Université Paul Sabatier
EIMAB3H1 - Analyse et exploitation de données

Enseignant : José G. Moreno

26/09/2019

TP 1. Similarité sémantique entre termes

Rappelez vous de l'équation Positive Pointwise Mutual Information (PPMI).

$$PPMI = \begin{cases} log\frac{p(x,y)}{p(x)*p(y)} & si \quad log\frac{p(x,y)}{p(x)*p(y)} > 0 \\ 0 & autrement \end{cases}$$

Elle permet de calculer des similarités entre mots. La premier étape consiste à calculer la fréquence entre mot ou la probabilité conjointe ***p(x,y)***. Ce calcul est fait dans un intervalle ou fenêtre, qui correspond à des partitions de l'information d'origine. L'utilisation d'une fenêtre glissante de taille fixe est très répandue. Dans ce cas, un nombre spécifique de mots sont considérés pour faire le calcule et puis la fenêtre se déplace sur la source. Considérez le suivant paragraphe:

```
A chaque ville, son explication. Les gares sur l'axe Atlantique
(Toulouse, Tours et Bordeaux) ont par exemple été touchées par les
travaux de la LGV Sud-Europe Atlantique, reliant Paris à Bordeaux en 2
h 4, inaugurée au début du mois de juillet. Toulouse, relié à Paris via
Bordeaux en TGV, a donc fait les frais de cette entreprise, débutée en
2012.
```

En savoir plus sur
http://www.lemonde.fr/les-decodeurs/article/2017/08/02/tgv-un-taux-de-retards-stable-en-quatre-ans-mais-inegal-sur-le-territoire-francais_5167883_4355770.html#JQKswFWRoJgyDQrY.99

Si la taille de la fenêtre est égale à 5 mots avec un déplacement d'un mot entre fenêtres, les 4 premières fenêtres seront :

```
[A chaque ville son explication]
[chaque ville son explication Les]
[ville son explication Les gares]
[son explication Les gares sur]
...
```

Si on considérer que le paragraphe a 60 mots, il aura 59 fenêtres c'est-à-dire le nombre des mots moins un. Pour le calcule de ***p(x,y)*** entre les mots Bordeaux et Toulouse, ***p('Bordeaux','Toulouse')***, avec une fenêtre égale à 5 mots et 59 fenêtres, le valeur de ***p('Bordeaux','Toulouse')*** sera égale à ***2/59***.

1. Écrivez un script en python qui permettra calculer la fréquence entre deux mots avec une fenêtre glissante de taille fixe.

2. Calculez les valeurs de PPMI entre mots pour tous les mots du paragraphe « A chaque ville, son explication... » .

3. Notez que vous avez construit une matrice de taille NxN (où N est la talle du vocabulaire). Cette matrice ne peut pas être facilement visualisable car tous les éléments sont de dimension N. Cependant, il existe de méthodes pour projeter les donnes dans un dimension visualisable comme 2D ou 3D. En utilisant la méthode SVD, projetez votre matrice dans un espace de deux dimensions. En utilisant SVD en 2D, visualisez les points pour les villes (Toulouse, Tours et Bordeaux) et pour les marques ferroviaires (LGV et TGV) . Voici un exemple d'utilisation de la méthode SVD (les valeurs de X sont des exemples, il faut que vous calculez X avec PPMI).

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

X = np.array([[-1, -1, -1, 0], [-2, -1, 0, 8], [-3, -2, 5, 1], [1, 1,
5, -1], [2, 1, 7, 1], [3, 2, -1, 0]])
pca = PCA(n_components=2)
Xp = pca.fit_transform(X)

red = [0,1,2]
bleu = [3,4,5]


plt.plot([x[0] for x in Xp[red]],[x[1] for x in Xp[red]], 'ro')
plt.plot([x[0] for x in Xp[bleu]],[x[1] for x in Xp[bleu]], 'bx')
plt.axis([-10, 10, -10, 10])
plt.savefig('foo.png')
```

4. Si vous faites le même calcule sur toute la Wikipédia, croyez-vous que les points vont se placer différemment ? Pourquoi ?

5. Calculez une autre matrice de similarité entre mots comme vous l'avez fait par PPMI mais en utilisant Wordnet. Wordnet est une ressource externe construite manuellement, donc plus riche et plus propre. Pour chaque mot, vous pouvez calculer sa similarité en analysant le chemin plus court entre synsets. Regardez les supports de cours si vous avez de doutes sur Wordnet et les concepts associés. Voici un exemple de comme utiliser Wordnet avec nltk pour calculer des similarités.

```
from nltk.corpus import wordnet

synsa = wordnet.synsets("monday")[0] #premier sens de monday
synsb = wordnet.synsets("friday")[0] #premier sens de friday
synsc = wordnet.synsets("dog")[0] #premier sens de dog

print("Wordnet similarity between monday and friday: ",
synsa.path_similarity(synsb))
print("Wordnet similarity between monday and dog: ",
synsa.path_similarity(synsc))
```

6. Vous pouvez aussi visualiser la similarité en 2D en utilisant SVD. Calculez la matrice de similarité entre tous les jours de la semaine (Monday, Tuesday, etc.) et les mois de l'année (January, February, etc). La matrice doit faire 19x19. Projetez cette matrice de similarité dans un espace de dimension 2 et

visualisez les 19 points comme vous l'avez fait précédemment. Notez vous des régularités ?

7. Une nouvelle technique pour représenter les mots apparue récemment est connue comme *word embeddings*. Elle est basée sur les réseaux de neurones. Regardez les supports de cours pour plus de détails sur les *word embeddings* et les concepts associés. