Léo Lebrillant
Hugo Laurent
Théo Driutti
Jean-Michel Dagba

Groupe E

Projet d'ingénierie linguistique

Qui a tué qui ? quand ? et où ?

Introduction

Pour ce projet nous avions pour objectif de programmer un enquêteur capable de d'identifier au moins 50 assassins dans les pages de Wikipédia ainsi que la victime, le lieu et la date du meurtre. Pour cela, nous avons utilisé différents outils afin de travailler en collaboration et de faciliter la programmation . Pour le travail de groupe, nos principaux outils étaient GitHub et Google Drive. Pour la programmation, nous avons utilisé les outils qui étaient proposé à savoir NLTK et stanfordNLP.

Afin de réaliser ce programme nous étions un groupe de 4 étudiants. Nous avons choisi de chercher des assassins commençant par le lettre E. Pour filtrer les pages wikipédia, nous avons choisi de garder les pages contenant le tag #death.

Implémentations réalisées

1. Tagging

Pour réaliser le tagging, nous avons utilisé Core NLP de Stanford sur le fichier texte du corpus. Pour cela, nous avons utilisé les arguments tokenize, ssplit, pos, lemma et ner. Voici la commande utilisée :

```
java -mx7g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,lemma,ner -file W:\Bibliotheques\Documents\Github\ProjetE-TAL\TAL\corpus.txt
```

2. Implantation de fonctions pratiques

```
def before(mot, corpus):
    a=corpus.split(mot)
    return(a[0])

def after(mot, corpus):
    a=corpus.split(mot)
    return(a[1])
```

3. Extraction de l'ID des phrases comportant une personne dont le nom commence par la lettre E

```
for i in range(len(corpus)):
    if corpus[i].find("<sentence id=")!=-1:
        s=before('"',after('"',corpus[i]))
    if corpus[i].find(">PERSON<")!=-1:
        if corpus[i-5].find("<word>E")!=-1:
            liste.append(s)
```

- ***s*** *est la variable dans laquelle l'ID de la phrase concourante (en cours d'exploration) est stockée.*
- ***liste*** *stock ainsi les ID désirées.*

4. Extraction des phrases à partir de leurs ID

```
i=0
for i in range(len(liste)):
    if corpuss.find('<sentence id="'+str(liste[i])+'"')!=-1:
        corpuss=after('<sentence id="'+str(liste[i])+'"',corpuss)
    liste2.append(before('</sentence>',corpuss))
```

- *La condition* ***si*** *permet d'éviter un out of range en cas d'absence de la chaîne recherchée*
- *Le* ***corpuss*** *(version chaîne de la liste complète du corpus initial) est élagué quand une partie à été traité*

5. Elaboration d'un champ lexical de verbes associé au meurtre

```
lemme=["assassinate","drown","execute","massacre","murder","poison","slaughter","slay",
       "annihilate","asphyxiate","crucify","electrocute","finish","garrote","guillotine"
       ,"immolate","liquidate","lynch","neutralize","smother","strangle","suffocate"]
```

6. Sélection des phrases comportant un lemme correspondant au champ précédemment défini

```
for i in range(len(liste2)):
    for j in range(len(lemme)):
        if liste2[i].find(lemme[j])!=-1:
            lpresent=1
    if lpresent==1:
        liste3.append(liste2[i])
    lpresent=0
```

