Ecole **N**ationale des **S**ciences **A**ppliquées - **A**l Hoceima

Travaux pratiques N°3 : Modèle de langue

Module : Traitement Automatique de la Langue Naturelle (NLP)

Deuxième Année Ingénierie des données

Préparé par : Pr. Tarik BOUDAA

Travail à faire

L'objectif principal de cet exercice est d'implémenter des modèles de langue n-gramme et d'évaluer leurs performances, et ce, en utilisant les équations vues dans le cours. Ces modèles seront par la suite appliqués à la génération du texte, la correction automatique, et l'auto-complétion d'un texte.

I- Première partie

Vous allez compléter le code donner dans le fichier *language_modeling.py* en implémentant les méthodes décrites dans la suite de ce TP.

Méthodes *prepare_data(self, infile, ngram_size=2)*

Cette méthode prend en entrée un fichier texte représentant le corpus, effectue la séparation des mots (*tokenization*), normalise le texte et ajoute les jetons de début et de fin de phrases. Cette méthode permet également de prendre en compte les mots (les *tokens*) hors vocabulaire. Pour ce faire, cette méthode recherche les mots qui apparaissent moins de N fois dans les données d'entraînement et elle les remplace par <UNK>. Cette méthode retourne le corpus prétraité sous forme d'une chaîne de caractères dont les tokens sont séparés par espace.

Méthode *train(self, ngram_size=2, infile)*

Permet de calculer les probabilités des bigrammes et trigrammes. Le corpus à utiliser est donnée dans *ngramv1.train.* Utiliser la méthode *prepare_data* pour le prétraitement.

Votre modèle doit prendre en compte les différents points étudiés dans le cours :

- Prise en compte du traitement des mots hors vocabulaire.
- Lissage (Utilisez le lissage *add-k* dans ce calcul).
- Transformer les probabilités en logarithmes comme discuté dans le cours pour éviter les problèmes de débordement des nombres flottants.

Méthode *predict_ngram(self, sentence, ngram_size=2)*

Cette méthode prend en paramètre une phrase (sous forme de chaîne de caractères), elle utilise la méthode ***prepare_data*** pour le prétraitement puis calcule la probabilité de la phrase en utilisant les modèles linguistique *bigramme* ou *trigramme*, selon la valeur d'un paramètre *ngram_size*.

Méthode test_perplexity(self, test_file, ngram_size=2)

Cette fonction prend le chemin d'un nouveau corpus (*ngramv1.test*) et calcule la perplexité par rapport à ce corpus de test. Et ce, en utilisant les modèles linguistique *bigramme* ou *trigramme*, selon la valeur du paramètre *ngram_size*.

Des détails supplémentaires sont donnés dans le code squelette.

II- Deuxième partie

Dans cette partie vous allez utiliser de taille plus grande pour l'apprentissage et l'évaluation des modèles ngrammes (Utiliser les données dans *big_data.txt*).

Vous êtes invités à analyser les corpus à utiliser pour savoir les différents prétraitements à faire pour construire les données à utiliser par votre modèle et les éventuelles mise à jour à faire sur la méthode *prepare_data*.

Vous allez réaliser à nouveau les étapes d'apprentissage et d'évaluation des modèles n-grammes avec le nouveau corpus.

Implémenter les fonctionnalités décrites ci-dessous :

Méthode generateText

Cette méthode permet de générer un texte en utilisant les modèles construits précédemment.

Pour générer une phrase avec un modèle bigramme, par exemple, commencez par le jeton <s> et échantillonnez le mot suivant proportionnellement à sa probabilité de suivi de ce jeton. Vous pouvez utiliser, par exemple, la fonction *numpy.random.choice*. Par la suite, examinez tous les mots qui suivent ce mot, et échantillonnez à nouveau. Continuez l'échantillonnage jusqu'à ce que vous générer le jeton de fin de phrase </s>, moment auquel vous terminez la génération.

Méthode autoComplete
Cette méthode prend un texte en paramètre et permet de prédire le mot suivant le plus probable

Méthode correction

Implémente une variante du correcteur d'orthographe réalisé dans le TP1 en utilisant un modèle de langue bigramme pour ordonner les corrections possibles par leurs probabilités selon le modèle de langue et selon la distance minimale d'édition.