École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
4e année
2012 - 2013

Rapport de Stage de 4e Année

Participation au développement de la base de données linguistique du LLL.

Encadrant
Marjolaine Martin
Enseigant-Chercheur
Laboratoire Ligérien de Linguistique
marjolaine.martin@univ-tours.fr

Étudiant
Simon Kesteloot
simon.kesteloot@etu.univ-tours.fr

DI4 2012 - 2013

Université Francois-Rabelais

Version du 16 septembre 2013

Table des matières

Remerciements

Pour le bon déroulement de ce stage, je remercie Valentin Doulcier, précédant contributeur du projet qui m'a aidé a appréhender le fonctionnement de l'application et la structure de la base de donnée. Je remercie Mme Tacquard qui a proposé et encadré ce stage et Mme Martin faisant partie de l'équipe bénéficiaire du projet qui m'a aidé à comprendre les données à traiter.

Introduction

Le laboratoire ligérien de Linguistique étudie la langue anglaise et ses variations. Pour ce faire il se base sur des dictionnaires. Hors, la recherche dans un dictionnaire papier, quand elle ne se base pas sur l'ordre alphabétique est rapidement longue et très peu productive. Pour simplifier et rendre ces recherches moins fastidieuses, le laboratoire a les versions informatisées de différents dictionnaires de référence. Il manque encore le logiciel qui permettra de traiter ces données facilement pour l'utilisateur, c'est le but de ce projet. Ce projet a été initié en 2009 puis refondu durant l'année 2012-2013 par Valentin Doulcier. Les fonctionnalités de base sont mises en place, mais les données ne sont pas encore intégrées dans le logiciel. Un stage a donc était proposé pour intégrer ces données dans la base de données du logiciel. Ce rapport décrit ce stage en commençant par présenter le laboratoire, ensuite la deuxième partie décrira le projet, pour finir, la réalisation du stage avec ce qui a été réalisé ainsi que les problèmes rencontrés.

Le laboratoire ligérien de Linguistique

3.1 Composition

Le laboratoire ligérien de linguistique (LLL) est une unité de recherche regroupant 1 chargé de recherche, 39 professeurs et maîtres de conférences, 8 ingénieurs et techniciens ainsi que 5 conservateurs et 34 doctorants répartie principalement entre l'université François Rabelais de Tours, l'université d'Orléans, la bibliothèque nationale de France (BnF) à Paris. De plus, une vingtaine de personnes français et étrangers se trouvant dans d'autres structures, s'ajoutent au compte total de collaborateurs participant au laboratoire. Cette diversité est regroupé sous une unité mixte de recherche (UMR 7270), donc financée par le CNRS et les universités.

3.2 Activité

L'activité de ce laboratoire est basée sur le traitement et l'exploitation du corpus oraux en couvrant toutes les disciplines de la linguistique : phonologie, morphologie, lexicologie, phonétique, sémantique, pragmatique, syntaxe et analyse de discours.

Le LLL participe activement à la description d'un grand nombre de langues, du créole à base portugaise à différentes langues subsaharienne ainsi que des dialectes Guyanais. Il poursuit aussi des travaux sur l'arabe, le berbère, le vietnamien, le japonais, le chinois, le français, l'anglais, l'allemand, l'espagnol...

3.3 Publication

Le laboratoire publie la Revue de Sémantique et Pragmatique [1].
Il publie aussi moins régulièrement Les Cahiers du PROHEMIO.

3.4 Lien avec le stage

La correspondante du LLL pour le stage est Marjolaine MARTIN, maitre de conférence à l'université de Tours et directrice des études du département Mundus de Polytech Tours.

Le stage ne s'est pas passé dans les locaux de LLL mais au département informatique de Polytech Tours, à cause du manque de place dans le laboratoire.

1. http://www.lll.cnrs.fr/RSP/

Le Projet

4.1 Finalité

Le projet a pour but de créer un logiciel à destination des chercheurs et des doctorants. Il permettra de faire des recherches avancées dans différents dictionnaires.
Ces dictionnaires sont stockés dans une base de données hébergée sur le poste client, au même titre que l'application.
Les dictionnaires sont des dictionnaires commerciaux anglais possédés par le laboratoire.

