Architecture des Systèmes d'Information
Institut national des sciences appliquées de Rouen

Rapport de Stage de spécialité

NLP for Requirements Engineering

Développer des prototypes basés sur les technologies de traitement automatique du langage naturel (NLP) permettant d'analyser et de modifier des documents technico-contractuels rédigés en langue française.

Auteur : **Anass El Yaagoubi**

E-mail : **anass.el_yaagoubi@insa-rouen.fr**

Maître de stage : **Samuel Renault**

Organisme d'accueil : **Luxembourg Institute of Science and Technology**

Adresse : **5 Avenue des Hauts-Fourneaux, 4362 Esch-sur-Alzette, Luxembourg**

Dates : **01/06/2018 - 24/08/2018**

Année universitaire : 2017/2018

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au bon déroulement de mon stage au LIST et qui m'ont apporté leur aide durant les différentes étapes du stage.

Tout d'abord, je tiens à remercier sincèrement mon maître de stage, M. Samuel RENAULT, chercheur au sein du LIST, pour son implication et son investissement durant mon stage et son accueil qui m'a très rapidement permis de m'intégrer au sein du LIST. Grâce à ses conseils et à son investissement dans le stage j'ai pu affronter progressivement et sereinement les différents objectifs du stage.

Je remercie également tous mes collègues du LIST et plus particulièrement les membres de l'équipe SPG à savoir Béatrix BARAFORT, Wael CHARGUI, Stéphane CORTINA, Michel PICARD, Alain RENAULT, Philippe VALOGGIA. Grâce à eux je me suis très vite senti membre de l'équipe. Je remercie également Céline DECOSSE pour ses conseils en ingénierie des exigences.

Je remercie aussi les autres stagiaires et doctorants pour les bons moments que nous avons pu passer ensemble, c'était très riche en échanges et en partage tant sur le plan technique qu'humain.

Table des matières

Liste des Abréviations

API : Application Programming Interface.

HTML : HyperText Markup Language : méta langage informatique de balisage.

XML : Extensible Markup Language : méta langage informatique de balisage.

DOCX : Format de fichier respectant la norme Office Open XML

NLP : Natural Language Processing. Le NLP peut être vu comme une branche de l'informatique, de la statistique et de la linguistique.

TALN : Traitement Automatique du Language Naturel, traduction française du NLP.

Pandoc : Outil utilisé pour faire de la conversion de format de fichier.

spacy : Librairie NLP Open Source.

NLTK : Librairie NLP Open Source.

NER : Named-Entity Recognition.

Machine Learning : Une sous-branche de l'intelligence artificielle.

RNN : Recurrent Neural Networks

CRP : Centre de Recherche Publique.

RTO : Research and Technology Organisation.

RDI : Recherche, Développement et Innovation.

ITIS : IT for Innovative Services.

SPG : Service and Process Governance

BART : Business Analytics and Regulatory Technologies

Introduction

De nos jours le Natural Language Processing (NLP) est un domaine en plein essor. Les performances actuelles des algorithmes et technologies NLP permettent leur utilisation dans des domaines d'application variés.

Dans les procédures d'appel d'offres, les exigences sont l'expression d'un besoin qui doit être documenté. Ce besoin peut être de différentes natures par exemple un système d'information, une plate-forme en ligne, un service ou un produit particulier etc. Ces exigences sont regroupées dans un cahier des charges, elles doivent spécifier et expliquer le besoin du client dans sa globalité.

Les exigences sont donc une partie critique dans le cadre d'appel d'offres. Il est donc nécessaire que les exigences présentes dans un cahier des charges respectent un certain nombre de bonnes pratiques afin d'éviter toute forme d'ambiguïté et de transcrire exactement et de manière claire les besoins du client.

Ce stage avait pour objectif le développement de prototypes utilisant les technologies NLP permettant de faire un certain nombre de tâches telles que : anonymisation automatique de cahier des charges, extraction de concepts sémantiques présents dans un cahier des charges, détection des patterns de langage et comparaisons avec des patterns de références, construction de critères d'évaluation à partir des exigences.

Partie 1

Présentation du cadre de travail

Dans cette partie je vais présenter le cadre dans lequel s'est déroulé mon stage. Je vais donc commencer par faire une présentation assez rapide du LIST et puis je vais décrire le suivi du stage ainsi que mes premiers pas au LIST.

1.1 Présentation du LIST

1.1.1 Historique et Localisation

Luxembourg Institute of Science and Technology (LIST) est un Centre de Recherche Public (CRP) luxembourgeois, créé en Janvier 2015. Il est né de la fusion de deux centres de recherches public à savoir : le CRP Henri-Tudor qui a été créé en 1987 et qui était basé à Luxembourg-Ville et le CRP Gabriel-Lippmann qui a été créé en 1987, son siege social était basé à Belvaux.

Aujourd'hui le LIST est basé à Belval. Suite à l'abandon d'une partie de la production d'acier au Luxembourg, le LIST à été délibérément créé sur le site de l'ancienne friche industrielle sidérurgique de Belval, il est situé à quelques pas de l'Université du Luxembourg.

1.1.2 Chiffres clés

Le LIST est une Research and Technology Organisation (RTO) à but non lucratif qui regroupe plus de 592 collaborateurs dont 35% de femmes et 65% d'hommes. Les chercheurs présents au LIST sont d'origines très diverses, environ 40 nationalités y sont représentées avec

FIGURE 1.1 – Vue du LIST, entre les Hauts fourneaux

79 doctorants. Le LIST c'est aussi 319 projets et contrats RDI dont 32 autofinancés.

Le LIST se compose de trois départements qui sont orientés Matériaux, Environnement et Informatique. Le département informatique s'appelle ITIS et se divise en quatre unités de recherche, l'équipe au sein de laquelle j'ai travaillé s'appelait Service and Process Governance et appartenait à l'unité Business Analytics and Regulatory Technologies.

1.2 Organisation et premiers pas

Durant les premiers jours, mon maître de stage et moi nous nous sommes mis d'accord sur l'organisation que nous allions suivre tout au long du stage. Nous avons donc décidé de fixer des réunions matinales journalières, ainsi tout les jours nous faisions une réunion entre 9h et 10h qui durait en moyenne en entre 20 et 50 minutes. Durant ces réunions matinales, je montrais l'avancée de mon travail de la veille et mes plans pour le lendemain, c'était aussi l'occasion de discuter des difficultés rencontrées la veille, donc quand je bloquais longtemps sur un point en particulier c'était le moment où nous en discutions afin de trouver une solution.

Étant membre de l'équipe SPG, je participais également à des réunions d'équipe qui se déroulaient le Lundi matin, c'était l'occasion de faire du "knowledge sharing" et de découvrir sur quels sujets les autres travaillaient, ça permettait également de se mettre au courant des avancées de différents projets.

Le stage ayant une nature exploratoire, il n'y avait pas de cahier des charges à proprement

parler pour ce stage. Nous avons fait des réunions tout au long du stage pour spécifier et ordonner par priorité les objectifs du stage.

L'organisation suivie durant le projet ressemblait à une approche de type Agile du fait des user stories et des réunions matinales qui ressemblaient aux mélées. Mais nous n'avons pas suivi de méthodologie particulière à la lettre, il n'y avait pas vraiment besoin car je travaillais seul la plus part du temps et je voyais mon maître de stage très régulièrement.

Nous avons utilisé un outil interne appelé Forge pour suivre l'évolution du projet, c'est dans cet outils qu'étaient répertoriées les user stories. Nous avons également utilisé GitLab comme outil de gestion des version.

Durant les premières semaines j'ai réalisé un état de l'art sur le NLP afin de bien maîtriser le sujet, cet état de l'art répertoriait aussi les principales technologies open source en NLP.

Partie 2

Présentation du sujet, ingénierie des exigences et NLP

Dans cette partie je vais présenter mon sujet de stage et expliquer quels étaient les objectifs de ce stage et comment on pensait résoudre certains d'entre eux, puis je vais présenter quelques concepts importants concernant le NLP.

2.1 Présentation du sujet

Le sujet de mon stage était le suivant : Développement de prototypes en traitement du langage naturel pour l'amélioration de documents techniques (cahiers des charges).

