Rapport de projet L3s6 : TNM3A03A : Linguistique de Corpus

Ousseynou GUEYE

May 15, 2018

CONTENTS

CHAPTER 1

DÉFINITION DES OBJECTIFS ET CHOIX DES OUTILS

1.1 Cahier de charge

Il s'agit pour nous de créer un automate en langage python[1] capable de dire si une phrase est comparative ou non[2].

L'automate devra être capable de traiter aussi bien le français que l'anglais.

1.2 Matériaux

Nous avons rassemblé un peu plus de 160 phrases en ligne que nous avons réparties en deux groupes :

- PHRASES DEVANT ETRE VALIDEES. (environ 145).
- PHRASES DEVANT ETRE inVALIDEES (15).

1.3 Choix des outils

Nous avions le choix entre utiliser unitex ou programmer de zéro en utilisant Python.

Nous avons choisi d'utiliser python3, par souci de comprendre le processus au maximum, et de ne pas s'encombrer d'un lourd écosystème.

[1] Python3.
[2] Comparative ou superlative.

CHAPTER 2

ORGANISATION DES DOSSIERS ET DU CODE

2.1 Organisation des dossiers

AVANT EXÉCUTION DU PROGRAMME

- **PROJECT_ROOT/**
  - src/
    - modules/
  - corpus_phrases/
  - from_outside_treetagger/
  - rapport/

APRÈS EXÉCUTION DU PROGRAMME

- **PROJECT_ROOT/**
  - src/
    - modules/
  - corpus_phrases/
    - **resultats/**
  - from_outside_treetagger/
  - rapport/

2.2 Les modules

La partie programmée est regroupée dans le dossier *src*.

code_source/

- main.py
- settings.py
- modules/
  - l1_big_process.py
  - l2_tagging.py
  - l2_transitions.py
  - l2_others.py
  - l3_pos.py
  - l3_table.py

Rappels

De même, nous aimerions préciser la logique de notre mouvement (flow). C'est en la suivant que j'ai pu, de manière systématique naviguer à travers le code.

En plus des messages souvent clairs de l'interprète de commande, je pars toujours du niveau 0 vers le niveau 2.

Au début de chaque fichier.py du module se trouve la **liste des fonctions** qu'il contient.

Description des modules

Niveau 0 : Si ce n'est pour contrôler la présentation, le niveau zéro fut très peu sollicité en cas de bug.

- **main.py** : Contient l'interface. C'est le fichier qui englobe le tout.
- **settings.py** : Pour faciliter la *gestion des liens*, nous avons regroupé les liens vers nos dossiers dans ce fichier. Il ne contient aucune fonction, mais juste des variables.

Niveau 1 : Ce niveau est toujours sollicité, car c'est lui qui coordonne le programme, mais aussi donne les paramètres aux différentes fonctions du niveau 2. C'est le niveau le plus délicat à manipuler.

- **modules/l1_big_process.py** : Ne contient que des *fonctions composées d'autres fonctions* des sous-modules. C'est le seul fichier du dossier modules/ qu'appelle le main. Il sert d'interface entre le main.py et les autres modules.

Niveau 2 : contient toutes les fonctions élémentaires. Quand un problème apparait, il y a de fortes chances qu'un des fichiers de ce niveau doive être vérifié après le niveau 1.

- **modules/l2_tagging.py** : contient toutes les fonctions aidant à *l'étiquettage morphosyntaxique* des mots.
- **modules/l2_transitions.py** : contient l'automate à proprement parler.
- **modules/l2_others.py** : contient diverses fonctions comme celle permettant la creation des dossiers.

Niveau 3 : Ce niveau, le plus bas, est composé de deux fichiers :

- **modules/l3_pos.py** : s'occupe de prendre les réponses de treetagger et de nous rendre une catégorie syntaxique selon des règles que nous avons précisées.

- **modules/l3_table.py** : contient les transitions qui sont utilisées par l'automate.

