THIEFFRY Rémy
VU Tri-Man William

Rapport de projet – TAL

Sujet 1 : Evaluation de deux plateformes open source d'analyse linguistique

Remarque : L'ensemble des informations liées aux scripts des TP et du projet est regroupé dans les fichiers *README.md*.

1. Introduction

Le Traitement Automatique de la Langue est un domaine relativement vaste regroupant un ensemble d'applications comme l'extraction d'informations, la reconnaissance vocale ou encore la traduction automatique. Plusieurs outils linguistiques existent afin de faire de l'analyse de texte, par exemple l'analyse morphologique, l'analyse morpho-syntaxique, l'analyse syntaxique ou encore l'extraction d'entités nommées.

Dans le cadre de ce projet, nous nous focalisons sur l'analyse morpho-syntaxique ainsi que l'extraction d'entité nommées. L'**analyse morpho-syntaxique** consiste à évaluer la forme (morphologie) ainsi que la fonction (syntaxe) des *tokens* d'un corpus. Les *tokens* sont obtenus après avoir découpé les chaînes de caractères du texte en mots (la *Tokenisation*). L'**extraction d'entités nommées** consiste à catégoriser des groupes de mots (noms d'organisations, noms de lieux, …).

L'objectif principal de ce projet est d'étudier et comparer les performances des deux plateformes d'analyse linguistique suivantes : **Stanford Core NLP** et **NLTK**.

2. Présentation des plateformes d'analyse linguistiques

2.1. Stanford Core NLP

Stanford CoreNLP est un pipeline d'annotations qui fournit la plupart des étapes communes du traitement du langage naturel, de la tokenisation à la résolution de coréférence. Cette plateforme d'analyse est une boîte à outils linguistiques utilisant l'apprentissage statistique à partir de corpus annotés. Les approches utilisées sont à base de règles, statistiques et CRF.

Le système est fourni avec des modèles pour l'anglais. Certains des modèles sous-jacents aux annotateurs sont formés à partir de corpus annotés à l'aide d'un apprentissage automatique supervisé, tandis que d'autres sont des composants basés sur des règles, qui nécessitent néanmoins souvent certaines ressources linguistiques qui leur sont propres.

Le POS tagger est basé sur le Multilayer Perceptron (MLP) étant un Perceptron. Il utilise alors les réseaux de neurones et est donc à base de corpus.

Le NE recognizer est basé sur le Conditional random field (CRF) étant un Classifieur. Il est donc également à base de corpus.

Voici les différentes fonctionnalités implémentées :

- Tokenisation du texte en une séquence de tokens. Le composant anglais fournit un tokenizer de style PTB. Le tokenizer enregistre les décalages de caractère de chaque token dans le texte d'entrée.
- Suppression de tout ou de la plupart des balises XML du document.
- Divise une séquence de tokens en phrase.
- Détermine le vrai cas probable des tokens dans le texte où l'information a été perdue, par exemple, pour tout texte en majuscules. Ceci est mis en œuvre avec un modèle discriminatif utilisant une séquence CRF étiqueteur.
- Labelle les tokens avec leur étiquette de catégorie grammaticale (POS tag), en utilisant un POS tagger à entropie maximale.
- Génère les lemmes pour tous les tokens dans l'annotation.
- Ajoute des informations de genre probable aux noms.
- Reconnaît les entités nommées (PERSON, LOCATION, ORGANIZATION, MISC) et numériques (MONEY, NUMBER, DATE, TIME, DURATION, SET). Avec les annotateurs par défaut, les entités nommées sont reconnues en utilisant une combinaison de marqueurs de séquence CRF formés sur différents corpus, tandis que les entités numériques sont reconnues à l'aide deux systèmes basés sur des règles, un pour l'argent et numéros et un système de pointe séparé pour le traitement des expressions temporelles.
- Implémente un NER simple et basé sur des règles, sur des constructions de séquences de tokens s'appuyant sur des expressions régulières Java.
- Fournit une analyse syntaxique complète, incluant à la fois la représentation des constituants et des dépendances, basée sur un analyseur probabiliste.
- Analyse des sentiments avec un modèle de composition sur des arbres utilisant l'apprentissage en profondeur. Les nœuds d'un arbre binarisé de chaque phrase, y compris le nœud racine de chaque phrase, reçoivent un score de sentiment.
- Implémentation de la détection d'une mention et les résolutions de coréférence pronominale et nominale. L'ensemble du graphe de coréférence d'un texte (avec les têtes de mentions comme nœuds) sont fournis.