Voici l'ensemble des objectifs prévus initialement :

1. Réalisation d'un état de l'art sur les technologies NLP open-source disponibles et de leurs capacités à répondre aux objectifs du stage.
2. Anonymiser un document (détection d'acronymes et d'entités nommées)
3. Extraire et structurer les principaux concepts sémantiques d'un document
4. Détecter différents types d'erreurs et d'incohérences dans des documents
5. Proposer des corrections pour les erreurs détectées
6. Détecter des patterns de langage et mesure de leur alignement par rapport à des patterns de référence
7. Extraire des critères d'évaluation

Initialement ce stage était prévu pour une durée de six mois, comme je ne pouvais pas effectuer un stage de six mois en ASI 4 nous avions dû revoir la durée du stage à la baisse, en conséquence les objectifs initiaux ont été revus à la baisse. Ainsi les objectif prioritaires étaient : anonymisation de documents technico-contractuels, extraction de concepts sémantiques et la reformulation d'exigences en question, mon stage a donc duré douze semaines.

2.2 Ingénierie des exigences

L'ingénierie des exigences devient une partie critique dès que l'envergure du projet devient conséquente. Que ce soit dans le domaine IT ou dans un autre domaine, les exigences doivent être bien rédigées et respecter par conséquent les bonnes pratiques déjà prédéfinies dans le domaine en question. Évidement on ne va pas rédiger les exigences de la même manière quelque soit le domaine, qu'il s'agisse de mettre en place une centrale de traitement des eaux usées ou de construire le nouveau système embarqué d'Ariane 6 on ne va pas se préoccuper des même choses, ce qui est critique et par conséquent doit être détaillé varie d'un projet à l'autre. Il existe plusieurs types d'exigences, nous considérerons les deux types d'exigences suivants :

1. Exigences fonctionnelles
   (a) à quoi sert le système
   (b) ce que doit faire le système
   (c) fonctionnalités utiles du système
2. Exigences non fonctionnelles
   (a) confidentialité
   (b) performance
   (c) portabilité
   (d) sûreté

Le monde de l'ingénierie des exigences est très vaste, durant mon stage on s'est concentré sur le domaine de l'appel d'offres. Les cahiers des charges peuvent être un élément sensible pour certaines entreprises, il arrive donc très souvent que ces cahiers des charges soient confidentiels. Ceci devient un problème lorsqu'on a besoin de communiquer un tel document à des fins d'exemple, même s'il est confidentiel. Quand cela se produit il y a besoin d'anonymiser le cahier des charges en question. De nos jours c'est une tâche qui se fait encore à la main et est donc très chronophage. Une des idées que nous avions pour réaliser cette partie se basait sur la notion d'extraction d'entités nommées.

Une fois la date limite d'une appel d'offre atteinte, on besoin de choisir qui remporte l'appel d'offre, il y a donc besoin de réaliser un classement. L'équipe à la quelle j'appartenais avait développé une méthode nommée CASSIS. Afin de choisir qui remporte l'appel d'offre, cette méthode produit un classement des candidats en réalisant une auto-évaluation. Afin de réaliser cette auto-évaluation, on doit produire un questionnaire qui reprend toutes les exigences du cahier des charges sous forme de questions. L'idée que nous avions était donc d'utiliser les technologie NLP pour la transformation de phrases en questions.

2.3 NLP

Il va sans dire que les progrès réalisés dans le Machine Learning ont beaucoup contribué à la médiatisation du NLP. De nos jours les algorithmes du NLP les plus avancés réalisent des tâches qui étaient inimaginables auparavant, telles que la traduction automatique, la génération de texte, l'analyse de sentiment, conversion de texte en parole ou parole en texte etc...

Mais ce n'est pas pour autant que le NLP est un domaine nouveau. Le NLP est né après la deuxième guerre mondiale. On remonte souvent les origines du NLP aux travaux d'Alain Turing et Noam Chomsky.

On peut voire le NLP comme étant un domaine à l'intersection entre la linguistique, l'informatique et la statistique. En effet pour faire du NLP nous avons besoin de comprendre le langage humain d'où le besoin de la linguistique, nous avons également besoin de modèles statistique et probabiliste pour faire des prédictions et de l'inférence d'où le besoin de la statistique et sans informatique tout cela ne serait qu'un tas d'idées, c'est donc avec l'informatique que l'on implémente tout cela. La plus part des tâches du NLP reposent sur les concepts suivant :

1. Morphologie : étude des mots et de leurs relations avec d'autres mots de la même langue. La morphologie analyse la structure des mots sous leurs différentes formes afin d'étudier les différents sens possibles d'un même mot
2. Racinisation : transformation des mots en leur racine
3. Lemmatisation : transformation des mots en leur lemme
4. Tokenisation : découpe d'un texte en mots que l'on appel tokens
5. Etiquetage morpho-syntaxique (POS Tagging) : association d'informations grammaticales aux différents tokens qui composent un texte

C'est donc la combinaison de certains de ces concepts qui va permettre de réaliser une tâche donnée. La plus part des librairies NLP mettent à disposition des outils permettant de faire les tâches cités ci-dessus.

Afin d'utiliser des outils tels que les réseaux de neurones ou autre, il est souvent nécessaire et même obligatoire de faire un prétraitement des données plus ou moins avancé selon la nature de ce qu'on cherche à faire.

Partie 3

Prétraitement des données

Dans cette partie je vais parler de la nature des données auxquelles j'avais accès, puis je vais expliquer comment j'ai réussi à extraire le texte qui représente les exigences.

3.1 Données brutes

J'avais à ma disposition au total une vingtaine de cahiers des charges, qui étaient rédigés entre 2002 et 2016. Les cahiers des charges n'avaient pas tous le même format, la plupart étaient en format DOCX, les autres étaient en format PDF.

Une des particularités de ces cahiers des charges est que certains étaient rédigés en Anglais voir même en Allemand ce qui réduit la quantité de données utilisable. En effet les modèles NLP dépendent très fortement de la langue d'origine du texte à traiter. Comme le Français était majoritaire nous avons décidé de ne traiter que les documents rédigés en Français. Certaines tâches comme l'anonymisation de documents peut être facilement adaptable pour la langue Anglais, pourvu que l'on dispose d'un modèle adéquat car le processus est le même.

3.2 Problème du format de données

Le format PDF étant un format orienté visuel et adapté pour l'impression, il n'est pas simple de traiter les documents PDF afin d'en extraire un texte qui soit bien structuré. Nous avons donc décidé de ne traiter que les documents en format DOCX.

Le format DOCX est un format adapté pour certaines applications de bureau et non pour l'extraction de données et NLP. Nous avons donc décidé de changer de format afin de faciliter l'analyse et l'extraction de la structure des documents.

Pour cela j'ai utilisé un outil appelé Pandoc, permettant de convertir le format très simplement, cet outil a l'avantage de supporter un grand nombre de formats. Nous avons fait le choix du HTML car c'est un format universel. De plus le HTML a l'avantage d'être omniprésent dans le WEB, et donc cela nous donnait accès à un grand nombre d'outils pour l'analyser et extraire le texte que nous voulions.

3.3 Extraction d'exigences

Une analyse des cahiers des charges une fois convertit en HTML permet de voir que toutes les exigences se trouvent sous des balises de type paragraphes ' Exigence '.

Nous avions donc besoin d'extraire les éléments de type paragraphes. Ce problème est très similaire à celui du "Web Scraping". J'ai donc décidé d'utiliser BeautifulSoup [1], une librairie permettant de faire du Web Scraping.

Chaque exigence est présente dans une balise de type paragraphe, mais l'inverse n'est pas vrai. Ceci pose donc un problème car les paragraphes ne sont pas annotés. J'ai donc développé une méthode permettant de déterminer si un paragraphe est une exigence ou non. Cette méthode détermine donc les phrases qui sont des exigences, elle se base sur la structure grammaticale de la phrase. Cette méthode a été utilisée par la suite dans l'extraction de concepts.

