Projet ingénierie linguistique

Groupe C - Louis Behr, Mathieu Collet,Pierre Daudier-De-Cassini,Léa Thamié

19 avril 2018

Table des matières

1 Introduction

Le but du projet est de développer un programme qui trouvera 50 assassins commençant par la lettre C ainsi que les modalités des meurtres qu'ils ont commis dans un corpus de texte prélevé de Wikipedia.

Le programme devra donc trouver :

— L'assassin
— La victime
— Le lieu du meurtre
— La date du meurtre

Nous avons réalisé ce projet en utilisant github, Stanford corenlp, NLTK et overleaf. Le programme a été réalisé en python grâce à PyCharm.

2 Technologies utilisées

2.1 Github

Github (lien vers le repository) nous a permis de travailler en collaboration sur le code sans avoir le besoin de nous rencontrer. Bien que nous travaillions régulièrement ensemble, il était nécessaire de pouvoir travailler indépendement. Cependant comme le projet a initialement consisté à nous familiariser avec la génération et l'exploration des fichiers xml, le travail individuel était le plus souvent des recherches ou de l'apprentissage et le travail en groupe se faisait le plus souvent sur un même ordinateur. C'est pourquoi, bien queGitHub soit un outil incontournable de la gestion de projet, il a été peu solicité pendant ce projet et son utilisation n'est pas représentative de l'investissement de chacun. Nou avons aussi utilisé Renater pour le transfert de fichier.

2.2 Stanford coreNLP

Stanford coreNLP est un programme qui nous a servi à effectuer le découpage de nos corpus. En effet, nous l'avons utilisé à partir de la console pour produire un corpus étiqueté. Pour cela, nous avons utilisé plusieurs annotateurs :

— tokenize qui permet de couper le corpus en tokens
— ssplit qui permet de couper le corpus en phrase
— pos qui permet d'étiqueter morpho-syntaxiquement le corpus (Stanford coreNLP utilise le tagset du Penn Treebank en langue anglaise)
— lemma qui permet d'associer chaque mot à son lemme
— ner qui permet de détecter les entités nommées et de les classer en différentes catégories (person, location, organization, date etc...)
— parse qui permet une analyse syntaxique et déterminer des dépendances entre les mots.

Avant d'effectuer cet étiquetage, nous avons utilisé python afin d'élaguer notre corpus.

2.3 NLTK

NLTK est un package que l'on importe dans Python qui permet de traiter des

ressources langagières. Nous l'avons utilisé afin d'établir une liste de synonyme pour effectuer une recherche approfondie dans le corpus.

2.4 Overleaf

Overleaf est un site internet qui permet une édition collaborative de document en LaTeX. Nous avons utilisé ce site pour produire ce rapport.

3 Déroulement du projet

Le projet s'est déroulé en plusieurs étapes.

3.1 Étape 1 : génération du corpus

La première étape de ce projet est la génération de corpus. En effet, pour effectuer des test, nous utiliserons un corpus prélevé de Wikipedia. Ce corpus se trouve sous la forme d'un .xml et nous l'avons donc transformé en .txt avec l'aide de cette page.

3.2 Étape 2 : segmentation du corpus

Une fois le corpus récupéré, nous avons procédé à une segmentation du corpus pour supprimer les phrases encombrantes, ou les lignes sans intérêt. Grâce à une analyse des phrases, nous avons remarqué que certains caractères étaient communs à toutes ces phrases et n'étaient pas contenus dans les autres. Nous avons donc créé un programme python qui permet de "corriger" le corpus en supprimant ces phrases. Le programme `segmentation.py` permet en effet la suppression des phrases encombrantes. Le programme ouvre le fichier .txt et repère les caractères choisis sur chaque ligne. Si la ligne ne contient pas ce caractère, il l'ajoute dans une variable. Un nouveau fichier est ensuite créé et on stockera la variable dans ce fichier. Le nouveau fichier est prêt à être étiqueter.

3.3 Étape 3 : étiquetage du corpus

La prochaine étape est l'étiquetage du corpus. Pour cela, nous avons utilisé Stanford coreNLP et nous avons entré cette ligne dans l'invite de commande :

$$-Xmx8g\ edu.stanford.nlp.pipeline.StanfordCoreNLP\ -$$
$$annotators\ tokenize, ssplit, pos, lemma, ner, parse - fileCorpus\_partie1.txt$$

Cette ligne de commande étiquette notre .txt et le transforme en .xml.

