Master 1 Informatique
Université des Sciences et Technologies
Lille I

Projet individuel

"PJI n°54 : Mais que font les députés ? Une sociologie informatique du travail parlementaire"

Encadrant universitaire : Samuel Hym
Encadrant : Etienne Ollion
Étudiant : Ouamar Sais

Table des matières

1 Remerciements

Merci à Samuel Hym et à Etienne Ollion pour leur accessibilité, pour leur suivi, leur réponses à mes questions, et pour avoir fournit ce sujet très intéréssant.

Introduction : Description générale du projet

Projet personnel

Le PJI est un projet personnel qui se déroule au second semestre de la première année de Master informatique à Lille 1. L'étudiant choisit un sujet puis le développe sur l'ensemble du semestre.

Ce rapport récapitulera le travail effectué sur ce projet ainsi que ses résultats. Dans un premier temps, il s'agira d'une description concrète du projet choisit, puis une présentation des outils qui ont été utilisés pour le développement de celui-ci ainsi que la raison pour laquelle il ont été choisis. Ensuite, nous aborderons a manière dont les données nécessaires au projet ont été récupérées afin de pouvoir continuer sur leur nettoyage et analyse. Enfin dans une dernière partie, il s'agira de la façon dont les données récupérées par l'analyse ont été stockées.

Description du sujet

Le projet est le numéro 54 dans la liste des PJI, celui-ci se nomme "Mais que font les députés ? Une sociologie informatique du travail parlementaire" et vise à exploiter les comptes rendus intégraux de l'assemblée nationale afin mieux connaître son fonctionnement. Il est encadré par Samuel Hym, enseignant chercheur au laboratoire CRYStAL de Lille 1 ainsi que par Etienne Ollion, chercheur CNRS au laboratoire SAGE de Strasbourg.

Plus particulièrement, durant les séances publiques, les discussions donnent parfois lieu à des échanges plus ou moins vif entre parlementaires, les comptes rendus font état de certaines de ces manifestations. Le but ici est de réussir à récupérer les indications de ces "humeurs" (applaudissement, protestation ...).

Le problème qui se pose dans la démarche d'analyse de ces comptes rendus est que ceux-ci, de par leur lisibilité moyenne, ne facilitent pas la récupération de ces informations de manière humaine. C'est pourquoi le projet demande la création d'un outil qui permettra d'analyser automatiquement chaque compte rendu de séance et d'en extraire les informations voulues.

2 Description générale de l'architecture du programme

3 Outils utilisés

Pour la réalisation du projet, aucun impératif concernant les technologies utilisés n'ont été donnés,

3.1 Java/Scala

Java et Scala sont les deux languages de programmation qui ont été utilisés pour mener à bien le projet, ils ont chacun servis dans deux parties distinctes.

Scala a été utilisé pour effectuer la récupération des fichiers nécessaire pour la suite. C'est un langage qui associe les deux paradigmes de programmation objet et fonctionnel quand que ceux-ci sont habituellement opposés. Il a aussi l'avantage d'avoir un typage statique et surtout d'être compilable sur la JVM ce qui permet de pouvoir utiliser les bibliothèques Java ou même de mixer du code Java avec du code Scala. Il est utilisé avec l'outil open source SBT qui facilite la construction d'application scala.

Le language Java est utilisé pour la seconde partie du projet, l'analyse. C'est le language objet que je maîtrise le mieux et avec lequel je me sens le plus à l'aise. Étant également un des langage les plus populaire, il est aussi plus facile de trouver des bibliothèques utiles au projet.

3.2 HTMLCleaner/Jsoup/OpenCSV

Plusieurs librairies Java ont été nécessaires dans la réalisation du projet. A noter que les librairies utilisées sont toutes open source.
La première librairie utilisée est HTMLCleaner, elle a été utilisée uniquement pour nettoyer le code HTML d'un point de vue "visuel" pour pouvoir se rendre compte de l'organisation de celui-ci.