4.2 Historique

Une première version du logiciel a été initié en 2009 par Gwénael Boissay alors employé par le LLL pour cette tâche. Le programme était composé de 5 composants logiciel exécutables séparément ayant chacun une tâche précise.
2 projets de fin d'étude consécutifs ont ensuite pris la main sur le projet afin de principalement refondre la base de données. La première version ne pouvait accueillir que les 3 dictionnaires possédés par le LLL. Le but était donc d'avoir une base de donnée plus générique et modulaire, car d'autres dictionnaires sont prévus à l'avenir.
Suite à ça, un troisème PFE mené par Valentin Douclier en 2012 devait recréer le logiciel qui fonctionnera autour de cette base de donnée. Les logiciels déjà existant, difficilement évoluables ont mené à la reconception complète du programme.

4.3 Structure du programme

L'application est composée de 3 parties distinctes :
— le logiciel client à développer
— la base de données locale
— la base de données principale

4.3.1 Base de données principale

La base de données principale a pour but de regrouper toutes les données du laboratoire.
Tous les chercheurs et doctorants ont un accès en lecture sur cette base, mais seul Jean-Michel Fournier, le directeur-adjoint du laboratoire et administrateur de l'application, peut écrire les dans cette bases.
Le logiciel devra pousser les demandes de modifications sur le serveur, qui seront validés ou non par l'administrateur.
Cette base de donnée est hébergé sur les serveurs du CNRS.

4.3.2 Base de données locale

La base de données locale est une copie de la base de données principale.
Cette base est une copie de la base de donnée principale, mais l'utilisateur locale a les droits d'accès en

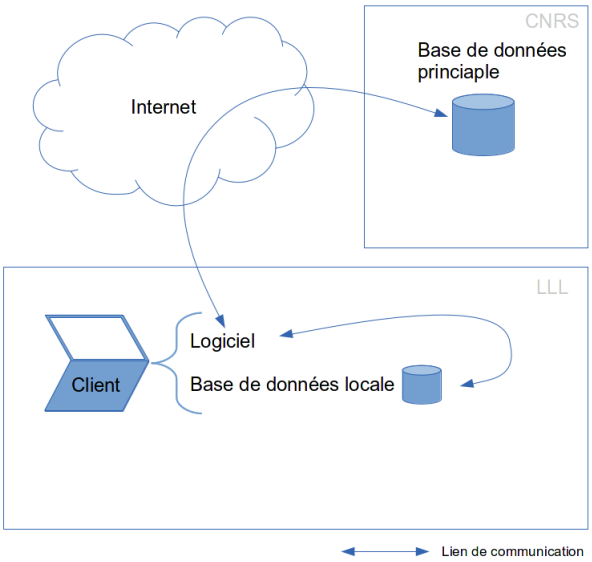

FIGURE 4.1 – "structure du système"

écriture dessus. elle sert de cache vers la base de données principales et de sauvegardes des modifications. Le fonctionnement normale est que cette base existe sur chaque poste client utilisant le logiciel, mais à ce jour, il est possible d'utiliser une unique base pour plusieurs machines, de façon collaborative car c'est un logiciel-serveur à part fait fonctionner ce sous-système.

4.3.3 Logiciel client

Le logiciel permet d'accéder à la base de donnée locale et dans une moindre mesure, à la base de données principale.

Fonctionnalité

Le logiciel est divisé en 4 parties, séparant les fonctionnalités.

"Recherche" C'est la partie la plus importante et qui sera normalement la plus utilisée. Elle permettra de faire des recherches simples comme chercher une chaine de caractère dans la liste des mots ou leurs définitions ou des recherches complexes, équivalents au expressions régulières ou la fouille dans la prononciation...
À ce jour, uniquement la recherche simple est implémentée. De plus, un historique est mis en place pour permettre d'accéder facilement au résultat précédeSans nom 1nt à des fin de comparaisons ou autres.

