ET5 - TAL
Anandou CANDASSAMY
Abdoulaye DIONGUE
Justine Thanh NGUYEN HOANG TUNG
S9 2019-2020

Rapport du projet de TAL

SOMMAIRE

1. Introduction

Ce document fera office de rapport pour le projet de Traitement Automatisé de Langage effectué par le groupe constitué de Anandou CANDASSAMY, Abdoulaye DIONGUE et Justine Thanh NGUYEN HOANG TUNG.

La rédaction de ce document a été faite par l'ensemble des membres du groupe. Il sera accompagné des fichiers sources du projet et des données utilisées pour les analyses.

Ce document détaillera l'objectif du projet, le travail effectué et enfin le résultat des analyses.

2. Objectifs

L'objectif du projet est de comparer les résultats des différentes analyses lexicales introduites lors des TPs de TAL, à l'aide de plateformes open source d'analyse linguistique (LIMA, Stanford et NLTK). Cependant, les différentes analyses lexicales utilisent des identificateurs différents pour leurs analyse. On cherchera donc à créer des scripts Python qui permettront de modifier les tags utilisés par l'analyseur syntaxique par les tags universels.

Une fois l'analyse avec les tags universels effectuée, on cherchera à comparer le résultat des différentes analyses en utilisant le script « evaluate.py ». Cette analyse sera précédé par une phase de traitement des fichiers pour faciliter les analyses.

3. Travail effectué

3.1 Analyse morpho-syntaxique

1. Travail

Pour l'analyse morpho-syntaxique, le tâches se déroulent l'une après l'autre. On commence avec le fichier pos_reference.txt.lima. Grâce au script LimaTransform.py, on obtient le fichier pos_reference.txt.lima2. La transformation consiste à modifier le formatage et certaines des étiquettes utilisées dans le fichier. En effet, certaines lignes du fichier pos_reference.txt.lima contiennent plusieurs mots. Pour permettre une analyse plus simple, on décompose ces lignes en plusieurs lignes en affectant les étiquettes adéquates à chaque ligne.

Le fichier pos_reference.txt.lima2 contient donc des lignes avec des étiquettes associées à des mots. Les étiquettes sont celles associées au format lima. Il est donc nécessaire de remplacer ces étiquettes par les étiquettes universelles. On utilisera le script LimaToUniv.py pour ainsi obtenir le fichier pos_reference.txt.univ.

Ce fichier contiendra les mots avec les tags universels. Pour lancer les différentes analyses morpho-syntaxiques, il est nécessaire d'en extraire le texte. On utilisera donc le script TaggedToTxt.py pour avoir le fichier pos_test.txt.

A partir de pos_test.txt on obtiendra le résultat des différents types d'analyses. Cependant le résultat des analyses n'est pas toujours dans le format qui permettra de comparer ces résultats avec le fichier pos_reference.txt.univ. En effet, pour permettre la comparaison, il doit y avoir le mot suivi de l'étiquette après une tabulation sur le fichier. On stockera donc les résultats initiaux des analyses dans les fichiers pos_test.txt.pos.lima.unrendered, pos_test.txt.pos.stanford.unrendered et pos_test.txt.pos.nltk.unrendered. On appliquera sur ces fichiers les scripts LimaConllToDC.py et StanfordToDC.py. Le fichier pos_test.txt.pos.nltk.unrendered ne nécessite aucune modification donc on utilisera le même fichier renommé pos_test.txt.pos.nltk.

Le format étant le même pour les deux fichiers, on lancera ensuite le script LimaToUniv.py pour remplacer les tags des différents fichiers par les étiquettes universelles. On pourra alors lancer le sript evaluate.py pour avoir les résultats pour chacune des analyse morpho-syntaxiques.

2. Résultats

Fichier	pos_test.txt.pos.lima.univ	pos_test.txt.pos.stanford.univ	pos_test.txt.pos.nltk.univ
Word precision	0.02526	0.02563	0.02669
Word recall	0.02526	0.02563	0.02669
Tag precision	0.02526	0.02563	0.02669
Tag recall	0.02526	0.02563	0.02669
Word F-measure	0.02526	0.02563	0.02669
Tag F-measure	0.02526	0.02563	0.02669

Comme on peut le voir dans les résultats, le fichier de référence et les fichiers obtenus par les analyses sont très différents. Plusieur facteurs expliquent ces différences. On peut néanmoins voir que la grande partie de ces différences est le résultat de l'étape de tokenization.

