Opérations matricielles et recherche d'information

8 décembre 2020

—Lexicologie, terminologie, dictionnairique—

Exercices TAL : Vecteurs de mots

Téléchargez le fichier exo1.5.py depuis `https://github.com/gkata/lex20` *qui calcule les fréquences des mots et des bigrams. Faites une copie, il pourra servir comme point de départ pour les exercices qui suivent.*

1. Récupérez une liste des 1000 mots les plus fréquents, en ordre décroissant (après filtrage des stopwords et suppression des mots avec une fréquence inférieure à 5).
2. Créez une matrice de co-occurrence 1000x1000 dans laquelle les lignes et les colonnes correspondent aux indices des mots dans la liste de fréquence en gardant l'ordre, et les cases i, j contiennent le nombre de co-occurrence bigramme des mots i et j (l'un à côté de l'autre, n'importe quel ordre).
3. Ecrivez une fonction qui demande à l'utilisateur d'entrer une requête d'un mot (standard input), et retourne le vecteur *one-hot* qui correspond à ce mot. P. ex. si l'utilisateur rentre le mot qui est à l'indice 100 dans la liste de fréquence, le vecteur prendra la valeur 1 à l'indice 100, et zéro partout ailleurs. Essayez d'éviter de parcourir le fichier topfreq ou la liste des mots de nouveau avec une boucle. Testez la fonction. (*Bonus : pensez aux exceptions si la requête n'est pas dans la liste.*)
4. Complétez le script pour demander deux mots à l'utilisateur et afficher leur valeur de co-occurrence en cherchant la case correspondante de la matrice.
5. Complétez le script pour demander un mot à l'utilisateur et afficher son vecteur de co-occurrences.
6. Créez une nouvelle matrice, binaire, où la case $matrix[n, m]$ prend la valeur 1 si le mot m apparaît dans le contexte de n, et 0 ailleurs. Sauvegardez-la.
7. Complétez le script pour demander un mot à l'utilisateur et afficher tous les mots qui appaaissent dans son contexte immédiat (aide : vous pouvez utiliser numpy.nonzero).
8. Complétez le script pour demander un mot à l'utilisateur et afficher tous les mots qui apparaissent dans son contexte immédiat.
9. Complétez le script pour lire deux mots en entrée standard et affiche tous leurs contextes partagés.
   - Affichez les contextes partagés les plus saillants (tip : np.argsort), où la saillance est calculée comme :
     - nb de co-occurrences avec mot 1 + nb de co-occurrences avec mot 2
     - nb de co-occurrences avec mot 1 x nb de co-occurrences avec mot 2

     Quelle différence entre les deux méthodes de calcul, du point de vue du classement ?

10. Nous allons maintenant travailler avec les vecteurs de mots. La norme l1 du vecteur est la somme de toutes les valueurs :

$$|x|_1 = \sum_{i=1}^{n} x_i \tag{1}$$

où n est le nombre des dimensions. Diviser les valeurs du vecteur de co-occurrences brutes par la norme l1 donne les valeurs de 'fréquence relative', qui peuvent être interprêtées comme une distribution de probabilité, parce qu'elles somment à 1.

La norme l2 du vecteur est sa longueur :

$$|x|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{2}$$

Il s'agit dans les deux cas des normes **du vecteur** ; pour obtenir le vecteur normalisé, il faut diviser les valeurs de chaque dimension par la norme.

Normalisez la matrice **par colonnes**, selon la norme l2. Utilisez np.linalg.norm.

11. Les mots seront représentés comme des vecteurs (ceux de notre matrice de co-occurrence). Nous cherchons les mots les plus similaires. Nous utiliserons la similarité cosinus entre deux vecteurs. Ele est compris entre -1 (vecteurs opposés) et 1 (similaires).

$$cos(x, y) = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} (x_i)^2}\sqrt{\sum_{i=1}^{N} (y_i)^2}} \tag{3}$$

Ecrivez une fonction qui prend deux mots en entrée, et retournela similarité cosinus entre leurs vecteurs. Testez-la avec des mots sur la liste topfreq.
Utilisez np.linalg.norm pour la norme des deux vecteurs.