"Synchro" La partie Synchro aura pour tâche de permettre la synchronisation des bases de données locales avec la base de données principales.
Chaque client a sa propre base de données installé localement. Lors de modification, annotation des données. il est possible de pousser ("PUSH") ces données vers une base principal hébergé sur les installations du CNRS.
Chaque modification sera validé par l'administrateur du système Jean Michel Fournier, avant d'être intégré à la base commune.

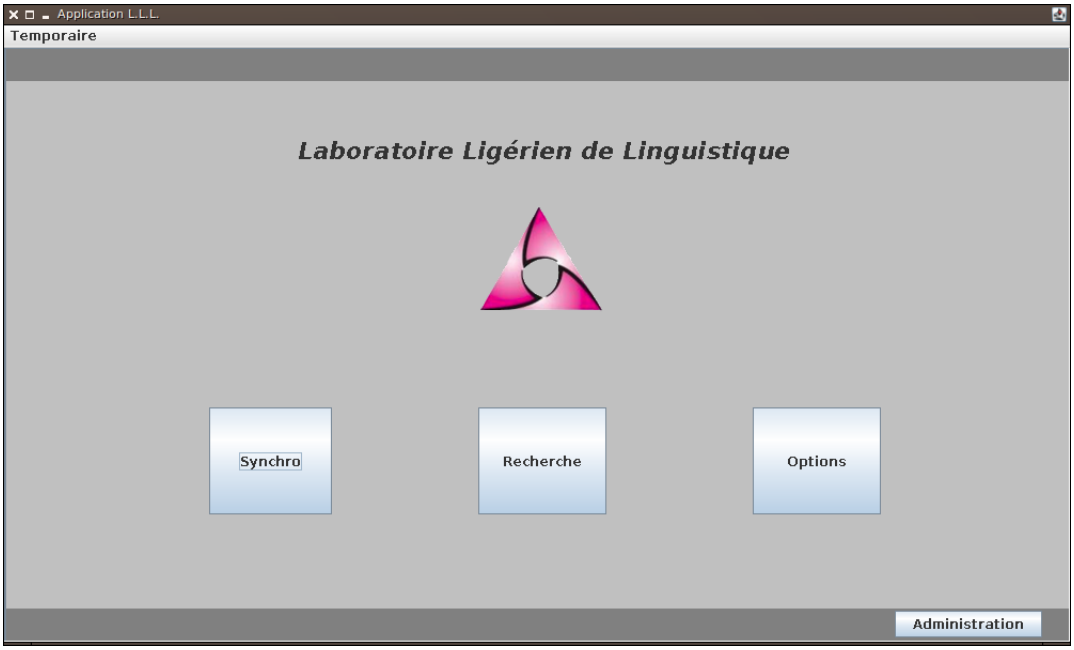

FIGURE 4.2 – "écran d'accueil de la nouvelle version du logiciel"

Il sera aussi possible de récupérer ("PULL") les données à jour ou de remettre à zéro ("RESET") la base de données locale.

"Options" Une dernière partie, "options" permet de uniquement de changer la configuration pour l'accès à la base de donnée locale, c'est à dire l'url, le login et mot de passe.

"Administration" L'administration est accessible uniquement aux personnes ayant un compte identifié comme administrateur.
Cette section permet principalement d'importer de nouveaux dictionnaires dans la base de données, de modifier directement le contenu de la base de données, de modifier les la configuration d'accès à la base de données principal et d'exécuter directement des requêtes SQL.

Structure

La structure principale du logiciel est visible figure 4.2, La séparation des fonctionnalités dans l'interface a servi de base pour l'organisation du code source. Chaque partie est composé de la gestion de l'interface et du contrôleur, Ensuite une couche "données" gérées par un ORM [1] qui fait l'interface entre le contrôleur et la base de données. Le logiciel est construit sur le modèle MVC [2] . Les composants communiquent uniquement aux composants qui leurs sont juxtaposés.

