DELAMARCHE RUBEN
MARTIN Hugues
03/03/2024
ET5 – Polytech Paris-Saclay

Rapport projet Traitement Automatique des Langues (TAL) : Sujet 1

1. Introduction

Le traitement automatique du langage (TAL) est un domaine de recherche et d'application en plein essor, visant à permettre aux ordinateurs de comprendre et de générer le langage humain de manière efficace. Parmi les tâches cruciales du TAL figurent la désambiguïsation morpho-syntaxique (POS tagging), qui consiste à attribuer à chaque mot d'un texte une catégorie grammaticale, et la reconnaissance d'entités nommées (NER), qui vise à identifier et classifier les entités telles que les noms de personnes, d'organisations ou de lieux.

La performance de ces tâches dépend largement des algorithmes et des modèles utilisés, ainsi que des plateformes d'analyse linguistique sur lesquelles ils sont implémentés. Dans cette étude, nous nous intéressons à la comparaison des performances de deux différentes plateformes d'analyse linguistique, NLTK et Stanford Core NLP, pour ces deux tâches cruciales du TAL : le POS tagging et la NER.

2. Présentation des plateformes d'analyse linguistique

NLTK (Natural Language ToolKit) :

NLTK est une bibliothèque Python populaire pour le traitement du langage naturel. Elle propose une gamme d'outils et de ressources pour effectuer diverses tâches de TAL, en mettant l'accent sur l'éducation et la recherche dans le domaine.

NLTK intègre également des modules pour l'entraînement et l'utilisation de modèles statistiques pour des tâches telles que la classification de texte, le POS tagging, et la NER. NLTK offre aussi des fonctionnalités pour la création et l'application de règles linguistiques personnalisées. Les utilisateurs peuvent définir des règles pour effectuer des opérations telles que le POS tagging, la lemmatisation, et d'autres tâches d'analyse linguistique. Cette approche permet une flexibilité considérable dans la conception de pipelines de traitement du langage adaptés à des besoins spécifiques. **Enfin,** NLTK offre une gamme d'algorithmes d'apprentissage automatique pour des tâches de classification, de clustering, et d'autres types d'analyses de texte. Ces algorithmes peuvent être utilisés pour des tâches avancées telles que la détection de sujets, l'analyse de sentiments, et d'autres applications de TAL.

Stanford Core NLP :

Stanford Core NLP est une plateforme d'analyse linguistique développée par le Natural Language Processing Group de l'Université Stanford. Elle utilise une combinaison d'approches basées sur des règles et des statistiques pour effectuer différentes tâches de traitement du langage.

La plateforme Stanford CoreNLP utilise des modèles statistiques entraînés sur de grands corpus de textes annotés. Ces modèles permettent d'effectuer des tâches telles que la désambiguïsation morpho-syntaxique (POS tagging) et la reconnaissance d'entités nommées (NER). En plus des modèles statistiques, Stanford CoreNLP intègre également des règles linguistiques préétablies pour améliorer la précision des analyses. Ces règles permettent de capturer des phénomènes linguistiques spécifiques qui pourraient ne pas être bien représentés dans les données d'entraînement ou qui nécessitent une expertise linguistique particulière. Enfin, Stanford CoreNLP intègre également des techniques avancées telles que l'analyse de dépendance syntaxique et la résolution de coréférence. Ces fonctionnalités enrichissent les analyses linguistiques en fournissant des informations sur les relations grammaticales entre les mots et en identifiant les références pronominales dans le texte.

3. Difficultés rencontrées

Nous commençons l'analyse par les difficultés, car elles expliquent les résultats que nous obtenons, et les analyses que nous avons pu ou non réalisées.

Le projet a dû être adapté. Au départ, l'analyse devait porter sur NLTK et LIMA, l'outil d'analyse morpho-syntaxique du CEA. Il y avait plusieurs problèmes, avec l'utilisation des ordinateurs de l'école, et avec l'apparition de caractères spéciaux dans le texte de référence, qui faisait bugger Lima.