2.3 Processus

Ce que nous dirons pour un élément (phrases, ... , mot) est valable pour tous ses éléments-frères.

- **-** [x] : NIVEAU GLOBAL
  1. [x] : Création dossier *resultats*.
  2. [x] : Récupération liste des phrases des fichiers *phrases_** où {*} est la langue (en pour english et fr pour français).
  3. [x] : Envoi des données et des paramétres.
- **-** [x] : NIVEAU PHRASES et MOTS
  1. [x] : Découpage de la phrase et récupération des mots sous formes de dictionnaire.
  2. [x] : Étiquettage des mots et remplissage du dictionnaire (numéro, pos, lemme).
  3. [x] : Utilisation de l'automate pour déterminer si la phrase passe le test ou non.
  4. [x] : Écriture des résultats.

CHAPTER 3

DÉMARCHES ET DIFFICULTÉ

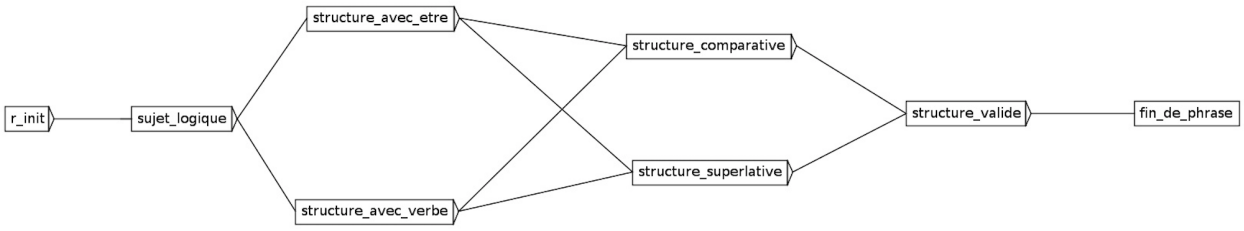
3.1 Approche

Dans un premier temps, nous avions une petite liste de phrases (5 par langue) que nous avons observé pour faire ressortir une structure générale. Ce temps fut notre brainstorming, et nous a aussi servi à réfléchir au processus global.

Ensuite, nous avons choisi un code à utiliser pour les POS.

Puis seulement, nous avons commencé à programmer. L'automate fut rapide à mettre en place, de même que les différentes fonctions.

L'automate est de la forme :



La plus grande partie du travail fut celle concernant les transitions. Elle a pris plus de 75% du temps.

Puis ces derniers jours furent consacrés au nettoyage du code, et à l'écriture de ce rapport.

3.2 Liste des problèmes

Voici la liste des problèmes rencontrés. Ils sont présenter par ordre chronologique du plus ancien au plus récent.

Num	Problème ou questionnement	Solution ou réponse
1	Sur quelle base doit-on fonder notre raisonnement ?	Nous avons pensé que les **POS** étaient plus indiqués. Dans de rares cas, nous avons aussi utilisé le lemme.
2	Comment formaliser une structure comparative ou superlative ?	Par l'**observation**, et en avançant pas à pas.
3	N'avons-nous pas besoin d'avoir des informations sur les mots et les phrases ?	Si, nous nous sommes appuyé sur **treetagger**.
4	Comme utiliser le même automate pour les deux langues ?	Heureusement, le français et l'anglais ont des structures assez similaires, dans ce cas précis au moins. Il a seulement fallu prendre en compte le fait que le tagger analyse de manière plus fine en anglais. D'où l'utilité du fichier **l3_pos.py**.

CHAPTER 4

RÉSULTATS

4.1 Fichiers finaux

Les résultats du programme sont inscrits dans deux fichiers par langue :

- phrases_*_results.txt,
- phrases_*_analyse_detaillee.txt[1].