4.4 Dictionnaire

Les dictionnaires possédés par le laboratoire sont 3 dictionnaires de référence dans leur domaine :
— The Cambridge English Pronouncing Dictionary (EPD) : spécialisé dans la prononciation britannique.
— The Longman Pronouncing Dictionary (LPD) : spécialisé dans la prononciation américaine.

1. Object Relational Mapping, https://fr.wikipedia.org/wiki/Mapping_objet-relationnel
2. Modèle-Vue-Contrôleur est un patron de conception, séparant l'interface, l'accès aux données et le contrôleur https://fr.wikipedia.org/wiki/Modèle-Vue-Contrôleur

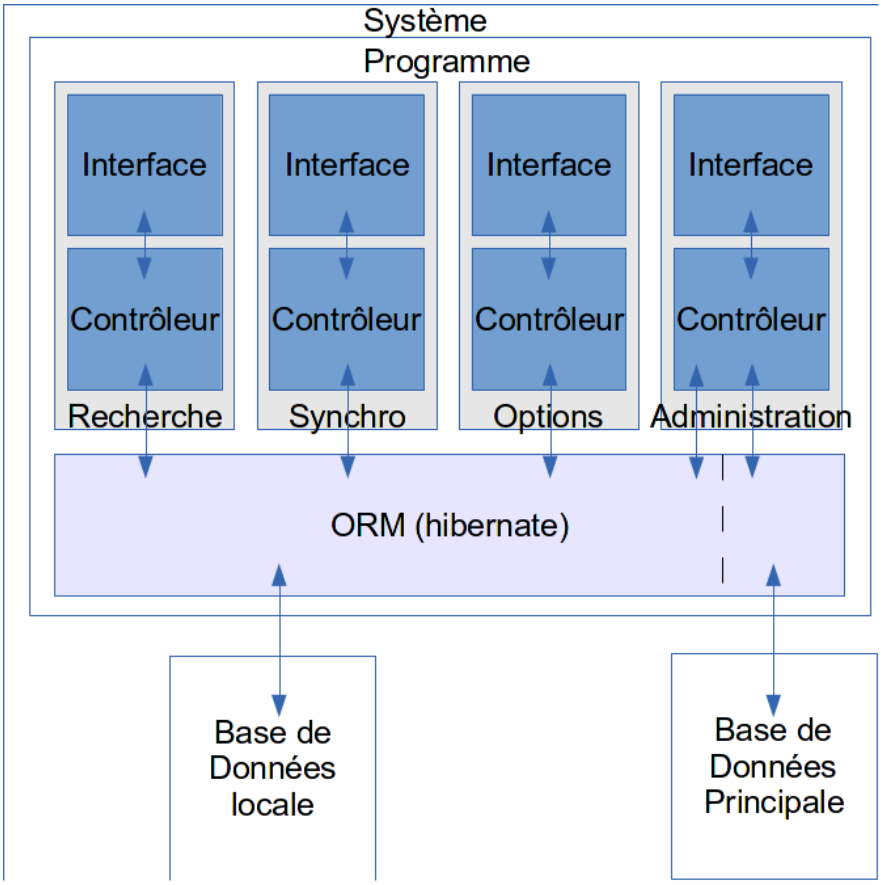

FIGURE 4.3 – "structure interne du logiciel"

— The Macquarie Dictionary (MCQ) : spécialisé dans la prononciation Australienne, le plus complet des trois sur les aspects généraux (définitions, mot composés, flexions...)

4.5 Technologie utilisé

Le programme doit être multi-plateforme car les chercheurs sont équipés de machines fonctionnant sous les systèmes d'exploitation Mac OS et Windows, le langage retenu est dès plus classique, mais répond bien au contraintes, Java SE 6.
Ensuite pour le stockage et l'accès aux données, une base de données relationnelle classique est utilisé, MySQL qui est donc déployé sur chaque client, car ce SGBD est utilisé depuis le départ du projet, c'est le plus répandu et il fonctionne sur un grand nombre de plate-forme.
Pour simplifier l'accès à la base de données depuis le programme, le framework Hibernate est utilisé, C'est un ORM , il permet de créer un ensemble de classe copiant la structure de la base et de faire des requêtes à cette base sans utilisé directement le langage SQL, renvoyant ainsi des objet qui reflète le contenu de la base.
Tout ces choix n'ont pas de licence à payer car elles sont toutes plus ou moins open-source [3].

