IFT 607 : Devoir 1
Travail individuel

Remise : 3 octobre 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_1/devoir_1.zip`.

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

```
turnin -c ift607 -p devoir_1 solution_distance.py solution_ngramme.py
```

1. [**5 points**] Programmez l'algorithme de la distance d'édition minimale entre deux chaînes de caractères. Le programme doit être écrit dans le langage Python. Plus spécifiquement, vous devez remettre un fichier `solution_distance.py` contenant une fonction nommée `min_edit_distance` et une autre nommée `illustration_alignement`.
   La fonction `min_edit_distance` prend en argument deux chaînes (cible et source) ainsi que trois fonctions déterminant les coûts d'édition, puis retourne la valeur de la distance d'édition ainsi qu'une matrice (tableau NumPy à 2 dimenions) de `bool`, déterminant quels caractères de la source et de la cible sont alignés ensemble. Spécifiquement, la valeur du tableau à la rangée `i` et la colonne `j` doit être `True` si et seulemnet si le caractère `target[i]` est aligné avec le caractère `source[j]`.
   IMPORTANT : Afin d'obtenir les résultats attendus, il faut utiliser la priorisation suivante entre les opérations d'édition : substitution, insertion, puis suppression. En d'autres mots, lors du calcul de l'alignement, dans le cas où plus d'une opération obtient la même distance, la substitution est choise en priorité, puis l'insertion.
   La fonction `illustration_alignement` a comme argument la chaîne cible et source ainsi que le tableau d'alignement retourné par la fonction `min_edit_distance`. Elle retourne une chaîne de caractères illustrant visuellemnet l'alignement, tel que vu dans le cours :

   ```
   INTE*NTION
   ||||||||||
   *EXECUTION
   dss is
   ```

   Voir le fichier `solution_distance.py` donné comme patron de départ, pour plus de détails sur ces fonctions. En particulier, les *docstrings* des deux fonctions détaillent leurs arguments et sorties.
   Le script Python `distance.py` importera les fonctions `min_edit_distance` et `illustration_alignement` contenue dans `solution_distance.py` (qui doit être dans le même répertoire) et les utiliseront afin de calculer la distance d'édition et afficher les résultats.
   Voici comment utiliser ce script :

   ```
   Usage: python distance.py [cible source]

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

   Optionnellement, des chaînes "cible" et "source" peuvent
   être fournies, afin d'exécuter le script sur ces chaînes,
   plutôt que le cas par défaut.
   ```

2. [**5 points**] Programmez des modèles de langue trigramme avec lissage *add-delta* et par interpolation. Le programme doit être écrit dans le langage Python. Plus spécifiquement, vous devez compléter les fonctions et méthodes du fichier `solution_ngramme.py` disponible sur le site web du cours. Vous avez à compléter les fonctions `extraire_vocabulaire` et `remplacement_unk`, la méthode `log_probabilite_phrase` de la classe parent `Trigramme`, ainsi que les méthodes `entrainement` et `log_probabilite_mot` des classes enfants `TrigrammeAddDelta` et `TrigrammeInterpolation`.
   Tous les détails sur ces fonctions et méthodes sont contenus dans leur *docstring* (voir les méthodes de la classe parent pour les méthodes de `TrigrammeAddDelta` et `TrigrammeInterpolation`).
   Le script Python `ngramme.py` importera `solution_ngramme.py` (qui doit être dans le même répertoire) et l'utilisera afin d'utiliser les deux modèles de langue sur le corpus Brown.
   Pour utiliser le corpus Brown, vous devez installer la librairie *nltk*, comme suit :

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

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

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

   Voici comment utiliser le script `ngramme.py` :

   ```
   Usage: python ngramme.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
   la log-probabilité de cette phrase.
   ```