1. crummy.com/software/BeautifulSoup/bs4/doc/

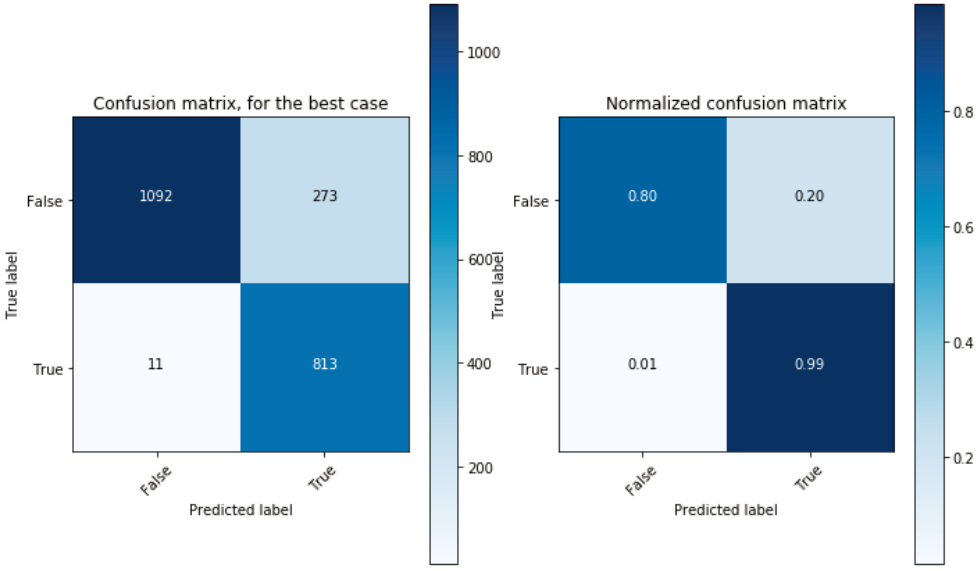

FIGURE 3.1 – Prédiction d'exigences à partir de la structure grammaticale

Partie 4

Anonymisation de cahiers des charges

Dans cette partie je vais détailler comment nous avons partiellement réussi à anonymiser un cahier des charges de manière automatisée, puis je vais analyser cette solution et dire ce qui pose problème ou ce qui limite cette approche et comment on pourrait éventuellement résoudre ce problème.

4.1 Solution imaginée

Après mûre réflexion j'ai décidé de procéder de la manière suivante : tout d'abord convertir le cahier des charges au format HTML, puis extraire l'ensemble des entités nommées présentes dans le cahier des charges en suite construire une table de correspondance pour les entités que l'on souhaite anonymiser, cette table associe à chaque entité nommée un identifiant texte générique qui permettra de garder un sens au texte sans avoir des informations critiques ou sensibles, suite à cela substituer dans le HTML chaque entité nommée par le terme qui lui correspond à partir de la table de correspondance et finalement reconvertir le fichier au format DOCX.

Durant la phase de substitution il faut faire attention à ne pas modifier les balises HTML, sinon la conversion de format risque de mal se passer.

Afin de réaliser l'extraction d'entités nommées j'ai décidé d'utiliser la librairie NLP nommée spacy [1]. Cette librairie est disponible en open-source, elle met à disposition des outils à la pointe de ce qu'on peut trouver en terme de performance que se soit en précision ou en vitesse.

1. spacy.io

Cette librairie présente également l'avantage d'être orientée business ce qui la rend très simple d'utilisation par ceux familier avec Python et le NLP. Je n'ai donc eu aucun problème de prise en main.

4.2 Extraction d'entités nommées

Les entités nommées peuvent être vues comme étant les éléments du langage qui font référence à une entité unique et concrète. Une entité nommée doit appartenir à un domaine ou catégorie spécifique qui peut dépendre du contexte, c'est ce qui rend cette tâche difficile. ces catégories peuvent être PERSONNE, LIEU ou ORGANISATION etc.

Le modèle NER en français proposé par spacy classifie les entités nommées en quatre catégories qui sont : PER (personne), LOC (lieu), ORG (organisation), MISC (autre).

L'extraction d'entités nommées avec spacy se fait de manière très simple et quasi-instantanée. Tout d'abord il faut charger le modèle NLP que l'on souhaite utiliser puis instancier un objet en lui donnant le texte en question comme paramètre.

Spacy fait par défault un certain nombre de tâches définies dans le pipeline, ce pipeline peut être modifié si besoin en rajoutant ou en éliminant certaines tâches.

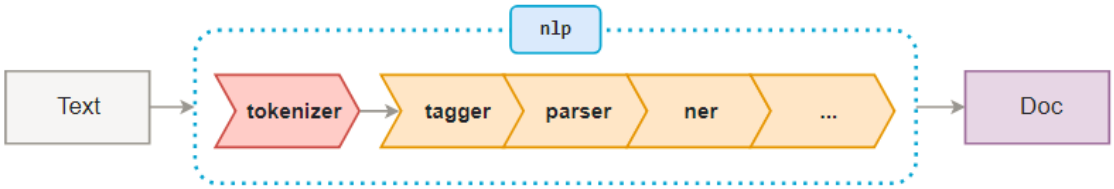

FIGURE 4.1 – Pipeline spacy

4.3 Anonymisation de cahier des charges

Une fois les entités nommées disponibles on peut aisément construire une table de correspondance. En procédant une entité nommée à la fois on peut ainsi substituer dans le HTML chaque entité par le texte qui lui correspond.

Une fois le HTML anonymisé, il suffit de convertir au format original à savoir du HTML vers DOCX en utilisant Pandoc.

L'analyse des cahiers des charges nous a permis de voir que certaines entités nommées étaient très bien reconnues, notamment lorsque celle-ci étaient des acronymes ou des mots capitalisés. Mais globalement on avait deux problèmes conséquents pour cette tâche d'anonymisation, tout d'abord certaines entités critiques telles-que les noms propres n'était pas toujours détectées ce qui pose une sérieuse limitation pour l'anonymisation, l'autre problème était celui des faux positifs c'est à dire que beaucoup d'entités nommées étaient détectées sans être vraiment des entités nommées, en voici deux exemples pour illustrer ce problème :

```
Entities :  [('Présentation de l'entreprise', 'MISC'), ('Business', 'LOC')]
Entities :  [('Présentation du projet 4', 'MISC')]
Entities :  [('A.3 Déroulement', 'PER')]
Entities :  []
Entities :  [('B.1 Gestion', 'MISC')]
```

FIGURE 4.2 – «Exemple de mauvaise extraction NER»

4.4 Amélioration du modèle NER de spacy

Une des raisons pour les-quelles le choix de spacy a été retenu pour ce projet, c'est que ça permet de prendre les modèles NLP de spacy est de les entraîner de manière simplifiée. Cela va idéalement permettre d'avoir un modèle spécifique et donc plus adapté à notre contexte c'est à dire au cahiers des charges.

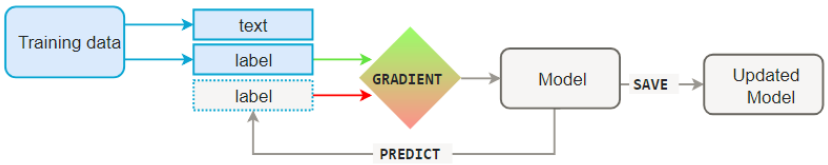

FIGURE 4.3 – Schéma type d'entraînement du modèle NER de spacy

Afin de pouvoir entraîner le modèle NER Francais, il faut que les données respectent la forme précisée par la documentation de spacy. [2]

Nous avons donc pensé aux différentes entités nommées qui pourraient porter un intérêt pour l'anonymisation, nous avons décidé d'en rajouter certaines comme CARDINAL, DATE,

2. https ://spacy.io/usage/training#section-ner

EMAIL, PERCENT et CURRENCY. L'annotation de documents de type cahier des charges est une tâche très chronophage. Nous avons donc annoté seulement deux cahiers des charges.

L'entraînement du modèle s'est fait grâce à la méthode update fournie par spacy, cette méthode exige un texte et des annotations entre autres. Les annotations sont une sorte de liste de tuple : catégorie, indice1, indice2 où les indices indiquent le début et fin de l'entité nommée dans le texte.