4.6 But du stage

Le logiciel a déjà une version basique qui permet de faire des recherches simples dans la base de données locale mais aucune information réelle n'y est stocké.
Donc avant de pouvoir l'utiliser, il faut importer les dictionnaires. Le but du stage est donc de créer le module d'importation de données du logiciel permettant d'acquérir les trois dictionnaires possédés par le laboratoire ligérien de linguistique.

4.7 Remarques sur le projet

Le projet est clair, bien structuré et bien documenté. Il est juste a noté que quelques petits points sont améliorables dans ce qui est déjà réalisé.

Documentation base de données Bien que le reste est documenté, la base de donnée, comme bien souvent n'as pas de note ou de documentation. La plupart du temps ce n'est pas nécessaire car le nom des tables et des colonnes est assez explicite et sans ambiguïté pour quelqu'un habitué à l'informatique et au développement.
Mais dans ce projet, c'est une base de donnée de linguistique, il faut donc un bagage de linguistique et un bagage informatique pour pouvoir qu'elle prenne tout son sens.
Il faut donc documenter la base de donnée pour permettre au moins à un informaticien de la comprendre. Ce manque a été corriger lors du stage.

Sécurité des mots de passe Bien que la base de données ne devrait pas être divulgué à n'importe qui, il est important de bien sécuriser les mots de passes stocké dedans, bien plus encore que tout autres informations personnelles ou critiques.
Hors ici la méthode de sécurisation choisi est une méthode classique de hachage [4] par l'algorithme MD5.
Cette algorithme est encore trop souvent utilisé car il a été "cassé", il est possible aujourd'hui de récupérer une clé à partir de l'empreinte, sans de voir faire une attaque par force brute.

3. Les machines virtuelles Java nécessaires au fonctionnement du programme sont le plus souvent des logiciels propriétaires sur les système d'exploitation ciblé mais reste gratuit

4. génération d'une suite d'octet, une Empreinte, à partir d'une clé (le mot de passe) qui authentifie la clé, se processus est en théorie impossible en sens inverse (retrouvé la clé à partir de l'empreinte. https://fr.wikipedia.org/wiki/Hashage

Il est vivement recommandé d'utiliser d'autres algorithmes tel que SHA256 ou SHA512, voir des plus exotiques comme Whirlpool ou Keccak, autant testé, mais moins utilisé ce qui diminue les nombres d'attaques dessus [5].

Choix de licence À ce jour, le code source du logiciel est disponible via la plateforme github.com, il est donc open-source (puisque le code source est volontairement public) mais aucune licence n'a été choisi. Une licence est un contrat via lequel le titulaire des droits d'auteurs du logiciel défini les possibilités et les conditions de réutilisations de tout ou d'une partie du logiciel [6].
Il est d'usage de fournir une licence afin d'informer qu'elles sont les droits et les devoirs dans le cas de l'utilisation du logiciel et/ou du code source.

Système de gestion de base de données Le SGBD utilisé à ce jour est un SGBDR soit une système de gestion de base de donnée relationnel [7]. C'est le type de base de données princialement utilisé pour tout les domaines. Hors depuis quelques années d'autres systèmes réapparaisse, regroupé sous le nom de NoSQL [8] Dans ce projet, une base de donnée orienté document [9] est surement plus adapté aux données à stocké car toutes les informations sont pour un mot.
La différence entre une bdd relationnel et une bdd orienté document est assimilable à la différence entre le typage statique et dynamique d'un langage de programmation. Le type dynamique est plus souple et permet de développer plus rapidement mais plus lent lors de l'exécution du programme. Inversement pour le typage statique. Une base de données orienté document est de la même façon plus souple car elle n'impose pas de format de données bien précis, mais contrairement au typage dynamique, les SGBD orienté document ont aussi beaucoup moins de limitation, de verrou et de vérification des formats des données, il sont donc aussi plus rapide.
Une bdd orienté document permet d'avoir toutes les informations à un mot dans un seul enregistrement (sous la forme d'arbre), un "document", alors que dans le schéma courant prévu pour une base de données relationnel, il faut rechercher des données dans 26 tables sur les 27 présentes dans le modèles.