En effet, les analyses n'identifient pas les mêmes tokens. Cette variation des tokens n'est pas due aux mots composés car les mots composées ont été séparés par le script LimaTransform.py. Cependant, les abréviations créent un problème dans leur identification. Par exemple, les expressions telles que "We are" peuvent être abréviées en "We're". Cette abréviation pourra être tokenisé de deux façons: {We, ', re} que l'on retrouvera chez Lima ou {We, 're} que l'on retrouvera dans l'analyse de Stanford. A partir des ces tokenizations, les éléments identifiés n'auront donc pas les mêmes étiquettes.

Ce type d'abréviation est assez commune dans la langue anglaise. Mais on rencontre ce problème de tokens pour des expressions plus communes comme l'expression du possessif en anglais. Donc beaucoup d'éléments poseront ce problème. On comprendra donc pourquoi le nombre de lignes n'est pas le même dans les trois fichiers.

Pour résoudre ce problème, il peut être intéressant d'ajouter des règles de tokenisation qui permettront d'effectuer un meilleur découpage. En ajoutant ces règles, il sera possible d'identifier la présence d'expressions abrégées ou encore les expressions courantes dans une langue donnée.

Un autre facteur qui peut injecter des erreurs dans l'analyse du corpus sont les mots dont le rôle dépend de leur contexte d'utilisation. Les noms propres ou encore les adjectifs sont des mots qui se retrouvent souvent dans cette situation. En effet, des noms peuvent assurer la fonction d'adjectif. Le découpage en mot ne permet pas d'identifier l'étiquette qui y correspond le mieux.

Un dernier élément qui peut être un frein à une analyse adéquate est la présence de citations dans le texte. Les citations sont encadrées par des guillemets. Les guillemets ne sont pas toujours identifiés comme des

signes de ponctuation. En effet, les guillemets sont pour certaines analyses reconnus comme faisant partie d'un mot et pour d'autre non.

Tous ces éléments réduisent la précision des analyses. Avec un texte d'une certaine longueur, les erreurs s'empilent et finissent par faire diverger les résultats de la référence dont l'analyse est partie.

On peut voir néanmoins que les trois analyses ont un léger décalage dans leur précision. Ce décalage montre que l'analyse nltk est celle qui donne la meilleure précision. Cependant l'analyse lima donne la précision la plus basse. L'efficacité de nltk réside dans sa capacité à effectuer un découpage mot par mot. Ce découpage est plus approprié pour une analyse syntaxique qui ne cherche pas à mettre en évidence les groupes linguistiques.

3.2 Reconnaissance d'entités nommées

3.2.1 Travail

Dans cette partie, le travail consiste à produire un fichier conll pour chaque recognizer à partir d'un fichier de référence afin d'effectuer leur évaluation. Pour ce faire, on a suivi une suite d'étape.

Dans un premier temps, on a utilisé le corpus "**ne_reference.txt.conll**" pour extraire les phrases ayant servi pour produire ce corpus annoté et sauvegarder le résultat dans le fichier "**ne_test.txt**". Cette tâche a été faite grâce au script python **TaggedToTxt.py**.

Ensuite, on a lancé les trois recognizers sur le fichier **ne_test.txt** pour obtenir des fichiers dont les lignes sont composées de deux colonnes, l'entité nommée et son type. Pour le stanford, on a utilisé la commande **java -mx600m -cp stanford-ner.jar:lib/* edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier classifiers/english.all.3class.distsim.crf.ser.gz -textFile nr_test.txt > ne_test.txt.ne.stanford.unrendered** qui écrit dans le fichier **ne_test.txt.nestanford.unrendered** les entités nommés et son type séparés par un "\". Après cela, on transforme ce fichier au format deux colonnes dans le fichier ne_test.txt.ne.stanford par le script StranfordNeToDC.py. Pour le lima, on utilise la commande de la partie une, ensuite, on lance le script LimaNeToDC.py pour obtenir le fichier **ne_test.txt.ne.lima** qui est au format deux colonnes. Et, pour le nltk, le **script Nltk_ne.py** permet les entités nommées et leurs types sous forme d'arbre qu'on transforme en ligne de deux colonnes.

