for abstract argumentation, *Explications derrière des sémantiques*, Federico Cerutti, 2018.

[11] Jena reasoning with rules, *Tutoriel sur la création de règles au sein d'une ontologie*, 2015, https://tutorial-academy.com/jena-reasoning-with-rules/.

Annexes Techniques

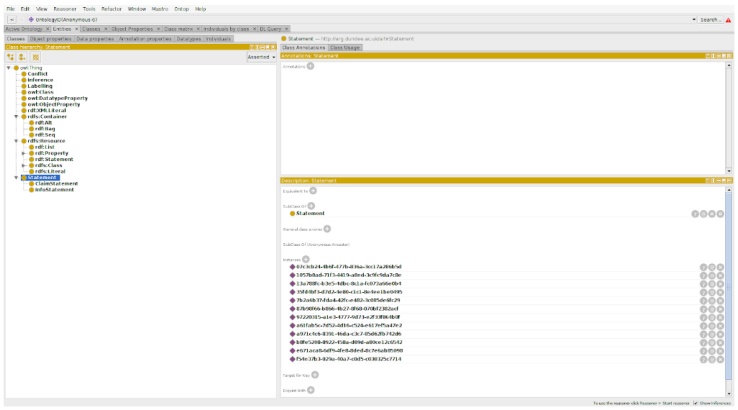

FIGURE 25 – Aperçu de l'interface protégé

```
399         private Set<Individual> rootBasedOn(Set<Individual> individuals) {
400                 Set<Individual> roots = new HashSet<Individual>();
401
402                 for (Iterator<Individual> it = individuals.iterator(); it.hasNext();) {
403                         Individual iInd = it.next();
404                         Set<Individual> intersection = new HashSet<Individual>(individuals);
405                         intersection.retainAll(this.getConclusions(iInd));
406                         if (intersection.isEmpty()) {
407                                 roots.add(iInd);
408                         }
409                 }
410
411                 if (debug) {
412                         NLG.log.log(Level.INFO, "***Roots list begins:");
413                         for (Iterator<Individual> r = roots.iterator(); r.hasNext();) {
414                                 NLG.log.log(Level.INFO,
415                                                 r.next().getPropertyValue(claimText).toString());
416                         }
417                         NLG.log.log(Level.INFO, "***Roots list ends.");
418                 }
419
420                 return roots;
421         }
422
```

FIGURE 26 – Fonction qui récupère les conclusions à partir de la *preferred extension*

```
546         private String individualsToString(Set<Individual> inIndividuals,
547                         Set<Individual> expanded, Set<Individual> outputted) {
548
549                 if (expanded == null) {
550                         expanded = new HashSet<Individual>();
551                 }
552                 StringBuilder out = new StringBuilder();
553
554                 Set<Individual> roots = this.rootBasedOn(inIndividuals);
555
556                 Stack<Individual> toExpand = new Stack<Individual>();
557
558                 for (Iterator<Individual> r = roots.iterator(); r.hasNext();) {
559                         toExpand.push(r.next());
560                 }
561
562                 this.recursiveNavigationIndividuals(out, toExpand, expanded, outputted);
563
564                 return out.toString();
565         }
```

FIGURE 27 – Fonction récursive qui parcourt les conclusions et prémisses

```
411                 if (debug) {
412                         NLG.log.log(Level.INFO, "***Roots list begins:");
413                         for (Iterator<Individual> r = roots.iterator(); r.hasNext();) {
414                                 NLG.log.log(Level.INFO,
415                                                 r.next().getPropertyValue(claimText).toString());
416                         }
417                         NLG.log.log(Level.INFO, "***Roots list ends.");
418                 }
419
420                 return roots;
421         }
```

FIGURE 28 – Partie de code qui génère des informations dans un log

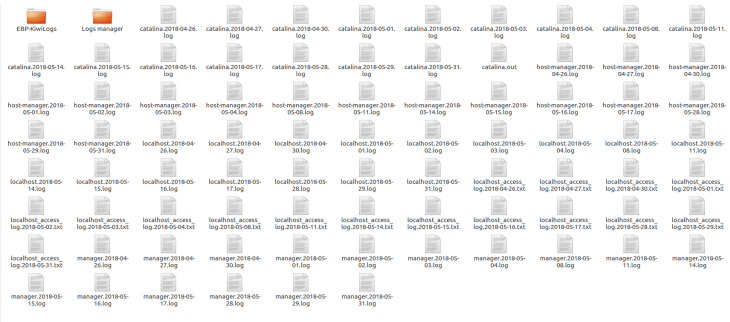

FIGURE 29 – Listes de logs

Résumé en Français Ce rapport de stage présente la participation d'un projet informatique au sein de l'Université de Cardiff. Au travers de celui-ci, nous verrons comment nous nous sommes organisés afin d'aboutir à une refonte de code java. CISpaces est une application web qui permet d'assister les analystes dans leur analyse, pour mieux traiter les informations, générer des hypothèses et extraire des informations de sources externes. Ce rapport permet de comprendre quelques notions derrière la Théorie de l'argumentation et le web sémantique. Nous retracerons également les différents tests employés ainsi qu'un aperçu sur l'espace de travail permettant de créer une analyse afin de générer un rapport en langage naturel.

Mots-clés : Refonte, application web, analyse,hypothèses, Théorie de l'argumentation, tests.

English summary This internship report shows my implication in a computer science project within Cardiff University. Through this report, we will see how we have organised ourselves to allow the refactoring of Java code. CISpaces is web-based service-oriented application that helps analysts in their different analysis, to make sense out of information, generate hypotheses and to extract informations from external sources. This report allows to understand a few notions behing Argumentation Theory and semantic web. We will also demonstrate the different tests that have been used and to also show how the workspace is able to create an analysis to then generate it into a natural language report.

Key words : Refactoring, web-based application, analysis, hypotheses, Argumentation theory, tests.