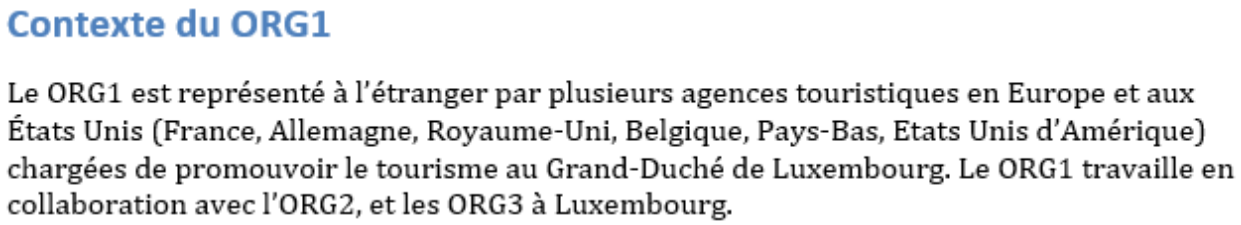

FIGURE 4.4 – Extrait d'un cahier des charges bien anonymisé

Dans cet exemple on comprend tout à fait le sens du paragraphe sans pour autant savoir de quelle organisation il s'agit. En analysant le contexte dans cet exemple on peut deviner que ORG1 représente le Ministère du Tourisme, mais dans d'autres cas c'est impossible de déterminer l'organisation en question car le contexte ne contient pas assez d'information. Cependant dans cet exemple on a aucune idée de ce que peuvent être les organisations 2 et 3.

Après entraînement nous pouvions extraire également les autres entités nommées de catégorie nouvelles :

```
Entities  :  [('Luxembourg', 'LOC')]
Entities  :  []
Entities  :  [('www.hrone.lu),', 'ORG'), ('www.luxe.lu', 'LOC'), ('2003 et 2004.', 'DATE')]
Entities  :  [('2007.', 'DATE'), ('Europe', 'LOC')]
```

FIGURE 4.5 – Exemple NER nouvelles catégories

Comme on peut le voir si dessus certaines entités nommées sont mal labellisées, et ce même après entraînement. C'est quelque chose de prévisible car nous n'avions pas annoté assez de documents, pour des raisons de ressource temporelle. Au total nous avions environ 300 entités nommées annotées, ce qui n'est clairement pas suffisant. En effet la documentation officielle de spacy suggère au minimum une centaine d'exemples pour faire l'apprentissage d'un seul label, or sur les 300 entités annotées, nous avons essayé d'apprendre plusieurs dizaines de labels.

Certain labels ont été plus tôt bien appris comme c'est le cas de PERCENT, c'est compré-

hensible par-ce que les entités nommées de type PERCENT ont un contexte textuel relativement peu variable ce qui explique pour quoi avec peu d'exemples c'est possible d'avoir des résultats corrects.

Nous avons pu résoudre le problème des faux positifs en fournissant des textes annotées sans entités nommées. Cependant pour les labels que nous avons entraînés, il persiste un problème de généralisation. Ceci est dû à la petite quantité de données pour réaliser l'entraînement du modèle. Il faudrait alors collecter d'autres cahiers des charges et extraire leur entités nommées et les annoter et passer suffisamment de temps pour bien entraîner le modèle NER.

Partie 5

Extraction de concepts sémantiques

Dans cette partie je vais décrire la solution que j'ai implémentée pour réaliser l'extraction de concepts. Pour comprendre la mise en œuvre de la solution je vais devoir présenter les notions de similarité qui ont été considérées mais également la notion de grammaire non contextuelle.

5.1 Solution proposée

5.1.1 Exigences et concepts sémantiques

Tout d'abord commençons par définir ce que l'on appellera concept sémantique par la suite. Ce que nous appelons concepts sémantiques ce sont les termes clés qui donnent le sens à l'exigence, sans les-quels la phrase n'aurait plus de sens, ce sont les éléments sur lesquels portent l'exigence. Par exemple considérons la phrase suivante qui représente une exigence fonctionnelle : "Le système doit permettre à l'utilisateur de définir le caractère exclusif ou pas de certains critères de classification", en tant qu'individus nous pouvons aisément extraire les mots suivant "système", "permettre", "utilisateur", "définir", "caractère exclusif" et les considérer comme étant des concepts sémantique clés de notre exigence, ce sont donc ces quatre concepts qui résument l'exigence.

La difficulté de cette tâche provient de la variabilité des phrases d'exigences. En effet, les exigences étant rédigées en langage naturel souvent non contrôlé, d'une exigence à une autre la structure grammaticale de la phrase peut varier énormément ce qui complique l'automatisation de la tâche.

5.1.2 Méthode utilisée

Une solution envisageable pour résoudre cette tâche serait d'utiliser des modèles de Deep Learning via des méthodes de word embedding. Les mots n'étant pas des vecteurs numériques on ne peut pas les utiliser directement pour le Deep Learning, il faut donc réaliser ce que l'on appelle word embedding afin de transformer le texte en séquence de vecteurs. Notre problème deviendra ainsi un problème de séquence à séquence, une sorte donc de résumé de texte pour lequel il existe beaucoup de modèles de type RNN (les réseaux neuronaux récurrents ont la particularité de posséder une sorte de mémoire permettant de détecter des patterns dans une séquence ..). Nous avons rapidement abandonné cette approche car elle nécessite énormément de données pour apprendre les paramètres du réseau, en effet nous n'avions clairement pas assez de données, cette approche est à garder à l'esprit car elle peut très bien fonctionner si elle se trouve bien implémentée.

Nous avons opté pour le choix d'une méthode radicalement différente, qui ne se base pas sur les réseaux de neurones. Nous avons décidé de réaliser un clustering (groupement d'exigences par similarité) des phrases d'exigences dans une première étape, une fois ce clustering des exigences réalisé, nous procéderons par groupe d'exigences à la fois en utilisant le concept des grammaires non contextuelles afin d'extraire les concepts clés. L'intérêt de cette approche est qu'elle permet d'être plus sensible aux différentes structures des exigences, l'avantage le plus considérable c'est surtout le fait que nous serons capable d'interpréter nos résultats et nous pourrons aussi utiliser des connaissances linguistiques établies au préalable.

5.2 Notions de similarité

Dans la partie précédente j'ai parlé de clustering, pour réaliser ce clustering il faut nécessairement une notion de distance ou de similarité entre les exigences. Comme nous souhaitions grouper les exigences par affinité de structure, il fallait que cette notion de similarité prenne en compte la structure de chaque exigence. Nous avons retenu deux approches de mesure de similarité entre les exigences, une se base sur la notion des NGrams, et l'autre se base sur la similarité entre Graphes.

5.2.1 Similarité basée sur la notion de NGrams

Lorsqu'on dispose d'une séquence (de longueur L) d'éléments, on appelle NGram de cette séquence la séquence (de longueur L + 1 - N, avec N $\leq$ L) des sous-séquences consécutives de longueur N. Dans notre cas nous avons pris N = 2. Ainsi nous avons la définition des NGrams avec N = 2 ou Bi-Grams :

$$NGram((A, B, C, D)) = \{(A, B), (B, C), (C, D)\}$$

La notion de similarité entre deux phrases devient donc la proportion d'éléments communs aux NGrams des deux phrases.

Cette notion de similarité [B1] contient plusieurs variantes qui consistent à comparer les éléments communs de NGrams par rapport à la taille du plus grand NGram (2), par rapport au plus petit (3), ou par rapport à la réunion des deux NGrams (1).

$$Sim_1(exigence1, exigence2) = \frac{2 * |NGram(exigence1) \cap NGram(exigence2)|}{|NGram(exigence1) \cup NGram(exigence2)|}$$
$$Sim_2(exigence1, exigence2) = \frac{|NGram(exigence1) \cap NGram(exigence2)|}{max\{|NGram(exigence1)|, |NGram(exigence2)|\}}$$
$$Sim_3(exigence1, exigence2) = \frac{|NGram(exigence1) \cap NGram(exigence2)|}{min\{|NGram(exigence1)|, |NGram(exigence2)|\}}$$

Une fois cette notion de similarité définie on définit la distance entre deux phrases s_1 et s_2 comme étant :

$$d(s_1, s_2) = 1 - Sim(s_1, s_2)$$

Nous venons de définir les NGrams par rapport à la notion de séquences, mais à partir d'une phrase il n'est pas clair de définir la notion de séquence. En effet on peut considérer la séquence de lettres ou séquence de mots etc. Comme nous voulons refléter la similarité entre structures nous avons construit des séquences à partir des tokens de chaque phrase, c'est à dire que le NGram d'une phrase d'exigence est la séquence de POS Tagging de chaque token de la phrase. Une illustration est disponible en annexe (Figure 7.4).