Enfin, on convertit les résultats des trois NE recognizers en utilisant les étiquettes CoNLL-2003. Pour stanford, lima et nltk, les conversions sont faites respectivement pas les scripts **StanfordNeToConll**, **LimaNeToConll** et **NltkNeToConll**. Cette conversion consiste à marquer le début et la suite des types des entités nommés qui ont le même type et se suivent. Autrement dit, si on a, par exemple Mary PERSON Barr a PERSON, la conversion donne Mary B-PERS Barr a I-PERS. Les résultats de ces conversions sont stockés dans des fichiers conll.

3.2.2 Résultats

Fichier	ne_test.txt.ne.stanford.conll	ne_test.txt.ne.lima.conll	ne_test.txt.ne.nltk.conll
Word precision	0.009237	0.012227	0.019864
Word recall	0.009237	0.012415	0.019864
Tag precision	0.009237	0.0122227	0.019864
Tag recall	0.009237	0.012415	0.019864
Word F-measure	0.009237	0.012320	0.019864
Tag F-measure	0.009237	0.012320	0.019864

Résultat de l'évaluation des différents recognizer avec le fichier de référence

On voit dans le tableau ci-dessus que les résultats trouvés lors de l'évaluation des différents recognizer sont différents et faibles. Ces résultats peuvent être expliqués par plusieurs raisons.

Etant donné que chaque recognizer a sa propre manière de générer les types des entités nommées à partir d'un texte, le nombre de lignes obtenus dans les fichiers conll peut être différent avec celui du fichier de référence. Ce qui fait un facteur d'explication des résultats de l'évaluation.

Après lancement du recognizer stanford, on a remarqué que les types des entités nommées reconnus sont seulement ORGANIZATION, LOCATION et PERSON, et toute autre entité nommée est considérée comme OTHER. De plus, une même entité nommée peut avoir un type différent sur le fichier de référence et sur le stanford, par exemple The United State est représenté par **The B-ORG United I-ORG B-LOC States I-ORG I-LOC** dans le fichier de référence alors que dans le fichier généré par stanford il est représenté par **The O United B-LOC States I-LOC.**

La même explication est valable pour lima. De plus, l'analyseur de lima possède plus de types d'entité nommée que le fichier de référence. Ainsi, on remplace tout type différent de OTHER, ORGANIZATION, LOCATION et PERSON sont remplacés par MISC, qui n'est pas présent dans le fichier de référence. En plus, Lima peut regrouper entité nommée composée dans un seul type, par exemple **December 10, 2013 B-PER** est découpé dans le fichier de référence. Cela apporte plus de différence aux fichiers à évaluer.

L'ensemble des explications ci-dessus sont à l'origine des résultats de l'évaluation avec génération des types des entités nommées par nltk.

3.2.3 Synthèse des trois recognizer sur la reconnaissance des entités nommées

En résumé l'évaluation de la reconnaissance des entités nommées par les trois recognizer donne à peu près le même résultat mais les implémentations diffèrent. En effet, le stanford est plus facile à utiliser car il ne demande pas beaucoup de traitements mais il donne moins de résultats. Par contre, lima donne des résultas complets pour l'analyse morphologique et la reconnaissance des entités nommées mais le traitement de ces résultats pouvant être lourd. Enfin, NLTK possède est simple à utiliser car la librairie offre plusieurs fonctions pour faire ce type d'exercice mais son défaut réside sur la durée de traitement car le téléchargement des packages et les fonctions mettent beaucoup de temps.

4. Contribution des membres

Anandou CANDASSAMY a travaillé sur la partie I et ses scripts python. Justine Thanh NGUYEN HOANG TUNG a travaillé sur la conversion du Lima en PTB (pour pouvoir les convertir en étiquettes universelles par la suite).

Abdoulaye DIONGUE a travaillé sur la partie II, pour convertir les fichiers Lima, Stanford et Nltk en conll. Juste l'a également aidé pour décider de la façon de regrouper certains tags sous l'étiquette MISC de conll.

Abdoulaye et Anandou ont écrit la rédaction du Readme.txt. Anandou et Justine ont écrit le rapport.