Robin Malmasson, Eurydice Ruggieri, Thomas von Ascheberg

Projet Traitement Automatisé de la Langue

Introduction

Les différences de langage ont toujours été un frein à la communication et à la compréhension entre les peuples. De fait, avec plus de 7000 langues parlées et 141 langues officielles selon l'ONU, il n'est pas surprenant de constater que la traduction est et a toujours été un enjeu majeur.

L'Histoire montre que pendant longtemps, l'absence de traduction généralisée des langues a engendré de nombreux conflits et de nombreux préjugés. À l'heure de la mondialisation, la traduction devient un enjeu que les entreprises et les institutions ne peuvent plus négliger. De ce fait, le Traitement Automatisé de la Langue (TAL) et ses applications se développent fortement avec les nouvelles technologies de traitement statistique de l'information (Machine Learning et Deep Learning principalement). Dès lors, la question centrale de ce rapport est d'analyser ces outils modernes.

C'est pourquoi, pour ce rapport, nous avons décidé de comparer 3 plateformes modernes de TAL :

- **CEA List LIMA** (que nous abrégeons en LIMA) : est une plateforme d'analyse linguistique multilingue utilisant des règles et des ressources validées par des experts linguistes.

- **Stanford Core NLP** : est une boîte à outils linguistiques utilisant l'apprentissage statistique à partir de corpus annotés.

- **NLTK** : est une boîte à outils linguistiques utilisant des approches hybrides combinant l'apprentissage automatique et des ressources linguistiques.

Mais avant cela, il est important de comprendre que le Traitement Automatique de la Langue est découpé en 5 étapes :

- 1. Découpage (ou Tokenization) : Ce module consiste à découper les chaînes de caractères du texte en mots, en prenant en compte le contexte ainsi que les règles de découpage. Ce module utilise généralement des règles de segmentation ainsi que des automates d'états finis.

- 2. Analyse morphologique : Ce module a pour but de vérifier si le mot (token) appartient à la langue et d'associer à chaque mot des propriétés syntaxiques qui vont servir dans la suite des traitements. Ces propriétés syntaxiques sont décrites en classes appelées catégories grammaticales. La consultation de dictionnaires de formes ou de lemmes permet de récupérer les propriétés syntaxiques concernant les mots à reconnaître.

- 3. Analyse morpho-syntaxique : Après l'analyse morphologique, une partie des mots restent ambigus d'un point de vue grammatical. L'analyse morphosyntaxique réduit le nombre des ambiguïtés en utilisant soit des règles ou des matrices de désambiguïsation. Les règles sont généralement construites manuellement et les matrices de bi-grams et tri-grams sont obtenues à partir d'un corpus étiqueté et désambiguïser manuellement.

- 4. Analyse syntaxique (Syntactic analysis ou Parsing) : Ce module consiste à identifier les principaux constituants de la phrase et les relations qu'ils entretiennent entre eux. Le résultat de l'analyse syntaxique peut être une ou plusieurs structures syntaxiques représentant la phrase en entrées. Ces structures dépendent du formalisme de représentation utilisé : un arbre syntagmatique, un arbre de dépendance ou une structure de traits. L'analyse en dépendance syntaxique consiste à créer un arbre de relations entre les mots de la phrase. Le module d'analyse syntaxique utilise des règles pour l'identification des relations de dépendance ou des corpus annotés en étiquettes morpho-syntaxiques et en relations de dépendance.

- 5. Reconnaissance d'entités nommées (Named Entity recognition) : Ce module consiste à identifier les dates, lieux, heures, expressions numériques, produits, évènements, organisations, présentes sur un ou plusieurs tokens, et à les remplacer par un seul token.

Dès lors, dans ce rapport, nous essayerons de comprendre où chaque plateforme est performante et où elle l'est moins.

REMARQUE :
Par la surcharge de travail que nous avons eu, nous n'avons remarqué que trop tardivement un bug dans la normalisation de l'analyse lexicale de nos fichiers. Le script evaluate.py évaluant sommairement les deux fichiers, en résulte des résultats médiocres. Cependant, l'erreur étant arrivée sur toutes les plateformes, cela ne change pas les rapports de proportionnalité dans nos chiffres et donc notre analyse.

