TP1 - Traduction automatique
Modélisation statistique du langage
M2 – 2017/2018

Contexte :

Les modèles de langage permettent de caractériser et d'évaluer la qualité des énoncés en langue naturelle. Ces modèles ont un rôle fondamental dans plusieurs applications comme la traduction automatique et la reconnaissance de la parole, la reconnaissance de l'écriture manuscrite, la reconnaissance de langue et l'extraction et la recherche d'information etc.

Ces modèles de langage probabilistes reposent le plus souvent sur un paradigme où la probabilité d'un événement linguistique est estimée en observant cet événement sur un corpus de texte de taille suffisante.

Les modèles *n-grammes* sont utilisés couramment pour la traduction automatique. La probabilité d'une phrase est estimée à partir des probabilités conditionnelles d'apparition d'un mot ou d'une classe de mots, étant donnés les *n-1* mots ou classes de mots précédents. Cette approche est particulièrement intéressante pour son efficacité et sa robustesse. L'objectif de ce TP est d'entrainer, d'évaluer et de comparer différents modèles de langage *n-grammes* qui sera utilisé par la suite dans un système de traduction statitique.

Outils :

1. L'outil SRI-LM : permet, entre autre, de créer et d'utiliser des modèles de langage *n-grammes. L'outil est open source et téléchargeable ici :*
http://www.speech.sri.com/projects/srilm/download.html

N.B: SRILM est installé sur les serveurs

2. Données d'apprentissage : /info/home/bougares/bitexts/
   Données d'évaluation : /info/home/bougares/dev

Attention on n'a pas la même partie cible (anglais) des corpus ***news-commentary*** pour toutes les paires de langue disponibles. Vous pouvez donc créer plusieurs LMs et faire l'interpolation entre eux.

Travail à réaliser

Exercice 1 : Construire un ou plusieurs modèles de langage trigramme pour chaque fichier d'apprentissage :

Calculer la perplexité sur votre corpus de développement en utilisant la commande suivante :

```
ngram –order 3 -debug 2 –unk -lm votre_modèle -ppl dev.en
```

- **Q1 :** Observez les fichiers .voc, .bo et .BO ; que contiennent ils ?
- **Q2 :** À quoi servent les options -gt2min et -gt3min? Dans quels cas elles sont utiles/nécessaires?
- **Q3 :** Faites varier la valeur des paramètres gt2min et gt3min: quelle est l'influence de ces paramètres sur les perplexités calculées ?
- **Q4 :** Quel est le trigramme le plus fréquent dans chacune des langues ?
- **Q5:** Montrer graphiquement l'impact sur les perplexités pour plusieurs tailles de fichier d'apprentissage.
- **Q6:** Montrer graphiquement l'impact sur les perplexités pour plusieurs ordres de n-gramme.
- **Q7:** Faire une interpolation entre plusieurs modèles ? quelles sont les poids d'interpolation obtenus ?
- **Q8:** Calculer la perplexité de votre modèle sur un corpus de développement d'une autre langue.

Remarque : La première colonne du fichier `.BO` représente le logarithme (base 10) de la probabilité conditionnelle du *n-gram* afficher dans la deuxième colonne. La troisième colonne représente le logarithme (base10) du poids *backoff* de même *n-gram*.

Exercice 2 : Refaire le même système avec des modèles n-grammes de CARACTÈRES.

- **Q1 :** Quelles sont les perplexités pour ce modèle de caractères ? Quels pourraient être les avantages/inconvénients du modèle de caractères par rapport au modèle de mots ?
- **Q2 :** Refaire le modèle en remplaçant l'espace par un symbole quelconque (p. ex., <SPACE>). Quelles sont les perplexités pour ce modèle ?