Rapport de laboratoire

Compréhension des langues naturelles

Cours	LOG635
Session	Été 2015
Équipe	03
Laboratoire	02
Chargé de laboratoire	Richard Rail
Professeur	Lévis Thériault
Étudiant(e)(s)	Moreau Max Mathieu St-Jean Champion François-Guy Gallant Steven Smith
Adresse(s) de courriel	max.moreau.1@ens.etsmtl.ca ak20290@ens.etsmtl.ca fggallant@gmail.com ac29620@gmail.com
Code(s) permanent(s)	MORM30038905 STJM31109005 GALF10038504 SMIS18027507
Date de remise	7 Juillet 2015

Tables des Matières

1 INTRODUCTION

Pour ce 2e laboratoire de LOG635, nous allons être initiés au traitement automatique des langues naturelles en utilisant une boite à outils NLTK afin d'atteindre un niveau intermédiaire d'efficience et de compréhension. Nous devons :

- rédiger une grammaire de type FCFG;
- faire la distinction entre l'analyse lexicale et syntaxique;
- manipuler des expressions Lambda;
- utiliser un analyseur de type diagramme.

Le tout encapsulé par un programme Python. L'application devra permettre l'analyse d'un texte prédéfini et en déduire des faits sous un format Jess. L'analyse du texte devra être sans ambiguïté (une seule interprétation) et présenter une certaine flexibilité afin d'accepter de nouvelles phrases employant le même vocabulaire et ses variantes du texte original.

Ce document est scindé en 5 parties : introduction, prétraitement, complexité, post-traitement et conclusion. À travers ces sections, nous allons préciser le fonctionnement de l'application et expliquer comment nous avons adressé chaque point demandé.

2 PRÉTRAITEMENT

Le prétraitement constitue toutes les étapes que nous effectuons avant d'appliquer notre fichier de grammaire avec NLTK. Ce prétraitement est effectué en Python.
Les phrases à analyser sont contenues dans le fichier 'Einstein.txt'. Nous effectuons notre traitement ligne par ligne, puis enregistrons les phrases obtenues dans une liste que nous transférons à NLTK.

Le premier traitement que nous effectuons est de vérifier si chaque ligne du fichier est un commentaire ou une phrase à analyser. Lors du développement il nous a été utile de modifier Einstein.txt pour en faire un dépôt avec de nombreuses phrases à tester, un peu comme si l'on roulait des tests unitaires. Lorsqu'il n'était pas nécessaire de tester toutes les phrases, leurs lignes étaient « commentées » en y rajoutant un symbole dièse (#) au début. Notre programme ignore les lignes qui commencent par un symbole dièse, et tente de traiter les autres phrases.

Le second traitement est le remplacement des caractères spéciaux comme le retour de chariot (\r), retour à la ligne (\n), barre oblique inversée (\) et autres. Tous ces caractères sont supprimés, car ils poseraient problème pour notre analyse.

Le 3e traitement est la gestion de la ponctuation. Dans le cas de notre laboratoire, nous ne nous occupons pas de la sémantique, mais uniquement de la syntaxe. De ce fait, nous nous sommes permis de supprimer les virgules (,) ainsi que les apostrophes ('). Seuls les points (.) sont traités, car nous l'utilisons pour séparer les phrases obtenues avant de les insérer dans une liste.

Le 4e traitement est un simple retrait des espaces inutiles, notamment en début et fin de phrase, en équivalent d'une fonction « Trim » que nous retrouvons dans plusieurs langages.

Enfin, notre cinquième étape est une normalisation des mots. Lors de cette étape, nous modifions chaque phrase pour qu'elles commencent avec une minuscule. Nous procédons ainsi afin d'alléger notre fichier de grammaire, car cela nous évite de mettre les mots avec et sans majuscules. Toutefois, nous ne modifions pas les mots qui sont des noms propres (on garde la première lettre avec une majuscule), car cela permet d'enlever un peu d'ambiguïté. Certains noms propres se confondent avec des noms communs ou des verbes lorsque la première lettre est minuscule (ex. violette, marie (marrier), maxime).

Finalement, nous effectuons une union de mot qui sont considéré comme un seul mot à l'aide du "_". Les mots "Blue" et "Master" deviendront alors Blue_Master. Dans la version préliminaire du projet, nous réunissons les deux mots qui sont séparés par un espace et qui commencent par des majuscules, mais cela peut causer des problèmes. Nous avons, par la suite, réuni les mots statiquement à l'aide du dictionnaire fourni.

3 COMPLEXITÉ

3.1 SYNTAXIQUE

La syntaxe s'est avérée plus complexe que prévu. Une phrase simple comme « Nicolas mange une pomme » est facile à interpréter, mais une phrase avec trois propositions incluant de multiples adjectifs et adverbes est une autre paire de manches.

