HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Cours TAL – Labo 3 : analyse syntaxique (v2)

Distribué le mardi 12 mars 2019, pour le vendredi 29 mars 2019

Objectifs

1. Appliquer un analyseur syntaxique de Stanford sur les données de test UD en français, puis – si le temps le permet – entraîner cet analyseur sur les UD 'train', et en étudier l'amélioration.
2. Importer les fichiers UD dans NLTK, les transformer en graphes de dépendances, et trouver les paires sujet-verbe les plus fréquentes.
3. En appliquant le parser de Stanford en constituants, extraire tous les groupes nominaux que cet outil détecte parmi les phrases test du corpus UD.

Informations

- Ce labo est à effectuer et à rendre en binôme. Merci d'envoyer votre *notebook* Jupyter (dans un fichier ZIP), sous forme de rapport incluant code, résultats, et discussion) avant le **ven. 29 mars à 23h59** par email à andrei.popescu-belis@heig-vd.ch et quentin.gliosca@heig-vd.ch.
- Le corpus est toujours à https://drive.switch.ch/index.php/s/5ZNllZOApTWHGwH

Exercices

1. Appliquer l'analyseur syntaxique de dépendances (*dependency parser*) de Stanford avec les modèles français sur les données de test UD en français, puis entraîner l'analyseur sur les données UD 'train', et observer si la performance s'est améliorée ou non.
   - La documentation pour ce parser se trouve à https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/parser/nndep/DependencyParser.html (regarder le 'main' et ses exemples à la fin), alors que la description générale des modèles inclus est à https://nlp.stanford.edu/software/lex-parser.html
   - Pour l'exécuter, vous devez indiquer à java : -cp stanford-corenlp-3.9.2.jar et vous pouvez récupérer ce fichier depuis http://central.maven.org/maven2/edu/stanford/nlp/stanford-corenlp/3.9.2/
   - Le modèle pré-entraîné fourni par Stanford (UD_French.gz) est dans le même dossier au sein d'un .jar très grand, mais pour gagner du temps il est disponible à https://drive.switch.ch/index.php/s/OqlSRUCSBvqKg3O
   - La documentation vous permettra de tester le parser sur un fichier annoté (et donc obtenir directement les scores), de l'appliquer à un fichier texte pur (mais rien n'est demandé sur ce point), ou de générer un nouveau modèle à partir des données d'entraînement.
   - Pour effectuer l'entraînement, plusieurs options indiquées dans la documentation peuvent être utiles : « -wordCutOff 3 » (pour traiter seulement les mots apparaissant plus de 3 fois, ce qui évite le problème des nombres – uniques – avec un espace), « -trainingThreads 8 » (pour

utiliser pleinement son processeur, indiquer le maximum selon le modèle), « -maxIter 5000 » (pour arrêter l'entraînement après 5000 itérations).

- La question est : quel est le score du modèle fourni, et quel est le score du modèle que vous avez entraîné (si le temps l'a permis) ?
- Indiquer les étapes réalisées, les commandes, les scores obtenus dans votre rapport.

2. En utilisant la classe nltk.parse.DependencyGraph qui permet de transformer une phrase au format CONLL (plusieurs lignes avec les mots et leurs annotations) en un graphe de dépendances, extraire des données UD toutes les paires NOM + VERBE, à savoir la relation 'nsubj'.
   - Considérez le début de la page http://www.nltk.org/howto/dependency.html pour transformer une phrase au format CONLL en un graphe de dépendances (un objet de la classe DependencyGraph). On montre aussi comment on accède aux informations de ce graphe.
   - Vous devrez donc lire le(s) fichier(s) UD phrase par phrase (attention aux espaces dans les nombres), et créer un graphe de dépendance pour chaque phrase. Pour pouvoir le parcourir, indiquer `DependencyGraph(bloc, top_relation_label='root').`
   - Il faut ensuite extraire les triplets ayant une relation 'nsubj' (entre sujet et verbe). Quelles sont les 10 triplets les plus fréquents dans tout le corpus ?
   - Note : on utilise ici l'annotation fournie avec le corpus, mais on aurait pu aussi effectuer l'analyse syntaxique avec le parser de Stanford.

3. Installer l'outil CoreNLP de Stanford *avec les modèles de langue pour le français*. Démarrer le serveur qui permettra à NLTK d'interroger le parser.
   - Pour consulter les outils CoreNLP depuis NLTK, la solution la plus récente consiste à démarrer un serveur CoreNLP, soit depuis NLTK soit en ligne de commande. Pour le démarrer, le serveur doit savoir où se trouvent le code et les modèles, qu'on indique ainsi : java -mx4g -cp "stanford-corenlp-3.9.1.jar;stanford-french-corenlp-3.9.2-models.jar" (en supposant que les deux .jar sont dans le dossier du notebook). Le 2e jar (277 MB !) se trouve à http://central.maven.org/maven2/edu/stanford/nlp/stanford-corenlp/3.9.2/
   - Pour le démarrage du serveur : https://stanfordnlp.github.io/CoreNLP/corenlp-server.html
   - L'utilisation du serveur depuis NLTK se fait en créant une instance de CoreNLPParser (c'est le parser en constituants, différent de celui de dépendances) voir https://www.nltk.org/api/nltk.parse.html#nltk.parse.corenlp.CoreNLPParser
   - Pour chaque phrase, le résultat est un objet 'Tree' auquel on peut appliquer plusieurs fonctions (voir https://www.nltk.org/_modules/nltk/tree.html#Tree).
   - On vous demande d'extraire tous les groupes nominaux (NP) du corpus UD test donné, et indiquer les 10 plus fréquents.