7. Détermination du rôle entre meurtrier et victime selon la présence ou d'un "by" suivant le verbe

8. Elimination des phrases où la lettre E correspond en réalité à une victime

9. Recherche de tagging de lieu et de date dans les phrases restantes

10. Extraction des tueurs, victimes, lieux et dates pour chaque phrase retenue

Problèmes rencontrés et solutions

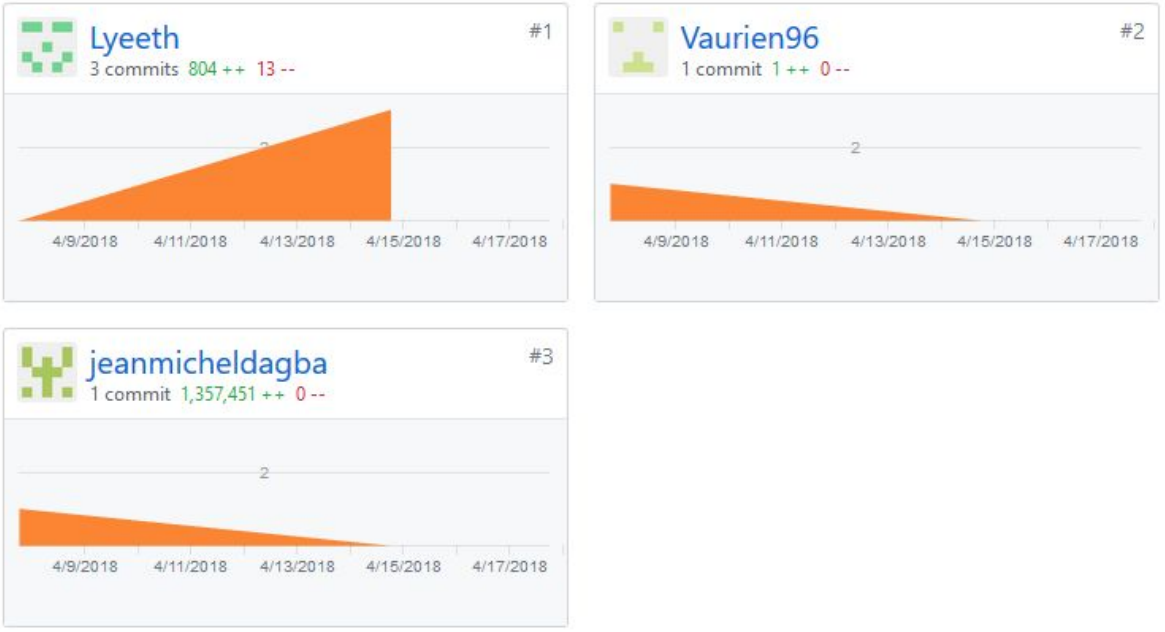
Dans cette partie nous allons aborder les problèmes rencontrés lors de la compréhension du sujet, de la recherche de corpus et le développement. Nous aborderons aussi les solutions que nous avons mises en place pour pallier à ces problèmes.

Lorsqu'il a été demandé de choisir une lettre pour le groupe, nous ne connaissions pas encore le sujet du projet. Nous avons alors choisi la lettre X. Toutefois nous nous sommes vite rendus compte lors de la découverte du sujet et du choix du corpus que cette lettre n'était pas adaptée. En effet, très peu de personnes habitant un pays anglophone ont un prénom commençant par la lettre X. Nous avons alors décidé de choisir la lettre suivant celle du dernier groupe dans l'alphabet, à savoir la lettre E.

Lorsque nous avons voulu tagger le corpus en utilisant coreNLP, nous n'avons pas réussi à utiliser la coréférence car le temps de traitement était beaucoup trop long. Hugo a fait tourner la commande toute la nuit mais malheureusement, une exception liée au manque de mémoire disponible a été levée. Nous avons alors décidé d'abandonner cet argument.

En ce qui concerne la programmation sur Python, nous n'avons pas réussi à utiliser le fichier .xml contenant le corpus tagué. Après avoir passé beaucoup de temps à essayer d'installer différents outils pour le .xml nous avons décidé de traiter un fichier .txt équivalent pour pouvoir avancer dans la réalisation du projet.

Gestion de projet



L'image ci-dessus montre l'activité du repository Github. Toutefois, il ne reflète pas bien l'implication des membres de l'équipe. En effet, le dépôt n'a été que tardivement car les traitements ont auparavant été réalisés sur une seule machine durant les TD. Les autres machines servaient par exemple à tester rapidement des expressions régulières sur l'outil de recherche de l'éditeur de texte Sublime Text, trouver de la documentation, affiner le corpus ou encore le tagger. Les différents fichiers ont alors tardivement été déposés sur Github, ce qui explique l'activité largement inégale montrée sur cette image.

Concernant la gestion du projet, nous avons principalement travaillé un groupe durant les heures de cours dédiées au projet. Nous n'avons pas désigné de

chef de projet ce qui nous a peut-être manqué pour nous répartir les tâches efficacement. La réalisation de ce projet est malheureusement incomplète car nous avons rencontré de nombreuses difficultés dans l'appréhension des outils de TAL.