2.2. NTLK

NLTK (Natural Language Toolkit) est une boite à outils linguistiques utilisant des approches hybrides combinant l'apprentissage automatique et des ressources linguistiques. Cet outil est codé en Python et sous licence open source GPL. NLTK est composé de plusieurs modules. Une partie de ces modules permettent de définir les types de données de base utilisés dans la boite à outils, comme *probability* et *cfg*. Tandis que d'autres modules correspondent à des modules de tâches où chacun a une tâche individuelle dans le domaine du traitement du langage naturel. Par exemple, le module *nltk.tokenizer* est dédié à la tokenisation d'un corpus, c'est-à-dire découper le texte en un ensemble de *tokens* ("jetons"). Tandis que le module *nltk.pos_tag* permet de réaliser l'analyse morpho-syntaxique de *tokens*. Il existe également le module *corpus* qui regroupe un ensemble de textes de référence. Enfin, il existe les modules de traitement qui sont implémentés sous la forme d'une classe. Chaque module de traitement définit une interface liée à une tâche. Par exemple, l'interface *Tokenizer* définit la méthode *tokenize*.

Dans le cadre de ce projet, nous utilisons plusieurs modules dont les suivants :

- *nltk.corpus* pour importer un texte de référence
- *nltk.word_tokenize* pour réaliser la tokenisation d'un texte en un ensemble de jetons
- *nltk.pos_tag* pour désambiguïser morpho-syntaxiquement les *tokens*
- *nltk.chunk.ne_chunk* pour identifier les entités nommées

NLTK dispose de plusieurs méthodes d'étiquetage comme les **approches à base de règles** et les **approches statistiques**. Les **approches à base de règles** sont composées de règles souvent établies "à la main" et basées sur les connaissances grammaticales d'une langue. Pour étiqueter un *token*, cette approche se base sur les *tokens* qui le précédent et le suivent pour deviner la catégorie syntaxique du *token* étudié.
D'un autre côté, les **méthodes statistiques** dépendent de formules probabilistes et statistiques basées par exemple sur la fréquence d'apparition d'un mot dans un texte afin de lui attribuer une étiquette. On recense deux techniques d'étiquetage à partir d'approche statistique : l'étiquetage à partir d'unigrammes et l'étiquetage à partir de bigrammes. L'étiquetage à partir d'unigrammes associe à chaque token l'étiquette la plus fréquente pour ce *token*. Tandis que l'étiquetage à partir de bigrammes associe à chaque *token*, l'étiquette la plus probable étant donnée l'étiquette la plus fréquente pour ce *token* et l'étiquette du *token* précédent. L'avantage des **méthodes statistiques** est qu'elles sont rapides à implémenter mais elles nécessitent de grands corpus pour être efficace.

3. Métriques d'évaluation

Afin de mesurer les performances des deux plateformes d'analyse linguistiques étudiées, nous comparons les étiquettes prédites par rapport aux étiquettes de référence pour chacun des *tokens*.

On distingue trois cas possibles :
TP (True Positive) : Nombre de *tokens* étiquetés correctement par le POS Tagger (les étiquettes de ces *tokens* sont les mêmes que les étiquettes de la référence)
FN (False Negative) : Nombre de *tokens* qui n'ont pas été étiquetés par le POS Tagger (ces *tokens* n'ont pas été étiquetés par le POS Tagger mais pourtant ils existent dans la référence)
FP (False Positive) : Nombre de *tokens* étiquetés non correctement par le POS Tagger (les étiquettes de ces *tokens* sont différentes des étiquettes de la référence)

Nous utilisons trois métriques d'évaluation : la précision, le rappel ainsi que la F-mesure.
Précision : Le ratio entre tous les *tokens* correctement étiquetées et l'ensembles des *tokens* étiquetés :

$$precision = \frac{TP}{TP + FP}$$

Rappel : Le ratio entre tous les *tokens* correctement étiquetés et l'ensemble des *tokens* étiquetés par les experts :

$$rappel = \frac{TP}{TP + FN}$$

F-mesure : La moyenne harmonique correspondant à la combinaison de la précision et du rappel

$$f = 2 \cdot \frac{precision \cdot rappel}{precision + rappel}$$

4. Evaluation de l'analyse morpho-syntaxique

4.1. Description du corpus d'évaluation

Le corpus d'évaluation *pos_reference.txt.lima* comporte 10 554 lignes où chaque ligne représente soit un *token*, soit un retour à la ligne. En retirant les retours à la ligne, on a un total de 10 074 *tokens*, ce qui correspond à 481 lignes (*pos_test.txt*).