5. Dans l'absolu SHA256 est a évité aussi car il existe depuis récemment des machines qui dispose d'accélération matérielle pour cette algorithme, ce qui diminue fortement le temps et l'énergie nécessaire pour une attaque par force brute dessus.
6. plus d'information sur les licences : https://fr.wikipedia.org/wiki/Licence_logicielle
7. organisation des données autour de tables reliés entre elles par des clés https://fr.wikipedia.org/wiki/Base_de_données_relationnelle
8. NoSQL signifie Not Only SQL.
9. https://en.wikipedia.org/wiki/Document-oriented_database

La réalisation du stage

Le stage s'est déroulé en trois parties, l'analyse des dictionnaires et de la base de donnée afin de trouver les correspondance pour pouvoir réaliser l'importation, ensuite l'analyse et les choix de la méthodes d'importation et pour finir le développement de l'importateur fondé sur les analyses et la méthode choisie.

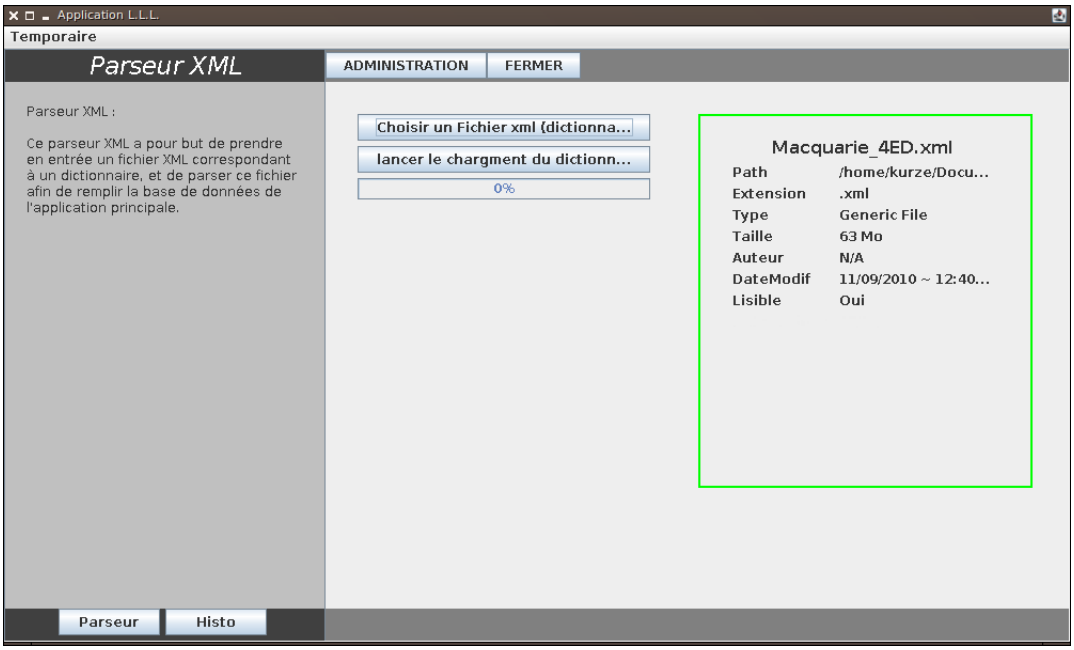

FIGURE 5.1 – "écran de la partie importation de dictionnaires"

5.1 Analyse

Les dictionnaires sont des fichiers XML, donc sous la forme d'arbre. Une personne précédente sur le projet avait déjà fait ce travail d'analyse mais le résultat n'était pas disponible et de plus utile uniquement pour l'ancienne version de la base de données. Il a donc fallu refaire cette analyse pour la nouvelle base de données de l'application.