Cela dit, nous sommes tout de même en mesure d'analyser des phrases contenant les constructions suivantes :

- Sujet (**Nicolas**)
- Verbe (Nicolas **mange**)
- Complément d'objet direct (Nicolas mange **une pomme**)
- Adjectif utilisé comme épithète (Nicolas mange une pomme **rouge**)
- Complément d'objet indirect (Nicolas mange une pomme délicieuse **sur la table**)
- Verbe d'état (La pomme **est** rouge)
- Adjectif utilisé comme attribut (La pomme est **rouge**)
- Conjonction « et » entre deux adjectifs (La pomme est rouge **et** délicieuse)
- Conjonction « et » entre deux propositions (La pomme est délicieuse **et** le ciel est bleu)

Par ailleurs, les verbes, les noms communs et les adjectifs ont également une dimension de genre et de nombre, pour s'assurer d'une conjugaison correcte. Le temps de verbe est également considéré, mais seul le temps présent est implémenté.

On peut penser quelques instants à d'autres constructions qui n'étaient pas demandées dans l'énoncé et faire des hypothèses sur leur complexité :

- Les listes avec deux points (« Il existe deux sortes de monotrèmes : les ornithorynques et les échidnés ») auraient été possiblement assez faciles à implémenter, sachant que la partie à gauche est une phrase valide en elle-même, et la partie à droite est une liste.
- Les citations qui utilisent des guillemets (« Bob a dit : « Sophie mange au restaurant » ») peuvent être analysées de la même manière que deux propositions séparées d'un *que* (« Bob a dit que Sophie mangeait au restaurant »). La logique plus loin sera par contre intéressante : veut-on considérer que Bob pourrait mentir?
- Beaucoup plus d'expressions doivent être prétraitées. Une phrase comme « N'en déplaise au ministre de l'Économie, l'augmentation du taux de chômage est bien réelle » est un cauchemar si l'on interprète « n'en déplaise » comme trois mots contenant un verbe, alors qu'il s'agit en réalité d'une seule unité syntaxique.

Pour conclure cette section : même si le texte est en français, peut-être n'est-il pas nécessaire que la représentation interne des mots soit entièrement en français. Par exemple, si l'on force qu'un verbe soit effectué par un sujet, nous arriverons à un mur devant des phrases comme « il pleut » ou « il est neuf heures ». Si l'utilisation d'un sujet nul brise la logique interne, peut-être peut-on représenter ces faits par des formules empruntées à d'autres langues, comme « la pluie pleut » ou « maintenant est neuf heures ».

3.2 LEXICALE

La première difficulté du lexique en langue française se trouve dans les conjugaisons. Les verbes et adjectifs peuvent prendre plusieurs formes selon le temps de verbe.

Pour ce laboratoire, comme notre vocabulaire était limité, nous avons pu faire une entrée différente pour toutes les conjugaisons de l'adjectif « bleu »; *bleu, bleue, bleus et bleues*. Il en a été de même pour les verbes, qui étaient seulement conjugués à l'infinitif présent; cependant, il est impensable de faire cela pour tous les adjectifs. Si l'on voulait faire un programme qui peut interpréter un texte entier, il faudrait générer toutes les conjugaisons possibles de chaque mot avec de la programmation, soit programmer directement la conjugaison dans l'interpréteur (probablement impossible dans notre cas, puisque NTLK s'occupe de cela).

Les adjectifs invariables (« des yeux émeraude »), que nous n'avons pas regardés, posent également problème. Veut-on les insérer dans la grammaire en quatre copies, donc à la fois comme masculin, féminin, singulier et pluriel? Et encore, voici une question d'une plus grande importance encore : qu'est-ce qu'un adjectif invariable? Si on liste « orange » « topaze » « émeraude » dans une liste d'adjectifs invariables, sous prétexte qu'ils peuvent être utilisés pour indiquer une couleur, on risque d'ignorer la construction « des yeux lapis-lazuli » qui, quoiqu'inusitée, est également correcte.

3.3 SÉMANTIQUE

La sémantique est une autre paire de manches.
Déjà, cette phrase pose problème. Est-ce que la sémantique est *littéralement* une paire de manches? Par ailleurs, puisqu'elle est une *autre* paire de manches, où était la première paire de manches? Était-ce la partie lexicale? On voit que lorsqu'on parle réellement de compréhension d'un texte, on s'enlise rapidement dans une montagne de détails.

Considérons l'utilisation de comme « seulement » ou « uniquement ». Notre programme génère un fait par phrase, alors qu'on peut facilement arguer que les phrases contenant ce genre de phrase devraient en générer deux : par exemple, la phrase « Je suis seulement disponible en après-midi » implique à la fois le fait « je suis disponible en après-midi » et le fait « je ne suis pas disponible en matinée, en soirée ou la nuit ».