Jsoup est la librairie qui à le plus servi dans le projet, elle permet d'extraire et de manipuler des données sur du code HTML. Cette bibliothèque offre API simple d'utilisation et très bien documentée ce qui rend son utilisation aisée. Seule la partie concernant l'extraction de donnée à été utilisée mais il faut savoir qu'elle permet également de créer/modifier directement du code HTML.

Enfin, la librairie OpenCSV a servi à la manipulation des fichiers csv sont stockés les informations récupérées. C'est une librairie simple dont les deux méthodes de sauvegarde ont été utilisées.

4 Récupération des données

Avant d'envisager une quelconque analyse des données, il a tout d'abord fallu récupérer les fichiers nécessaires sur le site de l'assemblée nationale.

4.1 Organisation des fichier sur le site

Sur le site de l'assemblée Nationale, les comptes rendus intégraux sont organisés par législature, par année, et par type de séance. Chaque page d'index est de la forme *http ://www.assemblee-nationale.fr/X/debats/index.asp* où X est le numéro de la législature. Sur cette page sont répertorié les liens vers la liste des séances par année.
Par exemple la hiérarchisation est effectuée comme suis pour la législature 14 :

```
14/debats/index.asp
    14/cri/2014-2015/
        14/cri/2014-2015/20150225.asp
        .
        .
    14/cri/2013-2014/
        14/cri/2013-2014/20140255.asp
        .
        .
    14/cri/2013-2014-extra/
        14/cri/2013-2014-extra/20141027.asp
        .
        .
```

Pour le projet, on commence par la législature actuelle puis on essaye de remonter au maximum dans le temps. En réalité le programme fonctionne jusqu'à la moitié de la douzième législature.

4.2 Choix du format

Deux format de fichiers on été envisagé, le premier était le pdf en téléchargeant toute les versions pdf des journaux officiels puis d'utiliser la librairie Java PDFBox afin de pouvoir les manipuler. Seulement après réflexion et lecture du code html des pages de compte rendu, il a finalement été jugé qu'il était plus facile de travailler avec ce code html en utilisant JSoup. En effet, la retranscription

écrite d'une séance en html suis un schéma clair et répétitif en ce qui concerne les intervention, du moins pour les législatures les plus récentes.

4.3 Téléchargement des données

Le programme qui effectue cette récupération est écrit en Scala, il comporte deux fichiers : *URLManager.scala* et *HTMLDownloader.scala*.
Le premier fichier contient plusieurs fonctions qui vont permettre de stocker dans une variable toutes les URL pointant vers une page html contenant le compte-rendu d'une séance pour un nombre de législature fixé.

Sur le site de l'assemblée nationale, on commence à partir de l'URL *http ://www.assemblee-nationale.fr/X/debats/index.asp* où X est le numéro de la législature. À partir de cette page, on récupère tout d'abord les liens qui pointent vers la liste des compte-rendus d'une année (ex :*http ://www.assemblee-nationale.fr/14/cri/2013-2014/* pour les comptes-rendus de la 14e législature et pour l'année 2013-2014).

Une fois que toute ces URL sont récupérées, le même procédé est appliqué avec les nouveaux liens afin de récupérer les liens des pages contenant les comptes rendus html (ex : *http ://www.assemblee-nationale.fr/14/cri/2013-2014/20140255.asp*).

Une fois cette première partie effectuée, une des deux fonctions *downloadAll* et *downloadGroupNb* du fichier *HTMLDownloader.scala* vont se charger respectivement de télécharger tout les fichiers html correspondant aux URL précédemment récupérées ou alors de télécharger un nombre de paquet de 100 donné. Le classement se fait par législature, années, et type de séance.

5 Analyse et extraction des données

5.1 Agencement des fichiers HTML

Après une lecture de plusieurs fichiers compte rendu, est possible de se rendre compte de la façon dont ceux-ci sont agencé. En effet a première vu, les fichiers peuvent paraître brouillon mais, grâce à la ré-indentation faite à l'aide de HTMLCleaner l'analyse visuelle à pu être facilité et il a été possible de se rendre compte que les fichiers suivaient en fait un schémas répétitif plus ou moins convenaient à la localisation des informations utiles pour le projet. Tout l'intérêt du html est de pouvoir avoir un document donc l'organisation du fond peut être faite de façon minutieuse, c'est de celà que nous voulons tirer profit ici.