Sommaire

Description des plateformes

CEA List LIMA

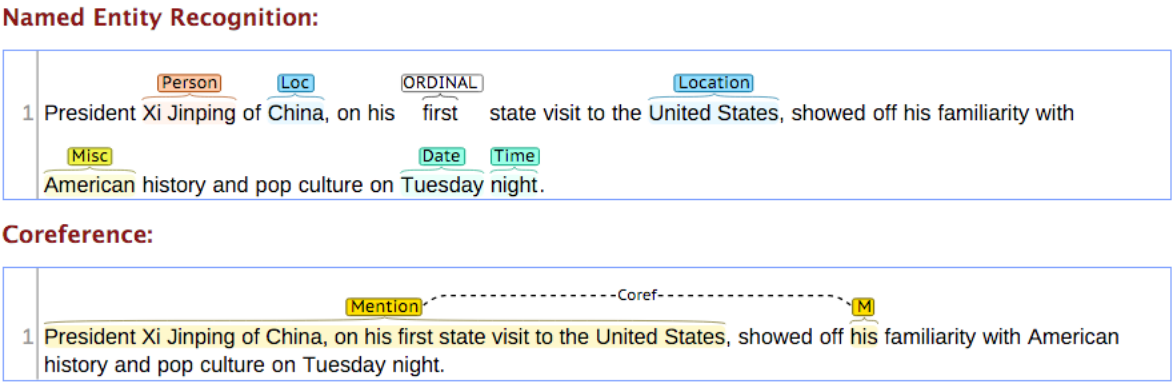
LIMA est développé par le CEA List, Laboratoire LASTI (Laboratoire d'analyse sémantique texte et image). Il a été publié en 2010.
Il existe une version open source et une version commerciale. La version libre fonctionne avec l'anglais, le français et le portugais, mais d'autres langues sont supportées dans la version commerciale, comme l'arabe, le chinois ou l'espagnol.
LIMA fonctionne avec plusieurs outils, qu'il est possible de pipeliner pour obtenir une analyse complète :

- Tokenisation : utilisation d'un automate basé sur des caractères avec une fenêtre de sept caractères. La tokenisation ne tient pas compte de l'ambiguïté. Les étapes ultérieures peuvent regrouper ou fractionner certains tokens.
- Vérification du dictionnaire : chaque token est comparé à un dictionnaire complet (un dictionnaire comprenant toutes les formes connues de mots simples), et des lemmes et parties de discours (PoS) possibles lui sont associés.
- Mots avec trait d'union : cette unité effectue un traitement spécial pour associer des lemmes et des catégories de mots avec trait d'union non présents dans le dictionnaire (des parties du mot divisé sont recherchées dans le dictionnaire).
- Alternatives de répartition des abréviations : les tokens comprenant une citation unique (comme «ne pas », «je suis », etc.) sont recherchés dans le dictionnaire qui indique les tokens dont ils sont faits.
- Expressions idiomatiques : expressions composées reconnaissant. Il réduit l'ambiguïté avant le marquage PoS en considérant l'expression dans son ensemble. Il utilise une unité générique de reconnaissance de formes basée sur des automates à états finis.
- Mots inconnus : les mots inconnus reçoivent un PoS par défaut à l'aide d'un devineur basé sur des indices typographiques.
- Reconnaissance des entités nommées : la même unité de reconnaissance de modèle est utilisée avec différentes règles pour reconnaître les nombres, les dates et les entités nommées.
- Marquage PoS : deux marqueurs PoS sont actuellement implémentés dans LIMA. Le premier historique utilise un algorithme de Viterbi sur les trigrammes PoS. Le second utilise SVMTool ++, un tagger LGPL basé sur SVM. Les deux utilisent des modèles tirés de corpus annotés.
- Analyse syntaxique : utilisation d'une grammaire de dépendance implémentée comme un ensemble de règles simples exécutées par l'unité de reconnaissance de modèle générique.

