Prof. Amal Zouaq – INF8460 (Automne 2020)
Chargés de laboratoire : Antoine Lefebvre-Brossard & Dylan Farvacque

École Polytechnique de Montréal
Département Génie Informatique et Génie Logiciel
INF8460 – Traitement automatique de la langue naturelle

Objectifs d'apprentissage

- Explorer les modèles d'espace vectoriel (*vector space models*) comme représentations distribuées de la sémantique des mots
- Implémenter la fréquence de cooccurrence et la PPMI
- Comprendre différentes mesures de distance entre vecteurs de mots
- Explorer l'intérêt de la réduction de dimensionnalité

Logiciels

Ce TP utilise la librairie NLTK, ScikitLearn et Jupyter notebook.

Le squelette du TP est fonctionnel sur Google Colab et peut être complété sur cette plateforme. Si vous n'utilisez pas Google Colab, il est conseillé d'installer Anaconda avec Python 3.7 ou plus.

Modalités de remise du TP

Vous devez compléter le squelette inf8460_tp2_A20.ipynb et le soumettre sous le nom matricule1_matricule2_matricule3_inf8460_tp2_A20.ipynb qui reprend les différentes questions, et implante les fonctionnalités requises avec des commentaires appropriés et comprend aussi les réponses aux questions lorsque demandé.

Critères d'évaluation

- L'exécution correcte du code
- Le professionnalisme du notebook
- La clarté des explications/commentaires qui l'accompagnent

Corpus et code

Les données doivent être téléchargées à partir de : IMDB large movie review dataset: http://ai.stanford.edu/~amaas/data/sentiment/

Le zip du TP contient :

- Le corpus qui contient des revues de films positives et négatives
- Un ensemble de données de test: un ensemble de parenté (relatedness) appelé The MEN Test Collection
- Le squelette du notebook qui doit être complété

Vous devez retourner le notebook complété.

Travail à faire

1. Pré-traitement (20 points)

a) (*10 points*) Le jeu de données est séparé en deux répertoires `train/`et `test`, chacun contenant eux-mêmes deux sous-répertoires `pos/` et `neg/` pour les revues positives et négatives. Un fichier `readme` décrit plus précisément les données. Commencez par lire ces données, en gardant séparées les données d'entraînement et de test. La fonction doit mettre les mots en minuscules, supprimer les stopwords (vous devez utiliser ceux de NLTK) et afficher le nombre total de phrases d'entrainement, le nombre total de phrases d'entrainement positives et négatives et le nombre total de phrases de test avec le nombre total de phrases de test positives et négatives ;

b) (10 *points*) Créez la fonction build_voc() qui extrait les unigrammes de l'ensemble d'entraînement et conserve ceux qui ont une fréquence d'occurrence d'au moins 5 et imprime le nombre de mots dans le vocabulaire. Sauvegardez-le dans un fichier vocab.txt (un mot par ligne).

2. Matrices de co-occurrence mot × mot M(w,w) (30 points)

a) (*5 points*) A partir des textes du corpus d'entrainement (neg/pos), vous devez construire une matrice de co-occurrence mot × mot M(w,w) qui contient les 5000 unigrammes les plus fréquents sous forme de **cadre panda**. Le contexte de co-occurrence est une fenêtre de +/-5 mots autour du mot cible. Le poids est la fréquence de co-occurrence simple. Sauvegardez votre matrice dans un fichier `tp2_mat5.csv` dans le répertoire vsm.

b) (*10 points*) Calculez maintenant une matrice de cooccurrence mais en ajustant les fréquences basées sur la proximité du mot cible par exemple en les multipliant par $1/d$ où d est la distance en jetons (mots) de la cible. Sauvegardez votre matrice dans un fichier `tp2_mat5_scaled.csv` dans le répertoire vsm.

c) (*10 points*) Vous devez créer une fonction pmi qui prend le cadre panda de la matrice M(w,w) et un paramètre boolean flag qui est à True lorsque l'on désire calculer PPMI et à False quand on veut calculer PMI. La fonction transforme la matrice en entrée en une matrice M'(w,w) avec les valeurs PMI ou PPMI selon la valeur du paramètre booléen. La fonction retourne le nouveau cadre panda correspondant.

d) (*5 points*) Calculez une matrice pmi et ppmi en vous basant sur les deux matrices que vous avez déjà créée en a) et b). Sauvegardez vos matrices dans un fichier tp2_mat5<_scaled>_{pmi|ppmi}.csv toujours dans le répertoire vsm. (votre nom de fichier doit contenir "_scaled" s'il est formé à partir Mww_scaled et "pmi" si le flag est false "ppmi" sinon)

3. Test de PMI/PPMI (20 points)

Pour le test des matrices de cooccurrences, nous allons comparer deux mesures de distance entre deux vecteurs, la distance euclidienne et la distance cosinus.

a) (*8 points*) Implémentez la fonction `voisins(mot, pd, distance)` qui prend un mot en entrée et une métrique de distance et qui retourne les n mots les plus similaires selon

la mesure. Pour un mot w, elle ordonne tous les mots du vocabulaire en fonction de leur distance de w en utilisant la métrique de distance distance (par défaut: cosine) sur le vsm pd. Les mesures de distance à tester sont : la distance euclidienne et la distance cosinus.

b) (*3 points*) En utilisant le cadre panda associé aux matrices Mww et Mww scaled, trouvez les 5 mots les plus similaires au mot « beautiful » et affichez-les, pour chacune des deux distances

c) (*3 points*) En utilisant les cadres panda associés aux matrices PMIs, trouvez les 5 mots les plus similaires au mot « beautiful » et affichez-les, pour chacune des deux distances.

d) (*3 points*) En utilisant les cadres panda associés aux matrices PPMIs, trouvez les 5 mots les plus similaires au mot « beautiful » et affichez-les, pour chacune des deux distances.

e) (*3 points*) Que constatez-vous entre la différence de performance de la distance euclidienne et la distance cosinus ? Que constatez-vous entre les différents types de matrices de cooccurrence ?

4. Réduction de dimensionnalité (20 points)

a) (*6 points*) Ecrivez une fonction lsa qui prend en entrée un cadre panda pd (qui contient votre matrice / vsm) et un paramètre K (qui indique le nombre de dimensions finales), et qui applique LSA avec ce paramètre k sur la matrice et retourne le vsm réduit sous forme de cadre panda.

b) (*2 points*) Exécutez lsa sur les cadres panda associés à vos matrices Mww et Mww_scaled avec une dimension k=100

c) (*2 points*) En utilisant les matrices de co-occurrence réduites avec LSA, trouvez les 5 mots les plus similaires au mot « beautiful » selon la distance cosinus et affichez-les.

d) (*2 points*) En utilisant les matrices PMIs et PPMIs réduites avec lsa, trouvez les 5 mots les plus similaires au mot « beautiful » selon la distance cosinus et affichez-les.

e) (*5 points*) En utilisant sklearn.decomposition.TruncatedSVD, créez les matrices réduites à partir des mêmes matrices que celles de la question précédente (la matrice pmi et la matrice pmi_scaled) Puis testez ces nouvelles matrices LSA pour trouver les 5 mots les plus similaires au mot «beautiful » Ici aussi, nous voulons aussi obtenir des matrices de dimension k=100.

f) (*3 points*) Commentez vos résultats

5. Evaluation (10 points)

Il est temps d'évaluer l'intérêt de nos modèles de vecteurs. Nous allons pour cela utiliser un ensemble de données de similarité de mots (relatedness) *The MEN dataset*, qui se trouve dans le répertoire MEN. L'ensemble de données contient une paire de mots avec un score de similarité attribué par des humains. En d'autres termes, une ligne (un exemple) de l'ensemble de données est de la forme : <mot_1> <mot_2> <score>.

Pour aligner les distances obtenues avec vos métriques, ce score est converti en nombre réel négatif par la fonction `read_test_dataset` que vous avez dans le squelette du TP.

La métrique d'évaluation est le coefficient de corrélation de Spearman ρ entre les scores humains et vos distances (voir https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient).

Nous allons maintenant évaluer les différents vsm obtenus sur l'ensemble de données: MEN_dataset.

On vous fournit la fonction `evaluate` qui calcule la corrélation de Spearman entre les scores de l'ensemble de données de test et celui du modèle vsm. La fonction retourne la corrélation de spearman ainsi que le nombre d'exemples qui ont été pris en compte.

a) (*5 points*) Testez chacun de vos modèles vsm (Matrice de base, matrice scalée, les PMIs et PPMIs, toutes les matrices LSA (de base, scalée, pmi, ppmi)) en appelant la fonction `evaluate` avec les deux mesures de distance (euclidienne et cosinus) et affichez vos résultats dans une seule table.

b) (5 *points*) Commentez vos résultats d'évaluation