5.2.2 Similarité basée sur la notion de distance entre graphes

L'autre approche que nous avons considérée se base sur les graphes et la notion de distance spectrale [B2]. Pour cela à partir de chaque exigence on construit une représentation sous forme de graphe. Chaque token d'une phrase donne lieu à un nœud du graphe et chaque dépendance entre tokens telle que sujet et verbe donne naissance à un arc entre les nœuds du graphes correspondants.

La distance spectrale c'est une sorte de distance L^2 entre les spectres des laplaciens représentant chaque graphe. Le laplacien L d'un graphe G est défini comme étant $L = D - A$ ou D est la matrice des degrés de G et A la matrice d'adjacence de G. Le spectre c'est les valeurs propres de la matrice L. Donc la distance spectrale entre deux phrases c'est la somme des carrés des différences entre les valeurs propres des laplaciens des graphes respectifs.

$$d_{spectrale}(G_1, G_2) = \sum_{i=1}^{k} (\lambda_{1,i} - \lambda_{2,i})^2$$

J'ai utilisé la librairie Networkx [1] afin de construire les graphes et de calculer les laplaciens et valeurs propres.

Une fois ces deux notions de distance en main nous pouvions calculer des distances entre les exigences, ainsi on avait pu construire deux matrices de distance precomputed (matrice de distance pré-calculée).

5.3 Clustering des exigences

Une fois les matrices de distance calculées, le clustering devient réalisable en utilisant l'option "precomputed" pour la matrice de distance, j'ai ainsi réalisé des clustering hiérarchiques car on ne connaissaient pas à l'avance le nombre de clusters que nous souhaitions. A l'aide des dendrogrammes on peut visualiser le résultat du clustering, voir annexe (Figure 7.9).

1. networkx.github.io

5.4 Extraction de concepts

5.4.1 Grammaire non contextuelle

En informatique théorique et en linguistique, une grammaire non contextuelle est une grammaire formelle qui se compose de règles appelées productions. Ces productions sont comme des règles d'écriture, elle définissent ainsi l'ensemble des combinaisons pouvant être générées de la sorte.

Nous avons utilisé cette notion de grammaire avec NLTK [2] car spacy ne dispose pas d'un tel outil. Nous avons utilisé cette notion de grammaire non contextuelle afin de détecter des construction grammaticales dans les phrases d'exigences après avoir réalisé une analyse de la phrase.

```
grammar = r"""
  DEBUT: {<DET><NOUN><AUX><VERB>}
  OBJEC: {<DET><NOUN><PUNCT><VERB>}
  NNOUN:{<DET><NOUN>}
  EXIGE1: {<DEBUT><OBJEC>}
  EXIGE2: {<DEBUT><NNOUN>}
  """
```

FIGURE 5.1 – Exemple de grammaire non contextuelle définie avec la syntaxe de NLTK

5.4.2 Résultats de l'extraction

L'idée derrière le clustering était de détecter des patterns récurrents parmi les exigences. Après analyse des résultats du clustering (tâche très chronophage par ailleurs), nous avons remarqué que certes les exigences dans le même groupe sont similaires mais pas au point de détecter un pattern commun récurent que l'on peut décrire avec une grammaire simple, à moins de couper le dendrogramme suffisamment bas ce qui n'est pas réaliste car on obtient beaucoup trop de groupes. En fait le seul pattern récurent d'intérêt détecté était celui de début d'exigence. On peut voir la grammaire construite pour ce pattern dans la section précédente.

Les résultats ainsi trouvés sont encourageants. Mais la solution proposée a besoin tout de même d'être améliorée. Quelques pistes envisageables pourraient être de revoir les notions

2. nltk.org

de similarité en essayant des tri-gram ou en améliorant la notion de distance entre graphes d'exigences. Une possibilité pourrait être d'essayer les réseaux de neurones à condition d'avoir suffisamment de données.

L'idée d'utiliser des grammaires non contextuelles fonctionne très bien à condition de bien définir une grammaire qui reflète la structure des exigences. Cette approche peut être poussée plus loin en définissant des grammaires sur lesquelles se baseraient les bonnes pratiques de rédaction des exigences, on n'aurait alors plus besoin de clusterer les exigences pour bien définir la grammaire car on ferait l'inverse.

	_sentence	concept1	concept2	concept3	concept4	concept5
0	Le prestataire devra proposer une offre de prix forfaitaire pour le projet d'adaptation du système pour répondre aux exigences formulées dans le cahier des charges.	prestataire	devra	proposer	offre	None
1	Cette offre fera clairement apparaître :	None	None	None	None	None
2	Le prestataire devra formuler une offre de prix pour l'utilisation du système en mode hébergé par les utilisateurs du Ministère du Tourisme, de l'Office National du Tourisme et des Offices	prestataire	devra	formuler	None	None
3	Cette offre de prix comprendra les coûts de mise à disposition et de maintenance du système en mode hébergé par le prestataire.	None	None	None	None	None
4	Cette offre de prix sera détaillée par mois et par utilisateur du système.	None	None	None	None	None
5	Le prestataire devra préciser dans son offre le tarif journalier pour l'analyse et le développement des demandes supplémentaires (ou change requests) qui pourraient survenir au cours du projet ou lors de l'utilisation du système.	prestataire	devra	préciser	None	None
6	Le système doit permettre à l'administrateur de définir quels utilisateurs (nommément) ou quels profils d'utilisateurs peuvent effectuer certaines activités particulières, comme l'effacement de contacts ou l'élaboration de critères de classifications.	système	doit	permettre	administrateur	définir
7	Pour la majorité des éléments d'informations manipulés dans le système, le Ministère du Tourisme souhaite pouvoir :	Tourisme	souhaite	pouvoir	None	None
8	Le système doit permettre à l'administrateur de définir quels utilisateurs (nommément) ou quels profils d'utilisateurs peuvent gérer les critères de classification.	système	doit	permettre	administrateur	définir
9	Le système doit permettre à l'utilisateur d'établir une classification, par exemple sous forme de critères et de sous-critères.	système	doit	permettre	None	None
10	Le système doit permettre à l'utilisateur de définir le caractère exclusif ou pas de certains critères de classification, c'est-à-dire le fait qu'un critère ne peut être choisi simultanément avec d'autres critères.	système	doit	permettre	utilisateur	définir
11	Le système doit permettre à l'utilisateur de préciser à quels éléments s'appliquent les critères de classification, par exemple les contacts et les articles de journaux pourraient être classés par thème (sport/vélo) mais les articles de journaux pourraient être également classés suivant le fait qu'ils sont mensuels ou annuels.	système	doit	permettre	utilisateur	préciser
12	Le système doit permettre à l'utilisateur de créer des unités organisationnelles pour représenter les agences, le Ministère du Tourisme, l'Office National du Tourisme, les Offices Régionaux du Tourisme et leurs relations de dépendances.	système	doit	permettre	utilisateur	créer

FIGURE 5.2 – Exemple de concepts extraient à partir d'un cahier des charges

Partie 6

Extraction de critères d'évaluation

Dans le cadre d'appel d'offre, la méthode CASSIS réalise un classement des offres via le principe d'autoévaluation. Afin de s'auto évaluer les entreprises doivent répondre à un questionnaire, ce questionnaire est généré manuellement en reprenant les exigences sous forme de question. D'où le besoin de reformulation d'exigences sous forme de questions.

6.1 Manque d'outils open-source pour la conjugaison automatique

Après une recherche d'outils existant en open-source, j'ai n'ai pu trouver aucun outil pour faire de la conjugaison automatique de verbes en français. J'ai également cherché une base de données qui contiendrait des données sur la conjugaison en français. La seule base trouvée contenait des données relatives à une cinquantaines de verbes ce qui n'est vraiment pas suffisant.

6.2 Solution implémentée

La solution trouvé était de reformuler la phrase d'exigence au tour du verbe pivot. Ce verbe est localisé à l'aide de la grammaire non contextuelle, une fois ce verbe localisé il suffit de le conjuguer à la troisième personne, et de regrouper la phrase dans le bon ordre.