LIMA utilise plusieurs systèmes de tag selon le résultat voulu, cherchant à obtenir le plus performant. Ainsi, il utilise le Free French Treebank (FFT), une version modifiée GATE's ANNIE system pour l'anglais ainsi qu'une partie du Penn Tree Bank (PTB). Il a ainsi son propre système de tag, composé de plus de 50 étiquettes.

Stanford Core NLP

Stanford Core NLP a été créée par Christopher D. Manning, professeur d'informatique à l'université de Stanford, Californie. La première version a été publiée fin 2010.
Il s'agit d'un module (API), utilisable pour la reconnaissance de langue dans des programmes informatiques.
Stanford est composé de plusieurs outils de reconnaissance du langage, tel que un tagger des parties de discours qui assigne à chaque mot un token (comme verbe ou nom), la reconnaissance d'entité nommée, un parseur naturel qui analyse la structure grammatical des phrases et peut identifier le sujet d'une phrase, un solveur de coréférence, un analyseur des sentiments en assignant à chaque mot des points positifs ou négatifs selon la négativité ou positivité du mot, et l'extraction d'information à partir de texte simple, en extrayant des relations entre les mots.

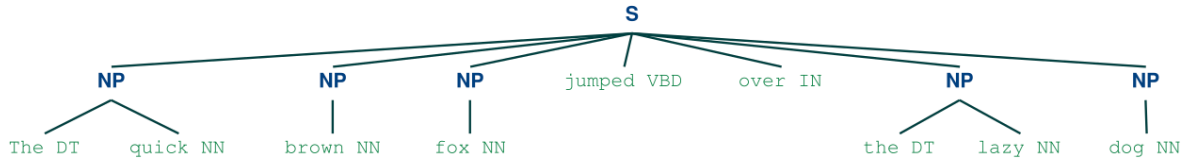


Il est possible de combiner les outils.
Selon les outils, l'analyse est basée sur des statistiques, le deep learning ou des règles.
Stanford supporte actuellement plusieurs langues tel que le français, l'arabe, le chinois, l'anglais, l'allemand et l'espagnol. Il est possible de l'utiliser sur d'autres langues, mais il faut alors développer des modèles de références pour pouvoir utiliser Stanford CoreNLP. Des modèles ont été développés par des personnes extérieures à Stanford pour l'italien, le portugais et le suédois.

Pour l'outil qui nous intéresse ici, le tagger, il est développé en Java et utilise le set de tag Penn Treebank. Il est déterministe, mais utilise aussi des heuristiques pour déterminer si les points sont des fins de phrases, si les guillemets font partie d'un mot...

NLTK

NLTK a été développé par Steven Bird, professeur à l'université de Pennsylvanie, et un de ses anciens étudiants, Edward Loper. La première version a été publié en 2001.
NLTK est un programme open source, tout le monde peut y contribuer en continuant le développement du programme, que ce soit sur son fonctionnement ou en ajoutant un corpus de référence. Il est possible de n'utiliser que le toolkit de traitement du langage, mais NLTK fournit aussi des démonstrations graphiques, des données-échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API).



Le toolkit contient en fait une suite de bibliothèques de traitement de texte pour la classification, la tokenisation, le stemming, le balisage, l'analyse et le raisonnement sémantique.
NLTK combine à la fois l'analyse grâce aux statistiques, mais aussi à base de règles.
NLTK peut être intégré à Stanford CoreNLP pour profiter de certaines fonctionnalités de celui-ci.