4.2. Comparaison entre Stanford Core NLP et NLTK

Les captures d'écran correspondant aux résultats de l'analyse morpho-syntaxique correspondent aux Annexes *1 et 2*.

Tableau des résultats

	Stanford	**Nltk**	**Différence absolue**
Précision	0.9266109785202864	0.9157978353688809	0.01081314315
Rappel	0.9077447637603507	0.9081331232768807	0.00038835951
F-mesure	0.9170808523202598	0.9119493745983092	0.00513147772

De manière générale, les deux plateformes d'analyse linguistique ont de bonnes performances. Chacune des métriques d'évaluation choisies est supérieure à 90% et les deux plateformes ont des performances similaires. En effet, pour chaque métrique d'évaluation, on observe une différence absolue inférieure ou égale à 1% entre les deux plateformes.

Cependant, nous pouvons noter que le rappel est légèrement moins bon pour Stanford. Cela peut s'expliquer par le fait que nous obtenons 7 anomalies avec Stanford. En effet, on ne retrouve pas exactement les mêmes *tokens* dans *pos_test.txt.pos.stanford.univ* et *pos_reference.txt.univ*. Par exemple, Stanford rajoute parfois un "." en trop en fin de phrase.

Au niveau de la précision et de la F-mesure, Stanford a des performances légérement meilleures que NLTK. Cependant, il faut gérer les différentes erreurs de Stanford (ajout inutile de ".", non traitement de certains tokens, modification des tokens pour les accolades et les parenthèses, ...). Tandis que NLTK semble légèrement moins bon mais nous n'avons eu aucune erreur à gérer pour cette plateforme.

5. Evaluation de la reconnaissance d'entités nommées

5.1. Description du corpus d'évaluation

Le corpus d'évaluation *ne_reference.txt.conll.txt* comporte 10 497 lignes où chaque ligne représente soit un *token* associé à une étiquette CoNLL-2003, soit un retour à la ligne. En retirant les retours à la ligne, on a un total de 10 068 *tokens*, ce qui correspond à 430 lignes représentant les phrases du corpus(ne_*test.txt*).

5.2. Comparaison entre Stanford Core NLP et NLTK

Les captures d'écran correspondant aux résultats de l'analyse morpho-syntaxique correspondent aux Annexes *3 et 4*.

Tableau des résultats

	Stanford	**Nltk**	**Différence absolue**
Précision	0.9078267779102106	0.9017145135566188	0.00611226435
Rappel	0.9992347217666995	0.9879860200961118	0.01124870167
F-mesure	0.9513400988810824	0.9428809672712112	0.0084591316

De manière générale, les deux plateformes d'analyse linguistique ont de bonnes performances. Chacune des métriques d'évaluation choisies est supérieure à 90% et les deux plateformes ont des performances similaires. En effet, pour chaque métrique d'évaluation, on observe une différence absolue inférieure ou égale à 1% entre les deux plateformes.

Cependant, nous pouvons noter que toutes les métriques d'évaluation sont légèrement moins bonnes pour Nltk. Il faut néanmoins gérer les différentes erreurs de tokenisation de Stanford (ajout inutile de ".", non traitement de certains tokens, modification des tokens pour les accolades et les parenthèses, ajout d'un "." en trop en fin de phrase, ...). Tandis que NLTK semble légèrement moins bon mais nous n'avons eu aucune erreur à gérer pour cette plateforme.

6. Points forts, limitations et difficultés rencontrées

Le point fort de ces deux plateformes est qu'elles offrent de très bonnes performances. De plus, notre code gère directement les anomalies de Stanford sans avoir besoin de modifier les fichiers des corpus étudiés.

Au niveau des limites de l'étude, nous nous apercevons que la conversion des étiquettes en étiquettes universelles implique une perte de l'information. En effet, LIMA est composé d'approximativement 20 étiquettes et Pen TreeBank en regroupe approximativement 40 alors que nous n'avons au final que 14 étiquettes universelles.

En ce qui concerne les difficultés rencontrées, nous avons eu plusieurs anomalies à traiter, notamment avec Stanford. Par exemple, à la fois pour l'analyse morpho-syntaxique et la reconnaissance d'entités nommées, Stanford Core NLP transforme les parenthèses ainsi que les accolades :

Token en entrée	Token en sortie de Stanford
(	-LRB-
)	-RRB-
{	-LCB-
}	-RCB-
]	-RSB-

Nous avons donc adapté les scripts *q1_3_TP1.py* et *qII_2_stanford.py* afin que ces *tokens* soient convertis en parenthèse, accolade ou crochet.

Nous avons également noté que Stanford rajoute parfois un "." en trop en fin de phrases et ignore certains *tokens*. En effet, nous pouvons voir dans les Annexes *5 et 6* que Stanford rajoute un "." en trop à la fin du mot "Corp". Au total, nous obtenons approximativement 7 / 11 000 *tokens* avec une anomalie pour l'analyse morpho-syntaxique avec Stanford.

De plus, nous avons mis à jour les tables de correspondance pour pouvoir réaliser l'analyse morpho-syntaxique. En effet :

- Pour la table de conversion des étiquette LIMA vers Penn TreeBank (*POSTags_LIMA_PTB_Linux.txt*), nous avons supprimé la ligne : "NONE NONE" car l'étiquette "NONE" n'existe pas dans l'autre table de conversion.
- Pour la table de conversion des étiquettes du Penn TreeBank vers les étiquettes universelles (*POSTags_PTB_Universal_Linux.txt*), nous avons rajouté les lignes suivantes pour pouvoir convertir les étiquettes du fichier de Stanford :

Etiquette Penn TreeBank	Etiquette Universelle
``	.
$	X
''	.

7. Organisation

7.1. TPs

Pour le TP1, Rémy et William ont fait l'évaluation de l'analyse morpho-syntaxique. Rémy a fait l'analyse syntaxique et William a fait l'extraction d'entités nommées.

Pour le TP2, William a installé les outils de l'université de Stanford et a réalisé l'analyse de l'outil de reconnaissance d'entités nommées. Tandis que Rémy a fait l'évaluation de l'outil de désambiguïsation morpho-syntaxique.

7.2. GitHub du projet

Rémy a créé le répertoire GitHub. Cependant, William et Rémy ont participé à la gestion de la hiérarchie des dossiers dans le répertoire.

7.3. Code du projet

Pour les deux TPs, William avait étudié l'extraction d'entités nommées et Rémy avait étudié l'analyse morpho-syntaxique. De ce fait, nous avons décidé de se séparer les parties du projet de la même façon. De plus, Rémy a réalisé le script *evaluate.py*.

7.4. Rédaction du rapport

Concernant la présentation des plateformes d'analyse linguistique, William a décrit le fonctionnement de la plateforme Stanford Core NLP. Tandis que Rémy a décrit le fonctionnement de NLTK. Rémy a rédigé l'évaluation de l'analyse morpho-syntaxique et William a rédigé celle de la reconnaissance des entités nommées. Pour les autres parties, nous nous sommes divisés le travail équitablement.

8. Conclusion

Les deux plateformes d'analyse linguistique ont de bonnes performances puisque chacune des métriques d'évaluation choisies est supérieure à 90%. De plus, les deux plateformes ont des performances similaires.

Cependant, nous pouvons globalement remarquer que l'approche par Stanford est légèrement meilleure que l'approche par Nltk. Néanmoins, l'étude avec l'outil Stanford nécessite de gérer les différentes erreurs qu'il peut générer lors de la tokenisation, pour pouvoir évaluer correctement les métriques avec le fichier de référence correspondant. Tandis que Nltk semble légèrement moins bon mais nous n'avons eu aucune erreur à gérer pour cette plateforme.

Ainsi, si l'on souhaite obtenir des résultats sans être contraints à devoir gérer quelques exceptions pour l'évaluation, il vaut mieux privilégier l'approche par Nltk. Sinon, dans le cas où nous préférons prendre le temps de gérer les exceptions afin d'obtenir un résultat légèrement meilleur, il vaudrait mieux privilégier l'approche par Stanford.

Pour finir, il ne faut pas oublier que les résultats obtenus avec les fichiers de référence ne signifient pas forcément que l'une des approches est plus performante que l'autre, puisque les fichiers de

référence peuvent comporter de nombreuses erreurs d'étiquettes que l'on peut repérer à l'œil nu par un humain. C'est notamment le cas pour le fichier de référence sur les entités nommées. Le fait d'observer quelle méthode se rapproche le plus d'un fichier de référence pouvant être "obsolète" n'est donc pas forcément représentatif de son efficacité, au contraire !

9. Annexes

```
(base) C:\Users\33750\Desktop\TAL\REPO_GITHUB\ET5_TAL\Projet>python evaluate.py data/pos_test.txt.pos.stanford.univ data/pos_reference.txt.univ
Nom de notre fichier en entree : data/pos_test.txt.pos.stanford.univ
Nom du fichier de reference en entree : data/pos_reference.txt.univ
Erreur au niveau de la ligne n° 1392  du fichier de reference et n° 1513  du fichier de predicition
Erreur au niveau de la ligne n° 2536  du fichier de reference et n° 2777  du fichier de predicition
Erreur au niveau de la ligne n° 3140  du fichier de reference et n° 3470  du fichier de predicition
Erreur au niveau de la ligne n° 4101  du fichier de reference et n° 4551  du fichier de predicition
Erreur au niveau de la ligne n° 6573  du fichier de reference et n° 7231  du fichier de predicition
Erreur au niveau de la ligne n° 7154  du fichier de reference et n° 7869  du fichier de predicition
Erreur au niveau de la ligne n° 8869  du fichier de reference et n° 9718  du fichier de predicition
7  erreur(s) referencee(s)

Metriques d evaluation :
Precision = 0.9266109785202864 = ( 9318 / (9318 + 738) )
Rappel = 0.9077447637603507 = ( 9318 / (9318 + 947) )
F-mesure = 0.9170808523202598 = 2 * 0.9266109785202864 * 0.9077447637603507 / ( 0.9266109785202864 + 0.9077447637603507)
```

Annexe 1 : Evaluation de l'analyse morpho-syntaxique de **Stanford Core NLP** sur le corpus d'évaluation

```
(base) C:\Users\33750\Desktop\TAL\REPO_GITHUB\ET5_TAL\Projet>python evaluate.py data/pos_test.txt.pos.nltk.univ data/pos_reference.txt.univ
Nom de notre fichier en entree : data/pos_test.txt.pos.nltk.univ
Nom du fichier de reference en entree : data/pos_reference.txt.univ
0  erreur(s) referencee(s)

Metriques d evaluation :
Precision = 0.9157978353688809 = ( 9223 / (9223 + 848) )
Rappel = 0.9081331232768807 = ( 9223 / (9223 + 933) )
F-mesure = 0.9119493745983092 = 2 * 0.9157978353688809 * 0.9081331232768807 / ( 0.9157978353688809 + 0.9081331232768807)
```

Annexe 2 : Evaluation de l'analyse morpho-syntaxique de **NLTK** sur le corpus d'évaluation

```
(base) C:\Users\user\Desktop\TAL\RepoGit\ET5_TAL\Projet>python evaluate.py data\\ne_test.txt.ne.stanford.conll.txt data\\ne_reference.txt.conll.txt
Nom de notre fichier en entree : data\\ne_test.txt.ne.stanford.conll.txt
Nom du fichier de reference en entree : data\\ne_reference.txt.conll.txt
0  erreur(s) referencee(s)

Metriques d evaluation :
Precision = 0.9078267779102106 = ( 9140 / (9140 + 928) )
Rappel = 0.9992347217666995 = ( 9140 / (9140 + 7) )
F-mesure = 0.9513400988810824 = 2 * 0.9078267779102106 * 0.9992347217666995 / ( 0.9078267779102106 + 0.9992347217666995)
```

Annexe 3 : Evaluation de l'analyse des entités nommées de **Stanford Core NLP** sur le corpus d'évaluation

```
(base) C:\Users\user\Desktop\TAL\RepoGit\ET5_TAL\Projet>python evaluate.py data\\ne_test.txt.ne.nltk.conll.txt data\\ne_reference.txt.conll.txt
Nom de notre fichier en entree : data\\ne_test.txt.ne.nltk.conll.txt
Nom du fichier de reference en entree : data\\ne_reference.txt.conll.txt
0  erreur(s) referencee(s)

Metriques d evaluation :
Precision = 0.9017145135566188 = ( 9046 / (9046 + 986) )
Rappel = 0.9879860200961118 = ( 9046 / (9046 + 110) )
F-mesure = 0.9428809672712112 = 2 * 0.9017145135566188 * 0.9879860200961118 / ( 0.9017145135566188 + 0.9879860200961118)
```

Annexe 4 : Evaluation de l'analyse des entités nommées de **NLTK** sur le corpus d'évaluation

```
Mazda   NNP
Motor   NNP
Corp.   NNP
.   .
In  IN
the DT
new JJ
position    NN
```

Annexe 5 : lignes 1 510 à 1 517 de *pos_test.txt.pos.stanford*

```
Mazda   NNP
Motor   NNP
Corp    NNP
.   .
In  IN
the DT
new JJ
position    NN
```

Annexe 6 : lignes 1 514 à 1 521 de *pos_test.txt.pos.nltk*