Pour chaque fichier, quelques mots on était extraits afin de décomposer et séparer les données contenus. Un fichier XML est un arbre ou le document est le noeud racine et chaque couple de balise ($<\ldots>$ et $</\ldots>$) est un noeud. Chaque sous-arbres extrait a été analysé afin connaitre l'utilité de chaque noeud et sur les noeuds retenu, les correspondances noeud XML - colonne base de données ainsi que les traitements à appliquer si nécessaires.

Les dictionnaires, comme la base de données, étant très complets, donc composé de beaucoup de type de noeuds différents c'est une étape longue et d'autant plus importantes car le logiciel ne sera pas utile si uniquement la moitié des informations sont accessibles. De plus, la base de données et les dictionnaires

ont des structures et des champs très différents, il y a donc beaucoup de "routage" et de traitement à faire pour transformer les données, d'un format à l'autre.

Cette étape a permis de générer trois graphiques représentant la base de données et les dictionnaires ainsi que les liens entres chaque champs. Ces schémas ont servi de base pour la validation par Mme Martin.

5.2 Développement

Pour le développement du module, la contrainte principale était la généricité. A l'avenir, de nouveaux dictionnaires devrait être ajouté au systèmes. Afin de simplifier l'insertion des nouveaux dictionnaires, la solution pour l'importation doit être aussi générique que possible.
Suite au grand nombre de traitements différents, et aux éventuelles nouveaux traitements pouvant apparaitre sur les nouveaux jeux de données, il n'est pas possible d'avoir une solution totalement générique car les 3 dictionnaires utilisé pour le moment utilise chacun une structure propre et sont très peu semblable. Les différentes solutions qui ont été proposées sont :

5.2.1 générique pauvre (sans descripteur)

La première solution, certainement la plus simple, est de créer un modules d'extraction pour chaque dictionnaires qui se base sur module généralistes contenant les méthodes, attributs qui peuvent être mis en commun. C'est la méthode classique la plus classique.

5.2.2 générique riche (descripteur simple)

La deuxième solution est de créer un moteur d'extraction générique qui s'appuie sur un descripteur pour chaque dictionnaire à importer. Dans ce cas une analyse préalable à l'importation est indispensable pour trouver les correspondances avec la base de données et créer ce descripteur, comme à chaque fois.

Hors après la première réunion avec Marjolaine Martin, il s'est avérée que il ne suffit pas de d'importer les données pour chaque noeud du fichiers XML, mais dans beaucoup de cas, un traitement supplémentaire est à prévoir, comme diviser ou concaténer certains champs, dupliquer certaines entrée dans plusieurs endroits de la base de données. Hors tout ceci impose un système de descripteur très complexe pour savoir gérer tout les cas. De plus il serait adapté au trois dictionnaires courant mais il manquera très probablement des possibilités pour les futurs dictionnaires. Ce qui imposera des modifications du logiciel et donc perdre l'intérêt de la généricité.

5.2.3 générique riche (descripteur évolué)

La dernière solution proposé est un système générique qui s'appuie sur un autre logiciel qui fait l'extraction à partir du dictionnaire et du descripteur, mais celui-ci devra être dans un langage dynamique, (javascript,...) qui permet d'importer des fonctions à l'exécution à partir d'un fichier de données. Le but est que le descripteur (par exemple au format JSON) est un ensemble de clé-valeur ou la clé sera le nom du noeud du fichier XML et la valeur, la fonction en javascript à effectuer. Dans ce cas là, c'est le descripteur qui contient "l'intelligence".

La solution Javascript est apporté car le langage Java utilisé pour le fonctionnement du programme ne permet pas d'importer des fonctions/méthodes à l'exécution du programme.

C'est un système "exotique" qui peut être difficile à utiliser si mal conçus et compliquer à faire pour être parfaitement utilisable.

C'est la solution "générique pauvre" qui a été retenu par l'encadrement afin d'avoir un logiciel plus simple et plus maintenable.

5.2.4 module commun

Le module commun comprend toutes les méthodes d'ouverture et de parsage des fichiers ainsi que la navigation dans l'arbre, la recherche de noeud et la conversion des caractères spéciaux.