3.4 Étape 4 : reconnaissance des différents mots intéressants

Le texte étant alors étiqueté, nous avons commencé à procéder à la reconnaissance des entités nommées catégorisées en tant que personne, date et location grâce aux fonctions $getLocation(sentence)$, $getDate(sentence)$ et $getName(sentence)$ situées dans le programme `dateExtract.py`. Ces fonctions parcourent le fichier à la recherche des entités nommées correspondantes et les ajoute dans des listes. Ainsi,

les listes sont retournées.
Par la suite, nous avons créé plusieurs autres fonctions. Les fonctions $doesThisSentenceHasAName(s$ $doesThisSentenceHasAKill(sentence)$ permettent de déterminer si une phrase passée en paramètre contient un nom qui commence par un C ou/et le lemme kill ou un de ces synonymes. Pour établir les différents synonymes du lemme "kill", nous avons utilisé la fonction wordnet du package nltk. Wordnet est une base de données lexicale pour l'anglais et nous a donc servi à obtenir les différents synonymes possibles pour kill. Ainsi, après avoir détecté ces phrases, nous les stockons accompagnées des 2 phrases qui suivent. En effet, ces phrases peuvent nous être utiles pour déterminer un des paramètres que nous voulons récupérer. La fonction qui nous permet de repérer ces phrases et de les stocker est $getSentencesWithKillAndNameV3()$.

3.5 Étape 5 : extraction du sens

Dans cette dernière étape nous nous intérressons à un autre aspect du fichier xml, les tags dépendence. Dans cette version du projet cette partie n'est pas achevée. Pour continuer dans notre logique d'étude des phrases nous avons créé des fonctions qui nous permettent d'avoir le relation de dépence en fonction de l'id d'une phrase et d'un mot. Ainsi on veut pouvoir trouver dans une phrase simple et préalablement sélectionnée qui fait l'action, donc on veut étudier les relations de dépendance entre les mots. Elle va nous permettre de savoir dans nos extraits "qui tue qui". Pour l'instant nous avons restreint nos fonctions au parcours des basic-dependencies, pour dans un premier temps essayer de traiter les phrases simples où l'information est dans une seule phrase.

Pour l'instant nous n'extrayons pas l'infomration mais nous avons créé les premières fonctions qui le permetteront.

4 Problèmes rencontrés

Le premier problème que nous avons rencontré est lors de l'élaboration de corpus. Nous avions tout d'abord établi un corpus en français avant de nous rendre compte que les programmes de Natural Language Processing (Stanford coreNLP et NLTK) étaient bien plus compatibles avec l'anglais. Nous avons donc réalisé un autre corpus cette fois en anglais et cela fut une bonne décision car nous avons pu nous servir de la base de données de wordnet afin d'établir une liste de synonymes. Par la suite, nous avons rencontré un problème lors de l'étiquetage des corpus. En effet, l'étiquetage a pris du temps à tourner e t nous avons du le réaliser deux fois dû à notre premier problème. De plus, l'étiquetage avec l'annotateur dcoref n'a pas fonctionné.

Une fois nos corpus établis il nous a fallu un certain temps avant d'être à l'aise avec la navigation dans l'arbre etree, même si aujourd'hui encore il nous reste beaucoup d'aspects à découvrir nous avons tous découvert une nouvelle façon de réfléchir et d'explorer un texte.