Pour réaliser la conjugaison automatique j'ai développé un scrapeur de site web de conju-

gaison, tels que celui du figaro [1] ou celui du lemonde [2].

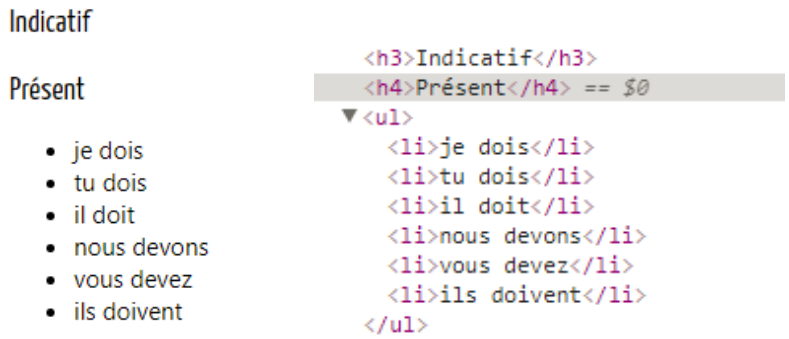

FIGURE 6.1 – Exemple verbe devoir, vue du site web à gauche et HTML à droite

Le site web final a été choisi en fonction de la diversité des verbes proposés en conjugaison en plus de la structure HTML du site web en question. Il est claire qu'une simple structure HTML simplifierait grandement la tâche.

6.3 Résultats

```
Exigence : Le système doit permettre de définir le caractère confidentiel de ces pièces jointes.
Critère  : Le système permet-il de définir le caractère confidentiel de ces pièces jointes ?

Exigence : Le système doit proposer une liste de contacts correspondant aux critères de classif-
           ication choisis par l'utilisateur.
Critère  : Le système propose-t-il une liste de contacts correspondant aux critères de classifi-
           cation choisis par l'utilisateur ?
```

FIGURE 6.2 – Exemple d'exigences reformulées en questions

Les résultats obtenus sont très satisfaisants, en effet une fois que la structure définie par la grammaire non contextuelle est détectée, la reformulation est quasi parfaite. On arrive même à reformuler des exigences dont le verbe pivot se trouve en milieu de phrase. Cependant il reste quelques problèmes, certaines exigences dont la structure est un peu à part ou avec une certaine particularité comme une redondance de verbes par exemple posent encore problème.

1. conjugaison.lemonde.fr
2. leconjugueur.lefigaro.fr

Partie 7

Synthèse

Dans cette partie je vais lister l'ensemble des technologies et les techniques utilisées pour réaliser les différentes parties de ce projet.

7.1 Technologies

Technologies \ Tâches	Anonymisation de documents	Extraction de concepts	Reformulation d'exigences
spaCy	✓	✓	✓
NLTK	✗	✓	✓
NER	✓	✗	✗
BeautifulSoup	✓	✓	✓
Numpy	✗	✓	✗
Scipy	✗	✓	✗
Networkx	✓	✓	✗
Matplotlib	✗	✓	✗
Pandas	✓	✓	✓
Urllib	✗	✗	✓

7.2 Techniques NLP utilisées

Techniques \ Tâches	Anonymisation de documents	Extraction de concepts	Reformulation d'exigences
Tokenisation	✓	✓	✓
Lemmatization	✗	✗	✓
POS Tagging	✓	✓	✓
NER	✓	✗	✗
Grammars	✗	✓	✓

Conclusion

Durant ce stage j'ai pu travailler essentiellement sur trois grandes tâches qui sont l'anonymisation de documents technico-contractuels, l'extraction de concepts sémantiques et la reformulation d'exigences en critères d'évaluation. Nous n'avons pas traité la tâche qui consiste à mesurer l'alignement des patterns avec des patterns de référence, certaines tâches peuvent être encore améliorées notamment la partie concernant l'anonymisation de documents.

Pour chacune de ces trois tâches un prototype à été réalisé sous la forme d'un ou plusieurs Jupyter Notebooks en utilisant des technologies open source en plus du package Library réalisé durant le stage. Ces prototypes ont pour objectif de montrer par proof of concept que certaines tâches qui sont aujourd'hui faites manuellement peuvent être automatisées sous certaines conditions, ces prototypes montrent également certaines limites que peuvent avoir les outils open source.

Ce stage m'a permis de découvrir et approfondir de monde de l'ingénierie des exigences et celui du NLP. Avant le début de mon stage j'avais des connaissances superficielles sur le NLP, ce stage a donc été l'occasion pour moi d'approfondir mes connaissances en NLP tant du point de vue théorique que du pont de vue pratique. Durant ce stage j'ai pu utiliser d'autres outils, avec les quelles je n'étais pas familier, notamment pour tout ce qui concerne le Web Scraping et l'extraction d'information mais également la représentation de graphes.

Bibliographie

[B1] Grzegorz Kondrak. 2005. N-gram similarity and distance.

[B2] Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos. 2016. DeltaCon : Principled Massive-Graph Similarity Function with Attribution.

[B3] Mavin Alistair, Wilkinson Philip, Harwood Adrian, Novak Mark. 2009. Easy approach to requirements syntax (EARS).

[B4] Arellano Andres, Zontek-Carney Edward, Austin Mark. 2015. Frameworks for Natural Language Processing of Textual Requirements. International Journal on Advances in Systems and Measurements.

