Université de Montréal

IFT6285 (TALN) — Devoir4
Analyse syntaxique avec NLTK/CYK

Contact :
Philippe Langlais
RALI/DIRO
Université de Montréal

+1 514 343 61 11 ext: 47494
felipe@iro.umontreal.ca
http://www.iro.umontreal.ca/~felipe/

▮ dernière compilation : 10 novembre 2023 (09:53)

Contexte

NLTK est une plateforme bien connue du traitement des langues qui offre de nombreuses implémentations d'algorithmes faciles à utiliser ainsi qu'un accès à une cinquantaine de corpus populaires. Dans ce devoir, vous allez explorer l'analyse en constituant via une variante de l'algorithme CYK vu en cours. NLTK vous donne accès à de nombreux analyseurs qui sont décrits ici. Vous aurez également besoin de quelques connaissances sur la façon dont une grammaire est représentée que vous trouverez ici.

Données

Le Penn Tree Bank (PTB) est une ressource payante distribuée par LDC (Linguistic Data Consortium) offrant des phrases manuellement arborées syntaxiquement et NLTK vous offre un accès gratuit à un sous-ensemble de ces phrases. Vous pouvez obtenir les arbres de ces phrases comme suit :

```
from nltk.corpus import treebank

for item in  treebank.fileids():
    for tree in treebank.parsed_sents(item):
         print(tree)
```

Vous devez également utiliser les données de la tâche `CoLA` du benchmark GLUE avec lequel vous travaillez dans votre projet 1. Une copie de ce jeu de données est disponible sur les machines du DIRO ici :

```
ls -l /home/www-labs/felipe/public_html/IFT6285-Automne2021/CoLA/
total 524
-rw-r--r-- 1 felipe rali  53717 Oct 19 15:00 dev.tsv
-rw-r--r-- 1 felipe rali  48788 Oct 19 15:00 test.tsv
-rw-r--r-- 1 felipe rali 428590 Oct 19 15:00 train.tsv
```

À faire

1. Installez NLTK et les données (au moins le Penn Treebank, 44è item de cette liste).
2. Prenez connaissance de cette documentation simple sur les facilités principales d'analyse syntaxique offertes par NLTK.
3. Utilisez les arbres des phrases du corpus PTB pour entraîner une grammaire PCFG. Vous pouvez utiliser la fonction `induce_pcfg` pour cela. Un exemple de code se trouve aux cellules 20 et 21 de la section PCFG de la documentation sus-mentionnée. Comme vous le constaterez, le nombre de règles peut être important, aussi pouvez vous garder un nombre donné de règles (les plus fréquentes par exemple) avant d'apprendre la grammaire (les probabilités associées aux règles).
4. Utiliser cette grammaire pour analyser les phrases grammaticalement problématiques du corpus dev de `CoLA`, c'est-à-dire celles marquées d'une étoile. Vous utiliserez l'analyseur ViterbiParser qui ne requiert pas des règles de production au format CNF.

   Vous remarquerez très rapidement que la présence d'un mot inconnu dans une phrase à analyser lève une exception. Vous devez régler ce problème de façon à pouvoir analyser une phrase contenant des mots inconnus. Une solution possible consiste à ajouter des règles A $\rightarrow$ UNK pour tout non terminal A intervenant dans une règle lexicale (e.g. A $\rightarrow$ the) et à remplacer les mots inconnus de votre grammaire par le mot UNK préalablement à l'analyse.
5. Produisez un rapport (format pdf, en français ou en anglais) qui contient les informations suivantes :
   (a) en un paragraphe, votre façon de gérer les mots inconnus de façon à pouvoir analyser les phrases en contenant.
   (b) La longueur moyenne des phrases du corpus PTB distribué dans NLTK (utilisez la fonction `leaves` qui retourne sous forme de liste les mots d'un arbre en constituant d'une phrase) ainsi que la longueur moyenne des phrases agrammaticales du dev de `CoLA`.
   (c) Une ou plusieurs figures (ou tables) rendant compte du temps d'analyse de `ViterbiParser` en fonction de la longueur des phrases analysées, le nombre d'analyses produites pour une phrase, ainsi

que le nombre de phrases non reconnues par la grammaire. Étudiez l'impact du filtrage de la grammaire (par exemple en retenant les règles les plus fréquentes). Vous avez au plus une page pour décrire cette analyse.

Ne soyez pas surpris, la grammaire obtenue du PTB contient environ 22k règles. Analyser à l'aide de cette grammaire des phrases de 30 mots peut prendre plusieurs minutes avec le code non optimisé de NLTK ... Réduire le nombre de règles réduit les temps de réponse (probablement au détriment des performances). Vous pouvez faire votre analyse sur une partie seulement des phrases du dev de `CoLA`.

(d) Définissez une taille maximale de phrases (en nombre de mots, par exemple 15). Analysez avec votre grammaire toutes les phrases agrammaticales du dev set de `CoLA` qui vérifient ce critère de longueur. Mentionnez le nombre de phrases pour lesquelles une analyse est retournée, en fonction de la longueur des phrases et des mots inconnus de votre grammaire. Montrez deux analyses de phrases agrammaticales. Vous pouvez utiliser la fonction NLTK `draw_trees` pour cela.

(e) En au plus 2 paragraphes, décrivez des traits (features) que vous pourriez définir à l'aide de votre analyseur afin de détecter si des phrases sont ou pas grammaticalement correctes.

Remise

La remise est à faire sur Studium sous le libellé `devoir4` sous la forme d'une archive de nom `devoir4-<noms>.tar|tar.gz|zip|gzip` où `<noms>` est à remplacer par le nom des personnes concernées par la remise. Cette archive doit contenir votre code, votre rapport (de nom `devoir4-rapport-<noms>.pdf` au format pdf, texte en anglais ou en français).

Le devoir est à remettre en groupe d'au plus deux personnes au plus tard jeudi 23 novembre à 23h59.