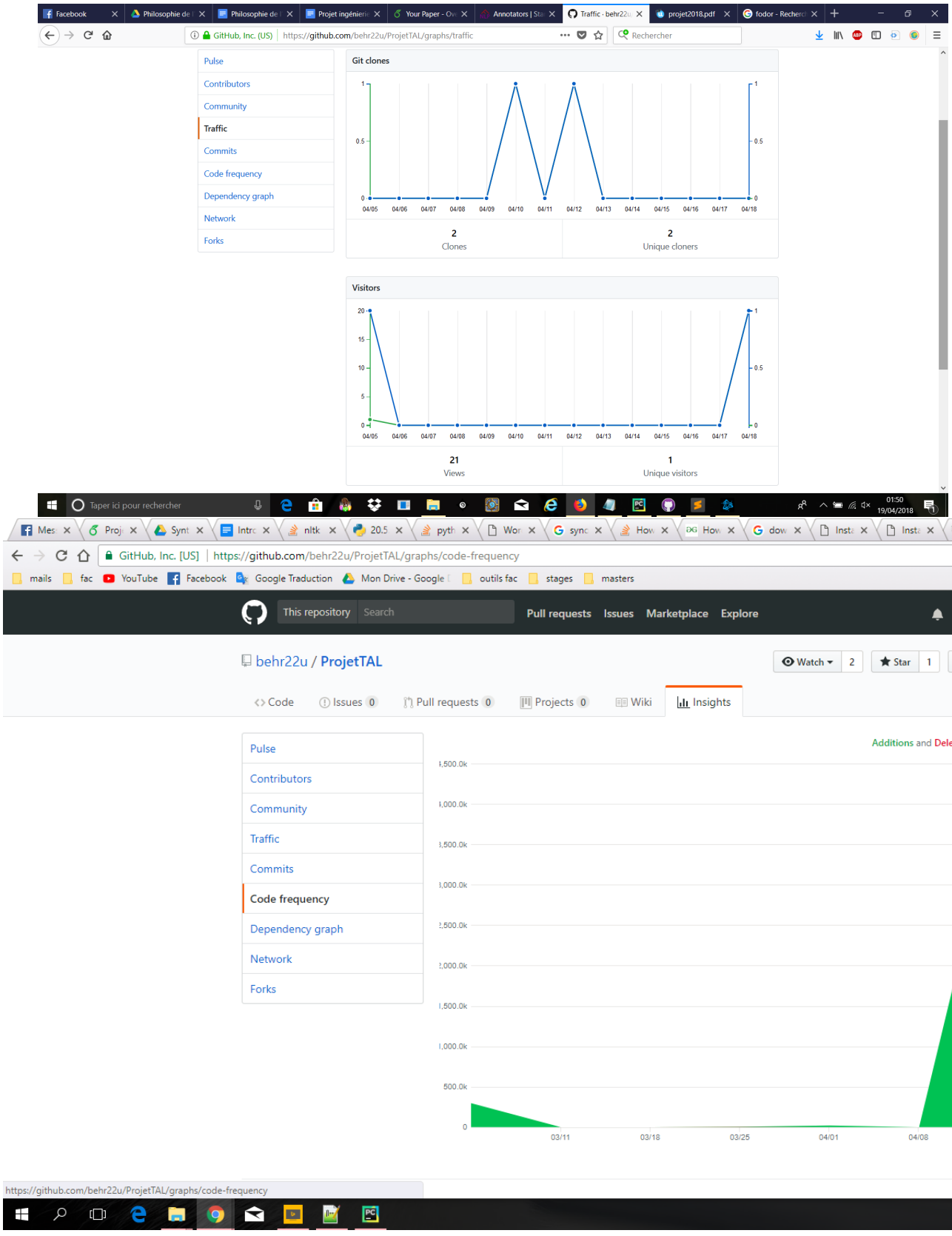
Aussi, nous n'avons pas pu atteindre tous nos objectifs par manque de temps et de compétences. La gestion de projet fut aussi un difficulté de ce projet, dans la mesure où nous ne maîtrisions ni les outils ni la méthodologie, la répartition des tâches était difficile, et le travail en groupe nous semblait souvent peu fructueux. Cependant chacun a finir par avancer dans la direction qui lui semblait la plus simple

en tenant le reste du groupe au courant de ses avancées.

5 Conclusion

Même si le programme que nous avons réalisé n'est pas fini il est fonctionnel et permet l'exploration du corpus et l'extraction de certains éléments en rapport avec le sujet. En effet, celui-ci nous a permis de se former dans l'utilisation d'outils de Natural Language Processing et de nous familiariser avec ceux-ci. Nous avons de même renforcé nos connaissances portant sur le Natural Language Processing acquises en CM. Cependant, le temps nous a manqué (notamment dû à une mauvaise gestion et à un emploi du temps chargé) pour mener ce projet à terme. Le projet nous a permis d'apprendre à utiliser des librairies que nous ne savions pas utiliser tels que Stanford coreNLP et NLTK. De plus nous avons pu mettre en pratique ce que nous avons appris lors de nos CM notamment en ce qui concerne la segmentation et l'étiquetage.

6 Annexes


Pulse
Contributors
Community
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Git clones
2
Clones
2
Unique cloners
Visitors
21
Views
1
Unique visitors
https://github.com/behr22u/ProjetTAL/graphs/traffic
https://github.com/behr22u/ProjetTAL/graphs/code-frequency
behr22u / ProjetTAL
Watch 2
Star 1
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Additions and Deletions per wee
03/11
03/18
03/25
04/01
04/08