Annexes

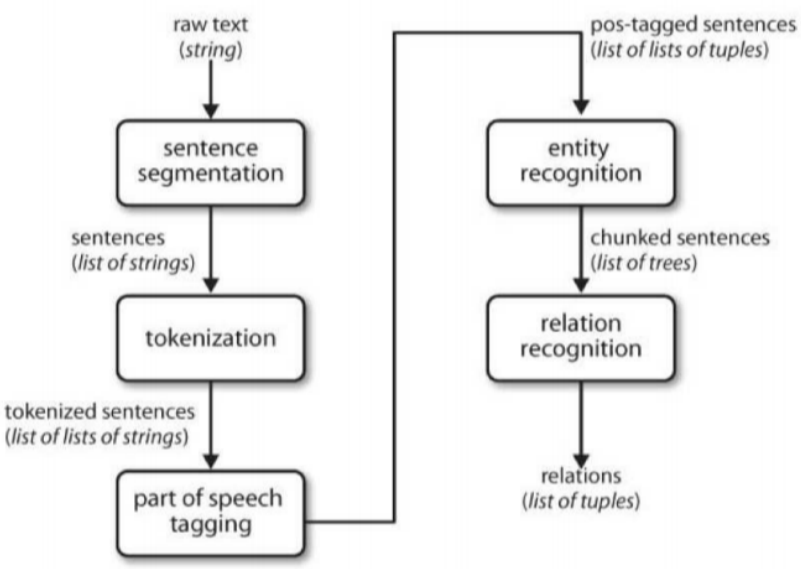

FIGURE 7.1 – Exemple d'un flux typique de traitement en NLP

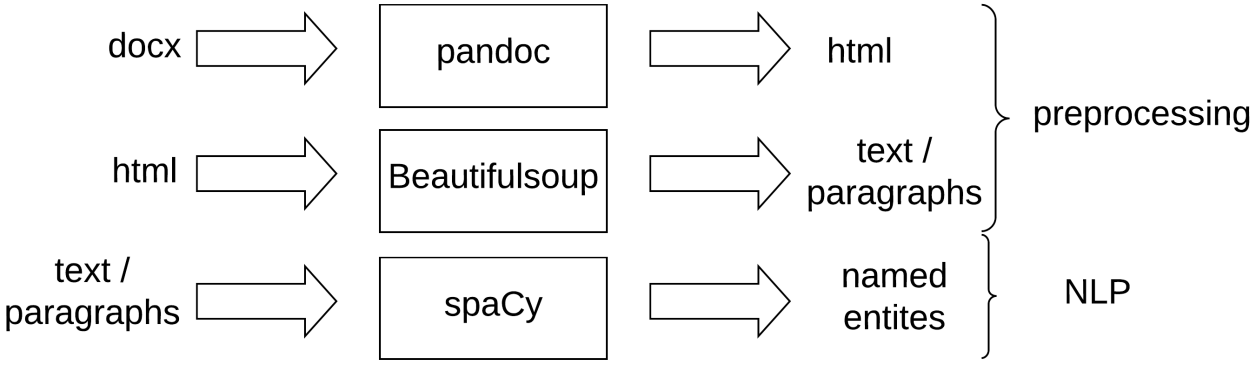

FIGURE 7.2 – Prétraitement des données réalisé en début de stage

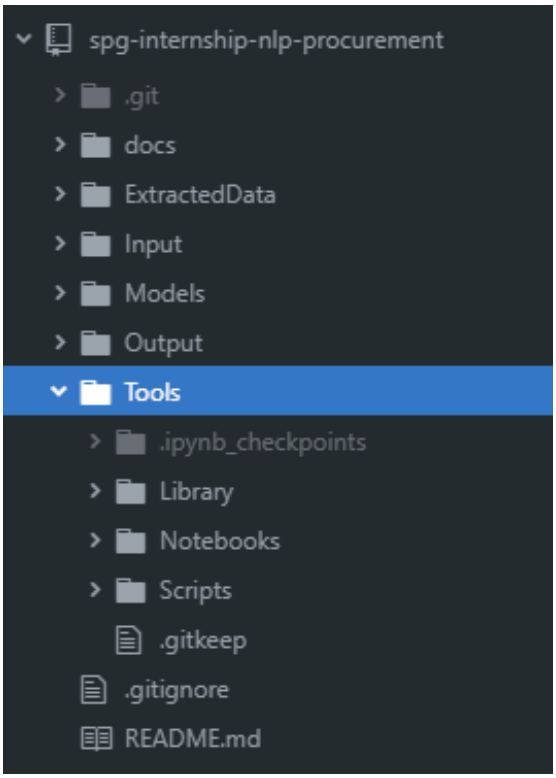

FIGURE 7.3 – Hiérarchie des dossiers du projet réalisé

```
( ('Le', 'DET')                ('système', 'NOUN') )
( ('système', 'NOUN')          ('doit', 'AUX') )
( ('doit', 'AUX')              ('permettre', 'VERB') )
( ('permettre', 'VERB')        ('à', 'ADP') )
( ('à', 'ADP')                 ('l’', 'DET') )
( ('l’', 'DET')                ('utilisateur', 'NOUN') )
( ('utilisateur', 'NOUN')      ('d’', 'PUNCT') )
( ('d’', 'PUNCT')              ('établir', 'VERB') )
( ('établir', 'VERB')          ('une', 'DET') )
( ('une', 'DET')               ('classification', 'NOUN') )
( ('classification', 'NOUN')   (',', 'PUNCT') )
( (',', 'PUNCT')               ('par', 'ADP') )
( ('par', 'ADP')               ('exemple', 'NOUN') )
( ('exemple', 'NOUN')          ('sous', 'ADP') )
( ('sous', 'ADP')              ('forme', 'NOUN') )
( ('forme', 'NOUN')            ('de', 'ADP') )
( ('de', 'ADP')                ('critères', 'NOUN') )
( ('critères', 'NOUN')         ('et', 'CCONJ') )
( ('et', 'CCONJ')              ('de', 'ADP') )
( ('de', 'ADP')                ('sous-critères', 'NOUN') )
( ('sous-critères', 'NOUN')    ('.', 'PUNCT') )
```

FIGURE 7.4 – Exemple de Bi-Gram construit à partir d’une phrase d’exigence

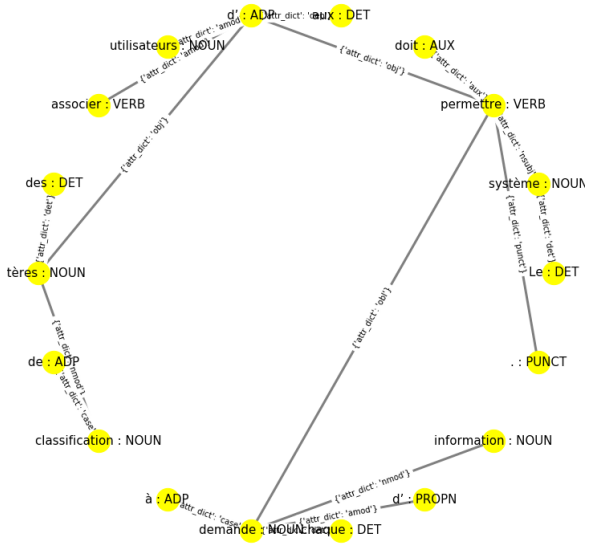

FIGURE 7.5 – Exemple de graphe construit à partir de la phrase d'exigence suivante : "Le système doit permettre aux utilisateurs d'associer des critères de classification à chaque demande d'information."

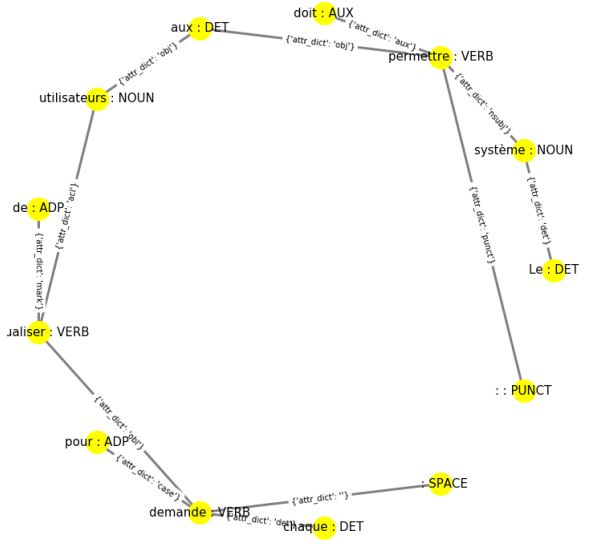

FIGURE 7.6 – Exemple de graphe construit à partir de la phrase d'exigence suivante : "Le système doit permettre aux utilisateurs de visualiser pour chaque demande :"

