TP semaine 3 : modèle de langue par n-gramme

On commence le TP avec la valeur par défaut $n = 2$, mais on veillera à ce que n soit un paramètre, de façon qu'on puisse facilement tester le code avec d'autres valeurs de n.

1 Modèle de base

1.1 Construction de la structure de données

Avec le corpus français de votre choix, pris par exemple dans la collection gutemberg, segmenté en phrases (délimitées par les pseudo tokens '`<s>`' et '`</s>`') et tokenisé (avec spacy ou avec nltk), pour une valeur de n fixée, remplir un dictionnaire python (faisant le décompte pour chaque $(n-1)$-gramme de tous les mots qui apparaissent après ce $(n-1)$-gramme (et son nombre d'occurrences).

1.2 Fonction de prédiction

À partir de ces décomptes (comptages bruts d'occurrences), on peut écrire une fonction de prédiction, qui, étant donné un contexte (on acceptera un contexte de longueur quelconque mais la fonction ne prendra en compte que les $n-1$ derniers mots), renvoie le mot qui a le score le plus élevé (sans passer par les probabilités). Cette fonction ne peut pas toujours faire de prédiction, puisqu'il peut arriver que $n-1$-gramme pertinent dans le contexte n'ai jamais été rencontré. Dans ce cas, on prédira conventionnellement le token '`XXX`'.

1.3 Evaluation intrinsèque : exactitude

En prenant des corpus de test de 200 à 500 occurrences, calculer le score d'exactitude du modèle, avec n=2, sur :

1. un texte tiré du corpus d'« apprentissage »
2. un texte du même auteur mais non vu à l'apprentissage
3. un texte complètement différent (p.ex. wikipedia)

2 Repli (backoff), lissage (smoothing), perplexité

Pour cette deuxième partie on prend par défaut la valeur $n = 3$.

2.1 Repli

Au lieu de répondre '`XXX`' quand on n'a jamais rencontré le $(n-1)$-gramme, on peut mettre en oeuvre une stratégie de repli, qui consiste à considérer comme contexte le $(n-2)$-gramme, s'il a déjà été rencontré, et ainsi de suite jusqu'à l'unigramme. Définir et évaluer le nouveau modèle ainsi obtenu.

2.2 Lissage

Le lissage consiste à redistribuer une partie de la masse de probabilité sur les évènements qu'on n'a jamais rencontrés à l'entraînement. En pratique, cela signifie que l'on doit déterminer le vocabulaire total et utiliser une des techniques de lissage connues pour attribuer une (faible) probabilité à tous les évènement jamais rencontrés. Essayer la méthode de Laplace (+1 avant normalisation). Est-ce qu'un modèle avec lissage peut se dispenser de faire du repli ?

2.3 Evaluation intrinsèque : Perplexité

Après avoir transformé les comptes d'occurrence en probabilité (après lissage), calculer la perplexité du modèle sur les trois corpus de tests de la question 1.3.

3 Génération auto-régressive

3.1 Génération gloutonne

En prenant comme principe que l'on génère pour chaque contexte le mot ayant la plus forte probabilité (le processus est donc déterministe), et que la génération s'arrête quand le symbole '`</s>`' (ce qui est une simplification drastique), tester la capacité de génération de votre modèle, en prenant d'une part comme point de départ le token '`<s>`' puis en tirant au hasard des $(n-1)$-grammes dans un corpus de test.

3.2 Génération avec échantillonage aléatoire

Au lieu de prédire systématiquement le mot ayant la plus forte probabilité, faire un tirage aléatoire basé sur la distribution de probabilité, pour tester (dans les mêms conditions que la question 3.1) les capacités génératives du modèle.