5.3 Problème rencontré

5.3.1 Analyse des échantillons

Lors du développement, beaucoup d'exception était déclaré lors de l'analyse de chaque mots, il s'est avéré que elles étaient dû à des problèmes d'analyses. Seulement un échantillon a été pris en compte, hors les données fournies pour chaque mot ne sont pas les mêmes, dans champs apparaissent, d'autre disparaissent à chaque entrées. Les mots n'ont pas touts les champs renseignés et les champs non renseignés ne sont pas fourni. Il a donc fallu élargir l'échantillon et augmenter le nombre de test pour éviter d'autre problèmes de ce type.

5.3.2 Disponibilité

Quelque petit soucis de disponibilité, de la part des encadrants du stage ont existé durant le début de celui-ci, dû au rendu des notes, les différents conseils de classes et à la finalisation de la maquette pédagogique. Ceci a légèrement ralenti le déroulement durant 1 à 2 semaines.

5.3.3 peu de documentation

Au début du stage Valentin Doulcier, a présenté tout le programme, ainsi que la base de données. Mais, comme précisé précédemment, la base de donnée n'était pas documenté, ainsi à chaque incompréhension, il était essentiel de lui demander directement, ralentissant ainsi l'avancement des tâches.

5.4 réalisé / non réalisé

5.4.1 Analyse

L'analyse des trois dictionnaires est réalisé.

5.4.2 Développement

Le développement de la base commune est terminé.
Le développement du module pour le dictionnaire Macquarie, le plus complet est normalement terminé.
Le développement des modules suivant n'est pas entamé.

5.4.3 Documentation

La documentation de la base de données manquante est rédigée et est en attente de validation par Valentin Doulcier.
La documentation du code est faite pendant le développement, intégré au code et suivant la norme Javadoc.

Conclusion

Ce stage m'a permis de découvrir le développement Java, dans un projet réel, la puissance du langage et des bibliothèques existantes. Mais il aussi confirmé que je ne suis pas fervent de la programmation orienté objet, préférant la programmation impérative et modulaire comme le C ou le Go qui sont plus facilement assimilable pour moi.

J'ai aussi découvert le framework Hibernate qui permet en théorie de simplifier l'accès au base de données relationnelle à partir du programme Java. Ayant fait quelques développement PHP/MySQL auparavant je préfère tout de même l'utilisation de requêtes SQL plutôt que d'appel à fonction au nom pas toujours explicite qui cache une bonne partie du fonctionnement et alourdi sensiblement le démarrage et le déroulement du logiciel. Préférant savoir exactement ce que je fais et plutôt adeptes de micro-framework et petites bibliothèques, accélérant et simplifiant le développement rapidement, je trouve que Hibernate est trop lourd et long à prendre en main pour être utile dès la première utilisation.

Pour finir, le langage de balisage XML utilisé dans les dictionnaires, ne permet le traitement automatique des données. La lecture des données doivent être entièrement décrite lors du développement ce qui rallonge considérablement l'étape de codage dans le cas présent. Je préfère largement le format Json crée en 2006, permettant les même chose que le XML mais en beaucoup moins verbeux et ajoutant la sérialisation/desérialisation entre autres. Ce format permet l'importation et l'exportation automatique, ce qui aurait considérablement accéléré le développement.

Participation au développement de la base de données linguistique du LLL.

Département Informatique
4e année
2012 - 2013

Rapport de Stage de 4e Année

Résumé : Ce stage a pour but de concevoir un module d'importation un ensemble de dictionnaires XML dans une base de donnée relationnelle

Mots clefs : Java, Hibernate, LLL, XML

Abstract: This internship aims to design an import module of a XML dictionary to a relational database.

Keywords: Java, Hibernate, LLL, XML

Encadrant
Marjolaine Martin
Enseigant-Chercheur
Laboratoire Ligérien de Linguistique
marjolaine.martin@univ-tours.fr

Étudiant
Simon Kesteloot
simon.kesteloot@etu.univ-tours.fr

DI4 2012 - 2013

Université Francois-Rabelais