Le premier se présente ainsi :

```
5 : Jean est le moins fort.
    ----> OUI : Reconnaissance réussie.
       Je pense que c'est une structure de comparaison.
```

Le seconde nous donne une analyse détaillée et se présente ainsi :

PAGE SUIVANTE —>

[1] * : en ou fr

```
    Phrase numéro 2 : The fish is smaller than The dog.

Key : 0 +++ forme : The +++ pos : DT +++ lemme : the
Key : 1 +++ forme : fish +++ pos : NN +++ lemme : fish
Key : 2 +++ forme : is +++ pos : VBZ +++ lemme : be
Key : 3 +++ forme : smaller +++ pos : JJR +++ lemme : small
Key : 4 +++ forme : than +++ pos : IN +++ lemme : than
Key : 5 +++ forme : The +++ pos : DT +++ lemme : the
Key : 6 +++ forme : dog +++ pos : NN +++ lemme : dog
Key : 7 +++ forme : . +++ pos : SENT +++ lemme : .

On est au mot 'The' de pos 'le+'
Le couple a chercher est : ('r_init', 'le+')
== Nous sommes dans 'sujet_logique' ==

On est au mot 'fish' de pos 'nom'
Le couple a chercher est : ('sujet_logique', 'nom')
== Nous sommes dans 'sujet_logique' ==

On est au mot 'is' de pos 'verbe_etre'
Le couple a chercher est : ('sujet_logique', 'verbe_etre')
== Nous sommes au coeur d'une structure avec être.
Il me faut avancer pour plus de détails. ==

On est au mot 'smaller' de pos 'adjectif_comparatif'
Le couple a chercher est : ('structure_avec_etre', 'adjectif_comparatif')
== On valide la 'structure_comparative' ==

On est au mot 'than' de pos 'que+'
Le couple a chercher est : ('structure_comparative_valide', 'que+')
== Nous sommes au coeur de 'structure_comparative' ==

On est au mot 'The' de pos 'le+'
Le couple a chercher est : ('structure_comparative', 'le+')
== Nous sommes au coeur de 'structure_comparative' ==

On est au mot 'dog' de pos 'nom'
Le couple a chercher est : ('structure_comparative', 'nom')
== On valide la 'structure_comparative' ==

On est au mot '.' de pos 'ponctuation'
Le couple a chercher est : ('structure_comparative_valide', 'ponctuation')
== Nous sommes à la fin de la phrase ==

!!! OUI : Reconnaissance réussie.
Je pense que c'est une structure de comparaison.
```

CHAPTER 5

LIMITES ET OUVERTURES

5.1 Limites

Si nous sommes un peu content du programme, nous sommes aussi conscient de ses limites.

Parmi les limites de notre automate :

- Il ne peut analyser que des phrases à deux membres maximum (A et B dans le cas d'une structure comparativw. A dans une phrase à structure superlative.).

- Il ne peut analyser les phrases interrogatives pour le moment, surtout en anglais (ex : Is it bigger ?).

- Il n'est pas exhaustif. À l'échelle d'une langue, 200 phrases ou cas ne représentent pas grand chose.

- Au sein même des motifs reconnus, il y a possibilité que certains puissent entrer en conflit avec d'autres. Un corpus plus grand et du temps auraient pu aider.

- Enfin, comme toujours, nous aurions aimé affiner le programme, proposer plus d'options.

5.2 Ouverture et réflexions diverses

Toutes les conclusions de notre devoir du Semestre 5 s'appliquent à nouveau ici.

De plus, programmer cet automate m'a permis de me rendre compte de ce que peut être un travail linguistique (le fait de répertorier des cas, de les analyser, de modifier ou ajouter des transitions au fur et à mesure).

Ce fut un grand exercice d'observation voire de méditation. On pense tous savoir ce qu'est une chose (ici une comparaison). La formaliser est une autre paire de manche. J'ai passé des heures à juste essayer de comprendre comment mon cerveau analyse une phrase et en déduit si elle est ou non comparative. Ce fut clairement le plus intéressant.

Enfin, si nous ne devions garder qu'une phrase de cette année, ce serait **«Testez votre programme sur un corpus.»**.