Expérimentation

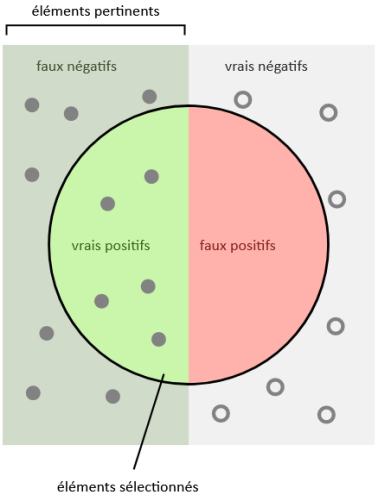
Métriques d'évaluation

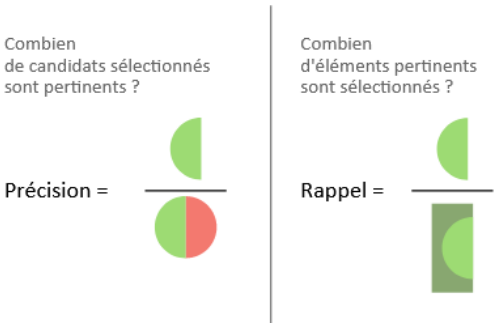
La principale métrique que nous étudierons sera la précision des plateformes. Par précisions, nous désignons la comparaison (mot-à-mot) d'un texte analysé par une plateforme à un corpus de référence. On fait ensuite le rapport nombre de mots identiques divisé par le nombre de mots total.

$$\text{précision}_i = \frac{\text{nb de documents correctement attribués à la classe } i}{\text{nb de documents attribués à la classe } i}$$

La 2e métrique est le rappel. On divise le nombre de de mots corrects par le nombre de faux positifs :

$$\text{rappel}_i = \frac{\text{nb de documents correctement attribués à la classe } i}{\text{nb de documents appartenant à la classe } i}$$





Finalement la dernière métrique la "F-mesure" est juste une moyenne harmonique du rappel et de la précision.

$$F = 2 \cdot \frac{(\text{précision} \cdot \text{rappel})}{(\text{précision} + \text{rappel})}$$

Évaluation de l'analyse morpho-syntaxique

Préparation des données (pré-traitement)

On a réalisé un certain nombre de prétraitement nécessaires pour pouvoir exploiter les données de base et comparer les performances des POS taggers de chaque plateforme :

- Correction de nombreuses incohérences du fichier initial, afin de pouvoir le traiter automatiquement :

    - Changement du tag PROPN pour NOUN
    - Suppression du caractère] à la fin de certaines lignes
    - Suppression des espaces à la fin des lignes
- Conversion des tags du corpus annoté « pos_reference.txt.lima » en tags universels et sauvegarde du résultat dans le fichier « pos_reference.txt.univ ».
- Utilisation du corpus annoté « pos_reference.txt.univ » pour extraire les phrases ayant servi pour produire ce corpus annoté et sauvegarder le résultat dans le fichier « pos_test.txt ». Dans ce corpus, une ligne vide indique la fin de la phrase courante.

On lance ensuite les trois POS taggers sur le fichier « pos_test.txt », en reprenant les TPs fait précédemment.
Enfin, pour pouvoir réaliser l'évaluation du résultat des POS taggers, on convertit les résultats des trois POS taggers en utilisant les étiquettes universelles.
On peut ensuite lancer l'évaluation. On a cependant ajouté un script afin de retirer les lignes vides du fichier de référence (car elles ont été supprimées dans nos scripts).

Résultats

On obtient donc les résultats suivants :
LIMA :

Word precision: 0.0662835628654
Word recall: 0.0615059299439
Tag precision: 0.0662835628654
Tag recall: 0.0615059299439
Word F-measure: 0.0638054363376
Tag F-measure: 0.0638054363376

Stanford :

Word precision: 0.015882469724
Word recall: 0.0146991272393
Tag precision: 0.015882469724
Tag recall: 0.0146991272393
Word F-measure: 0.0152679040031
Tag F-measure: 0.0152679040031

NLTK :

Word precision: 0.0156839388525
Word recall: 0.0145153881488
Tag precision: 0.0156839388525
Tag recall: 0.0145153881488
Word F-measure: 0.015077055203
Tag F-measure: 0.015077055203

Analyse des résultats

Les outils POS tagger de NLTK et Stanford offrent des performances similaires. En revanche, celui de Lima est environ quatre fois plus précis que ces deux derniers. Il semble donc pertinent de s'orienter sur cet outil pour l'étape du POS tagging.