Les plateformes NLTK et Stanford n'ont pas la même tokenisation que le fichier de référence. Cela pose un problème dans l'évaluation des plateformes car il y a un décalage de lignes entre le fichier de référence et ceux que l'on créé à partir de la tokenisation des plateformes. Les analyses que nous avons pu produire sont donc superficielles, faites à l'œil nu, plutôt qu'avec une vraie comparaison entre les fichiers produits et le fichier de référence. De plus, l'analyse de Stanford est longue, nous avons donc limité l'analyse aux trente premières lignes, pour que le programme se termine dans un temps raisonnable. Pour pallier ce problème, il aurait fallu avoir une tokenisation NLTK dans le fichier de référence, connaître l'algorithme de tokenisation du fichier de référence, ou améliore le script evaluate.py. Par manque de temps, nous n'avons pu faire la dernière solution, qui aurait été faisable.

Par ailleurs, le fichier evaluate.py a un comportement étonnant. Il compare les mots et les tags, ce qui semble logique, cependant à la lecture des fichiers, il stocke token et tag dans une même chaîne de caractère. En comparant les mots, il compare aussi les tags, et en comparant les tags, il ne compare en réalité rien. Cependant il fournit quand même des valeurs

Affichage du token candidat à chaque itération :

```
E:\1_Documents\Documents\Cours\Po
[('Pierre\tNOUN', '-NONE-')]
[('Vinken,\tNOUN', '-NONE-')]
[('61\tNUM', '-NONE-')]
[('years\tNOUN', '-NONE-')]
[('old,\tADV', '-NONE-')]
[('will\tVERB', '-NONE-')]
[('join\tVERB', '-NONE-')]
[('the\tDET', '-NONE-')]
[('board\tNOUN', '-NONE-')]
[('as\tADP', '-NONE-')]
[('a\tDET', '-NONE-')]
```

4. Evaluation de l'analyse morpho-syntaxique

NLTK :

```
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.009827278141751042
Word recall: 0.009095084979329352
Tag precision: 0.009827278141751042
Tag recall: 0.009095084979329352
Word F-measure: 0.009447015601889402
Tag F-measure: 0.009447015601889402

Process finished with exit code 0
```

Stanford :

```
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.060842433697347896
Word recall: 0.05652173913043478
Tag precision: 0.060842433697347896
Tag recall: 0.05652173913043478
Word F-measure: 0.058602554470323066
Tag F-measure: 0.058602554470323066

Process finished with exit code 0
```

Nous observons, avant de comparer, que très peu de mots et de tags correspondent bien, entre le fichier de référence, et les analyses produites. Ce chiffre anormalement bas s'explique par une tokenisation différente entre les fichiers. La tokenisation de Stanford est la plus proche du texte de référence, c'est pourquoi les mesures y sont plus élevées. Cependant cela ne veut pas dire que Stanford annote effectivement mieux les textes. Nous ne pouvons donc rien retirer de l'analyse via le script evaluate.py. Ici la différence de tokenisation porte sur les groupes de mots. « 61 years old » est un seul token pour le fichier de référence, pas pour NLTK ou Stanford, qui découpe en trois tokens cette chaîne de caractère.

Si nous comparons les fichiers à l'œil nu, nous pouvons observer des changements (outre la tokenisation mais pas tant que ça. Logiquement les analyses fonctionnent quand même, mais nous ne pouvons le voir dans ce projet. Il y a quand même des différences, par exemple le mot « Dutch » dans « the Dutch publishing group » est un nom pour le fichier de référence, mais un adjectif pour les analyses NLTK et Stanford. Dans ce contexte, ce sont les nouvelles analyses qui ont raison. Ces outils sont donc fonctionnels

5. Evaluation de la reconnaissance d'entités nommées

NLTK :

```
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.023043305522447356
Word recall: 0.023043305522447356
Tag precision: 0.023043305522447356
Tag recall: 0.023043305522447356
Word F-measure: 0.023043305522447356
Tag F-measure: 0.023043305522447356

Process finished with exit code 0
```