5.2 Différences notables

Bien que les fichiers suivent un schémas similaire, ils ne sont pas tous identiques pour autant. En effet les plus récents sont les plus "propres" et également les plus claire : le code html utilise des class utiles dans les balises qui indiquent clairement le contenu ("intervention", "somprésidence"...), les intervenant ont un lien vers une fiche d'identification et chaque paragraphe correspond à l'intervention d'une personne différente. (ex Figure 5.1)

FIGURE 5.1 – 2015

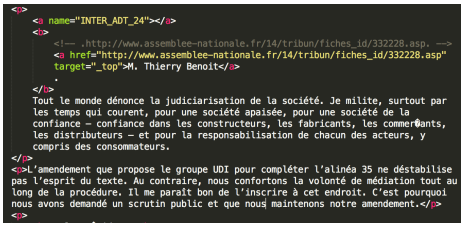

FIGURE 5.2 – 2013

Cependant dés que l'on commence à remonter un peu dans le temps, la présentation diffère et devient de moins en moins claire, les class disparaissent, le schémas n'est parfois pas respecté dans un même fichier ou d'autre modifications (ex Figure 5.2). C'est pourquoi il a fallut faire en sorte que le code du programme s'adapte en fonction du fichier à traiter.

5.3 Extraction des informations

Le processus d'extraction d'informations se fait via la classe *TalkAnalyser*, c'est elle qui s'occupe de filtrer et rechercher les informations nécéssaires dans les fichiers puis de les sauvegarder. Une fonction main parcours l'arborescence des fichiers html et appelle la fonction *process* de la classe *TalkAnalyser* pour chaque fichier, c'est la méthode qui se charge de récupérer toute les informations et se déroule comme suit :

— lit le fichier : récupère le contenu html du fichier sous forme d'une chaîne de caractère.

— récupère les données : appelle la fonction *getData()* qui va se charger de récupérer toutes les informations nécessaires de la bonne manière en fonction du fichier.

— sauvegarde : enregistre les données récupérées dans des csv.

De manière plus détaillée, la fonction *getData()* utilise la librairie Java Jsoup afin de filtrer le code html et extraire les informations. Il faut tout d'abord parser le code html lu précédemment, cela retourne un objet Document qui est une représentation hierarchique du code html sur lequel Jsoup est capable d'effectuer diverses recherches/filtres.

Ce filtrage s'effectue en utilisant la syntaxe CSS en temps que "requête", tout les éléments qui correspondront à la spécification CSS seront retournés sous forme d'*Elements* . Par exemple pour récupérer toute les interventions d'un fichier récent (div "intervention"), il suffit d'utiliser la fonction select sur l'objet Document comme suit : *doc.select("div.intervention > p")*, il est ensuite possible d'itérer sur chaque paragraphe récupéré afin de les traiter séparément et d'en extraire intervenant, intervention et réaction si présentes.

6 Stockage et forme des données

7 Mode d'emploi

Conclusion

Lors de ce projet, j'ai tout d'abord eu l'occasion de mettre mes compétences en informatique au service d'un domaine qui n'a à priori aucun lien, celui de la Sociologie. Cela à permis de changer des projets habituels moins originaux, j'ai aussi pu lire quelques compte-rendu de séance de l'assemblée nationale très intéressant. D'un point de vue plus technique, j'ai pu apprendre à récupérer fouiller un site internet afin d'y récupérer des documents.
Ensuite j'ai également appris à me servir de plusieurs librairie Java, en particulier JSoup que je ne connaissais pas et qui se révèle être très puissante et intéressante, il est probable que je la réutilise dans d'autre contexte. En résumé, je suis entièrement satisfait du sujet proposé et j'ai pris plaisir à travailler dessus, en espèrent qu'il conviendra à l'utilisation qui en sera faite.

Bibliographie