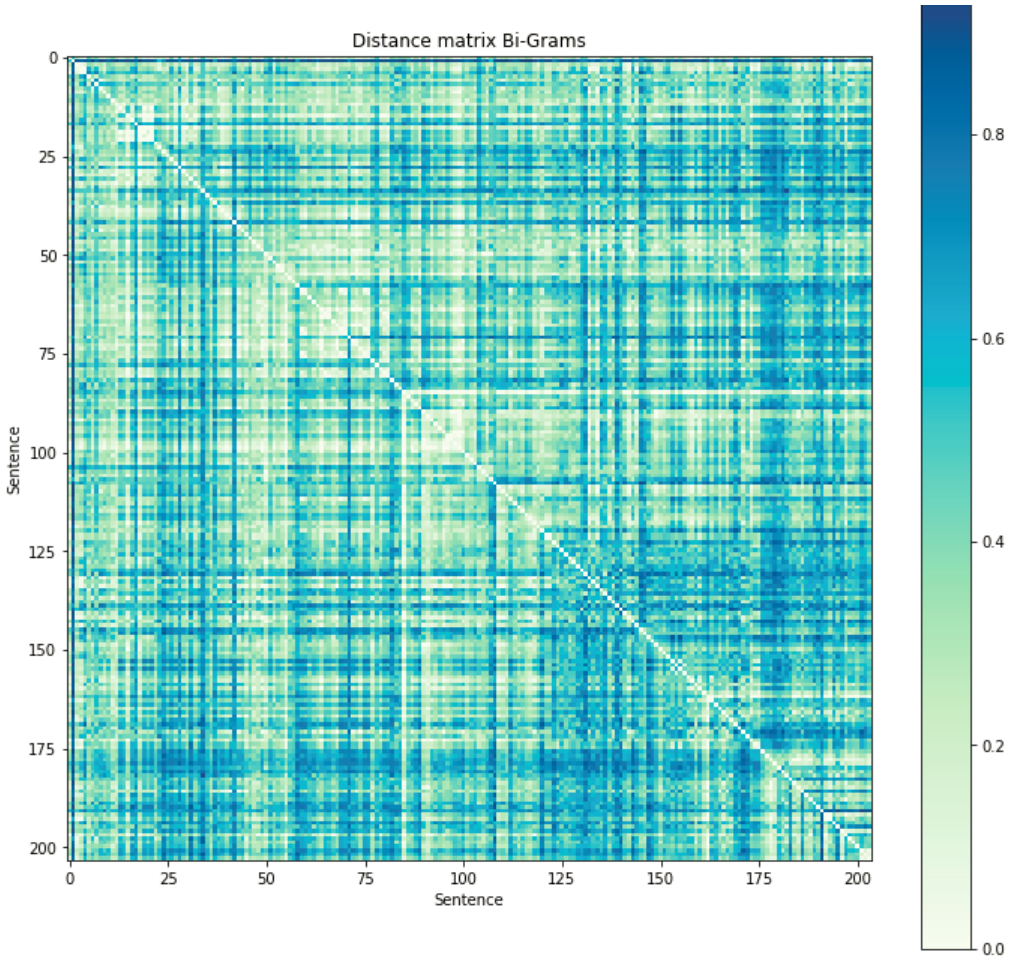

FIGURE 7.7 – Matrice de distance Bi-Gram, 204 exigences

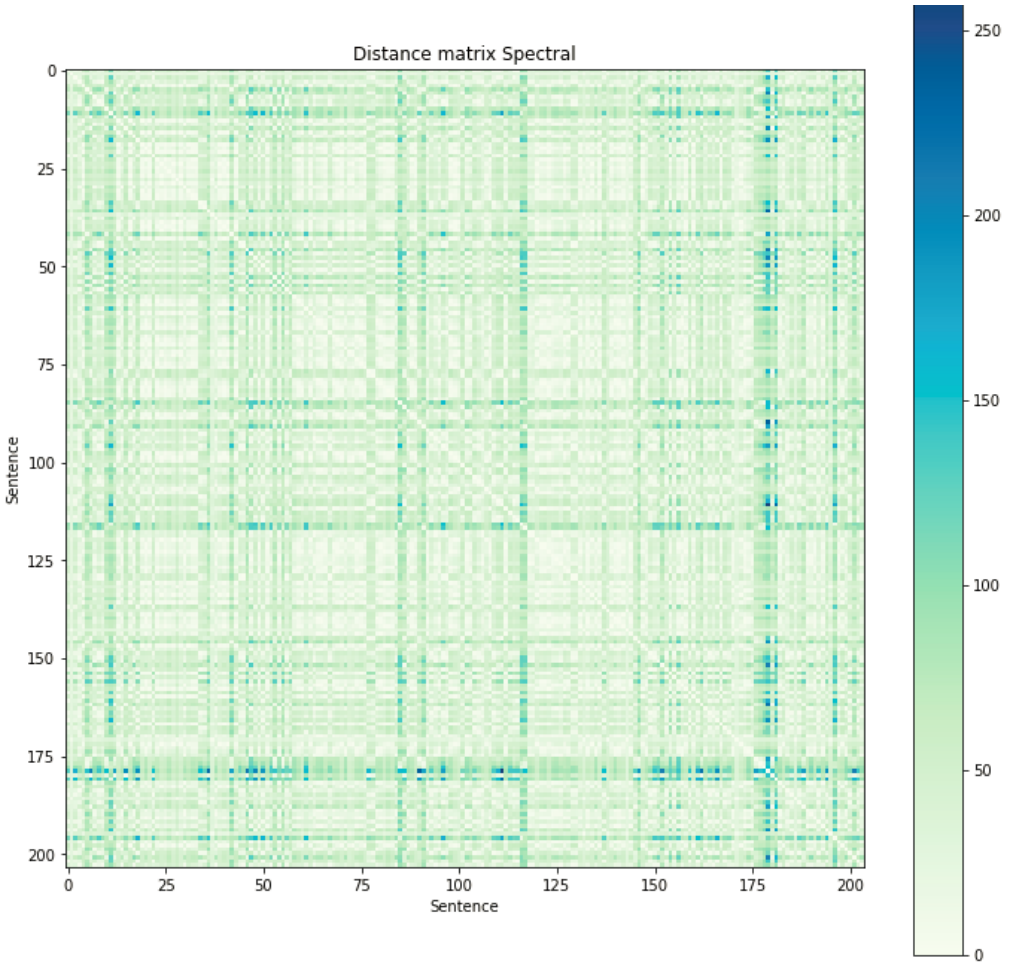

FIGURE 7.8 – Matrice de distance Spectrale, 204 exigences

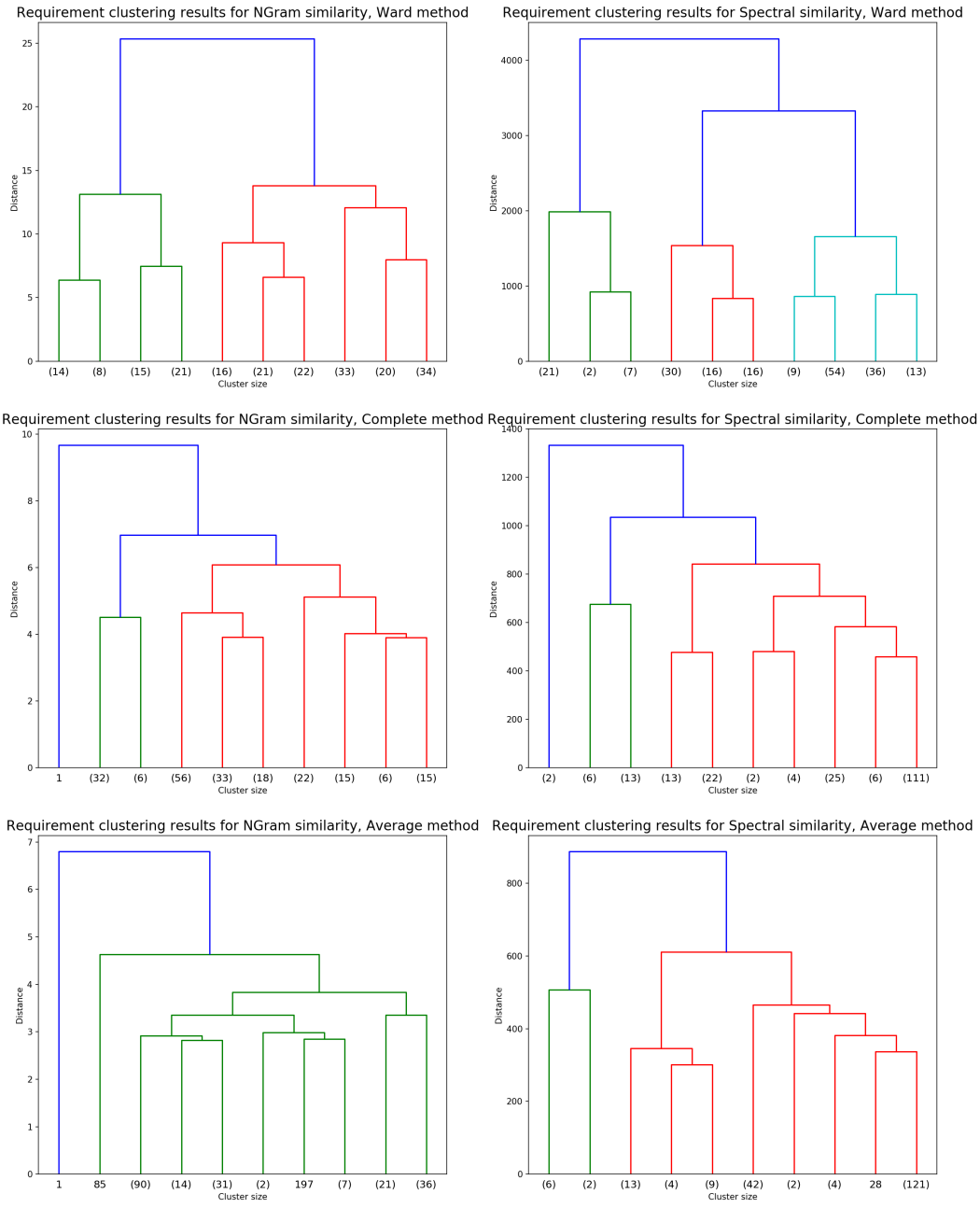

FIGURE 7.9 – Dendrogammes du clustering pour les deux notions de similarité, avec différentes méthodes de fusion. A gauche avec l'approche NGrams et à droite l'approche Graphes, de haut en bas les méthodes de fusion sont respectivement Ward, Complete et Average