Stanford :

```
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.1492776886035313
Word recall: 0.1492776886035313
Tag precision: 0.1492776886035313
Tag recall: 0.1492776886035313
Word F-measure: 0.1492776886035313
Tag F-measure: 0.1492776886035313

Process finished with exit code 0
```

Comme précédemment, c'est la tokenisation qui fausse les résultats. A l'œil nu, cependant, les résultats sont beaucoup plus partagés. Les différences d'analyse sont plus marquées, notamment car les analyses ne partagent pas les mêmes tags, donc l'universalité n'existe pas. NLTK utilise le tag « GPE » pour les groupes géopolitiques, Stanford se contente de mettre « LOCATION » ou « ORGANIZATION ». De plus, le fichier de référence englobe plus d'entités nommées. Il prend « The United States », et non pas « United States », comme les analyses que nous avons produites.

6. Points forts et limitations

Sans parler de résultats, nous pouvons déjà trouver un point fort de NLTK sur Stanford : la rapidité d'exécution. Sur quelques lignes, la différence est sensible mais peu importante. Sur un fichier de 500 lignes, la différence se compte déjà en minutes.

En termes de praticité, NLTK est aussi supérieur. Stanford nécessite des fichiers annexes à télécharger, et à lier aux taggers, alors que NLTK ne nécessite que d'installer le module.

Au niveau du code, c'est assez similaire, il n'y a ni avantages ni inconvénients.

Puisque nous ne pouvons comparer avec de vrais résultats, nous nous basons sur les comparaisons visuelles. Les deux bibliothèques font des « choix » d'analyse, que ce soit dans l'analyse syntaxique ou pour la détection d'entités nommées. Nous ne savons pas si ces choix sont bons ou mauvais, mais il faut impérativement les prendre en compte lors de la création d'application incluant de l'analyse de texte (que ce soit pour des chatbots ou pour l'analyse de d'opinions sur les réseaux sociaux)

7. Organisation

Pour la réalisation de ce projet, nous nous sommes réparti le travail suivant les deux analyses à produire. Tout d'abord, nous avons réalisé chacun de notre côté les 3 TPs. Concernant ce projet, nous avons choisi de réaliser l'évaluation d'une plateforme chacun pour les deux tâches que sont la désambiguïsation morpho-syntaxique et la reconnaissance d'entités nommées. Ruben a réalisé l'analyse de la plateforme **Stanford Core NLP** et Hugues celle de **NLTK**. Chacun a alors réalisé sa partie de code en essayant de garder une uniformité de code. Concernant la rédaction du rapport, Hugues a réalisé l'introduction ainsi que la présentation des plateformes linguistiques. Puis chacun a réalisé l'évaluation de la plateforme étudiée pour l'analyse morpho-syntaxique et la reconnaissance d'entités nommées. Ruben s'est quant à lui occupé d'expliquer les points forts, limitations et difficultés ainsi que l'organisation.

Voici le GitHub où vous pourrez trouver notre projet et les TPs.

https://github.com/UggFR/Traitement_Automatique_Langue

8. Conclusion

Ce projet n'est malheureusement pas une grande réussite. Son cœur, l'analyse de performance, n'a pu être fait. Cependant la gestion de fichiers et de structurations des informations est toujours bénéfique et intéressante. Les TPs, qui sont fonctionnels, ont permis de mettre en pratique la théorie. Nous aurions préféré pouvoir réellement fait ce projet mais les aléas des outils que nous devions utilisés ne nous ont pas permis de tout faire.

En somme, le cours et les TPs ont été très intéressants, et serviront par la suite, surtout dans le domaine de l'intelligence artificielle. Ruben fera sans doute sur son temps libre le second sujet, pour s'entraîner. Si le projet n'est pas une réussite, le module de traitement automatique des langues l'est.