Évaluation de la reconnaissance d'entités nommées

Préparation des données (pré-traitement)

Ici aussi, il a été nécessaire de réaliser un certain nombre de prétraitement nécessaires pour pouvoir exploiter les données de base et comparer les performances des NE recognizers de chaque plateforme :

- Utilisation du corpus annoté « 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 ». Dans ce corpus, une ligne vide indique la fin de la phrase courante.

On peut ensuite lancer les trois NE recognizers sur le fichier « ne_test.txt ». Les résultats doivent avoir le format du corpus annoté « ne_reference.txt.conll » (c'est à dire 2 colonnes séparées par une tabulation).
On doit ensuite convertir les résultats des trois NE recognizers en utilisant les étiquettes CoNLL-2003, afin de pouvoir comparer les résultats et lancer l'évaluation dessus.
On lance cette évaluation et encore une fois, on a ajouté un script afin de retirer les lignes vides du fichier référence car elles ont été supprimées dans nos scripts.

Résultats

On obtient donc les résultats suivants :

LIMA :

Word precision: 0.0114150555327
Word recall: 0.0115951112504
Tag precision: 0.0114150555327
Tag recall: 0.0115951112504
Word F-measure: 0.011504378919
Tag F-measure: 0.011504378919

Stanford :

Word precision: 0.00923718712753
Word recall: 0.00923718712753
Tag precision: 0.00923718712753
Tag recall: 0.00923718712753
Word F-measure: 0.00923718712753
Tag F-measure: 0.00923718712753

NLTK :

Word precision: 0.0205601907032
Word recall: 0.0205601907032
Tag precision: 0.0205601907032
Tag recall: 0.0205601907032
Word F-measure: 0.0205601907032
Tag F-measure: 0.0205601907032

Analyse des résultats

Contrairement à l'évaluation des POS taggers, ici c'est l'outil NLTK qui offre la meilleure précision (environ deux fois supérieure aux deux autres outils). On en conclut qu'il peut être intéressant d'utiliser des outils de différentes sources aux différents stades de la Traduction Automatique de la Langue afin de maximiser la précision du résultat final.

Conclusion

Résumé du travail fait

Étant donné que l'ordinateur personnel de Robin a pu exécuter les TP en temps voulu, nous avons pu faire tous les TP (1,2 et 3). De plus nous avons analysé toutes les plateformes (NLTK, LIMA, Stanford) et répondu à toutes les questions du sujet. En somme, tout est fait.

Répartition du travail

Étant donné que Eurydice et Thomas ne pouvaient pas faire les TP (puisqu'ils n'ont pas de PC portable), c'est Robin qui s'est occupé du code lié aux plateformes. Dans ce cadre, Eurydice et Thomas se sont principalement occupés des scripts annexes (non liés directement à la plateforme) pour la mise en forme notamment des données, et de la rédaction de ce rapport. La rédaction du rapport c'est fait en commun.

Limitation de chaque plateforme

Au niveau d'analyse où nous avons été, il est difficile pour nous de critiquer l'une ou l'autre plateforme, car nous n'avons pas une connaissance approfondie et une expérience profonde de ces outils. Ainsi, nous n'avons pas remarqué de limitation significative aux plateformes utilisées.

Pistes d'amélioration pour chaque plateforme

Il faut savoir rester humble face à de tels monstres technologiques. Étant donné notre niveau actuel en TAL, il nous paraît difficile de critiquer et d'imaginer des pistes d'améliorations crédibles aux plateformes que nous avons utilisées.

Bibliographie

Langue (Wiki) : https://fr.wikipedia.org/wiki/Langue
Informations sur LIMA :

- G. de Chalendar. The LIMA multilingual analyzer made free: FLOSS resources adaptation and correction. Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014, May 2014, Reykjavik, Iceland. pp.2932-2937.
- https://github.com/aymara/lima/wiki

Informations sur Stanford : https://stanfordnlp.github.io/CoreNLP/
Information sur NLTK : https://www.nltk.org/ , https://github.com/nltk/nltk/wiki

Explication des métriques : https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel