IFT 607 : Devoir 3
Travail individuel

Remise : 21 novembre 2014, 12h00 (**au plus tard**)

Ce devoir comporte 2 questions de programmation. Vous trouverez tous les fichiers nécessaires pour ce devoir ici : `http://info.usherbrooke.ca/hlarochelle/cours/ift607_A2014/devoir_3/devoir_3.zip`.

Veuillez soumettre vos solutions à l'aide de l'outil **turnin** :

```
turnin -c ift607 -p devoir_3 solution_classification.py solution_cky.py
```

1. [**5 points**] Programmez un classifieur de documents par régression logistique et pondération tf-idf.
   Le programme doit être écrit dans le langage Python. Plus spécifiquement, vous devez compléter les fonctions et méthodes du fichier `solution_classification.py` disponible sur le site web du cours. Vous avez à compléter les fonctions `extraire_vocabulaire_ordonne`, `remplacement_unk`, `sac_de_mots`, `idf`, `tfidf`, ainsi que les méthodes `entrainement` et `prediction` de la classe `Classifieur`.
   Tous les détails sur ces fonctions et méthodes sont contenus dans leur *docstring*.
   Le script Python `classification.py` importera `solution_classification.py` (qui doit être dans le même répertoire) et l'utilisera afin d'entraîner un classifieur par régression logistique sur un corpus de critiques de films, représenté selon un sac de mots et selon la représentation TF-IDF.
   Pour utiliser le corpus de critiques de films ("movie_reviews"), vous devez installer la librairie *nltk*, comme suit :

   ```
   pip install --user nltk
   ```

   Ensuite, vous devez télécharger le corpus "movie_reviews". Pour ce faire, exécuter les instructions Python suivantes (par exemple via l'interpréteur) :

   ```
   import nltk
   nltk.download('movie_reviews')
   ```

   Voici comment utiliser le script `classification.py` :

   ```
   Usage: python classification.py

   Une comparaison sera faite avec un cas pour lequel les résultats
   attendus sont connus.
   ```

2. [**5 points**] Programmez l'algorithme CKY pour l'analyse syntaxique. L'algorithme doit retourner l'arbre sous la forme d'une chaîne de caractères, selon la *bracketed notation*, telle :

```
[S [NP I] [VP [Verb prefer] [NP [Determiner a] [Nominal [Nominal morning] [Noun flight]]]]]
```

Le programme doit être écrit dans le langage Python. Plus spécifiquement, vous devez compléter la fonction `cky` du fichier `solution_cky.py` disponible sur le site web du cours. Tous les détails sur cette fonction sont contenus dans sa *docstring*.

Le script Python `cky.py` importera `solution_cky.py` (qui doit être dans le même répertoire) et l'utilisera afin d'exécuter l'algorithme CKY sur une phrase donnée.

La grammaire hors contexte utilisée par l'algorithme CKY est la suivante :

```
S -> NP VP
NP -> Determiner Nominal | me | I | you | it | Alaska | Baltimore | Los Angeles |
      Chicago | United | American
Nominal -> Nominal Noun | flight | breeze | trip | morning
VP -> Verb NP | Verb NPPP | Verb PP | is | prefer | like | need | want | fly
NPPP -> NP PP
PP -> Preposition NP
Noun -> flight | breeze | trip | morning
Verb -> is | prefer | like | need | want | fly
Pronoun -> me | I | you | it
Determiner -> the | a | an | this | these | that
Preposition -> from | to | on | near
```

À noter que cette grammaire est déjà en forme normale de Chomsky.

Voici comment utiliser le script `cky.py` :

```
Usage: python cky.py [mot1 mot2 ...]

Si aucun argument n'est donné, une comparaison
sera faite avec un cas pour lequel les résultats
attendus sont connus.

Optionnellement, une phrase, spécifiée mot à mot,
peut être fournie. Le programme retournera alors
son arbre syntaxique. Si la phrase ne fait pas partie
de la grammaire, une chaîne vide ('') doit être retournée.
```