Le summum de la complexité se trouve peut-être dans des phrases ayant une implication non mentionnée. La phrase « je n'ai tué personne la semaine dernière » est logiquement vraie toutes les semaines pour la majorité des gens (nous l'espérons), mais il serait mieux d'éviter de dire cette phrase à des inconnus, qui pourront insinuer l'information « il y a eu des semaines où j'ai tué quelqu'un ». Une machine qui voudra communiquer avec des humains le plus fidèlement possible devra sans doute également interpréter ce genre d'insinuation.
Le seul cas que nous traitons dans notre laboratoire est celui des références à travers plusieurs phrases. (ex.: "Jean est à la maison. Il regarde la télé").
Dans ce cas-ci "il" fait référence à Jean vu que c'est l'unique sujet présent. Nous remplaçons donc le "Il" par "Jean" afin d'avoir des faits non couple.

Nous nous estimons heureux de ne pas avoir eu à travailler cela davantage!

4 POST-TRAITEMENT

Cette partie intervient après que nous ayons des phrases analysées dans NLTK.
À ce niveau, nous avons déjà nos faits Jess, mais exprimés sous une mauvaise syntaxe.
Le traitement que nous effectuons ici est effectué en Python afin de transformer ces faits en faits_traité (valide pour la sémantique de Jess).

Soit les exemples suivants représentant les faits obtenus :

E1 : fumer(propriétaire(maison(bleue)),Pall_Mall)

E2 : etre(couleurs(maison),rose(vert,rouge,bleu))

On doit les transformer de la façon suivante pour qu'ils soient valides en Jess

J1 : (personne fumer Pall_Mall avoir chiens)

J2 : (couleurs maison etre (list rose vert rouge bleu))

Les faits Jess valides sont d'un français assez approximatif (qu'on dirait couramment "Banane"). Jess considère que le premier mot après une parenthèse ouvrante est le nom d'une fonction, d'où le fait qu'il nous faut toutes les enlever à moins que l'on veuille faire une énumération en utilisant les listes (comme l'exemple E2).
Bien que Jess utilise plus une notation LISP, on remarque que nous avons mis le verbe entre le couple ((A,B) représente un couple) a la place de la virgule.
La notation LISP standard serait plus du genre (verbe A B), mais cela n'a en fait aucune importance pour la règle Jess qui l'utilisera vu que l'emplacement des différents éléments devra être connu afin de créer la règle. De ce fait, nous avons choisi une notation plus "française" et moins "LISP".

Afin de réaliser ceci, nous avons les 3 règles suivantes :

R1 : (A) => A

lorsque l'élément est seul, sans couple, on élève les parenthèses autour;

R2 : v(A,B) => A v B

Lorsqu'un couple est précédé d'un verbe, on met le verbe entre les 2 éléments puis enlève les parenthèses.

R3 : A(B,C,D,F) => (list A B C D F)

Lorsqu'un couple est composé de plus de 2 éléments (un *tuple*), on crée une liste dans Jess. Les listes Jess ne sont pas séparées par des virgules, mais simplement des espaces.

Le séquencement de chacune de ces règles est assez simple. On applique R1 de façon récursive sur les faits en entrée. Lorsqu'on ne peut plus l'appliquer, on effectue la même séquence pour R2. Une fois l'exécution de R2 terminée, on vérifie qu'on ne peut effectuer de nouveau R1 et ainsi de suite. Lorsque finalement R1 et R2 ne peuvent s'effectuer, on vérifie R3 puis commence l'ensemble de nouveau.
Ceci converge vers un fait sans aucune parenthèse, ce qui signale que la conversion est finie. On encapsule alors notre résultat entre parenthèses et obtient notre fait Jess.

L'ensemble est effectué par les simples expressions régulières, suivies d'une modification de notre *string*.

N. B. Les listes sont marquées de façon temporaire par un caractère ne pouvant se trouver dans les phrases; ici, nous avons choisi la barre verticale (|). Ceci permet de ne pas alourdir l'algorithme par des exceptions, tout en permettant d'avoir des compositions de listes (une sous-liste de liste).

5 CONCLUSION

Ce laboratoire nous a permis de prendre conscience de la complexité et de la richesse des langues, notamment dans l'optique de la vérification. Bien des concepts sont partagés par d'autres langues, mais on a pu aller assez loin pour voir les défis propres à la langue française. La même analyse en anglais aurait nécessité les mêmes concepts de verbe, adjectif, propositions reliées par un « que » (« that »), mais nous aurait permis d'éviter de devoir conjuguer les adjectifs en genre.

Accessoirement, cela nous a permis de réviser nos règles de français; nous n'avons abordé que la partie syntaxique et avions une explosion de cas. Nous avons donc limité l'analyse à un certain groupe de phrases avec des variations possibles. Ces restrictions sont effectuées en prétraitement et post-traitement par un script Python.

L'analyse sémantique serait la prochaine étape bien que celle-ci sera assurément encore plus complexe. Néanmoins, ayant maintenant obtenu des faits Jess, cette analyse pourrait être faite en Jess.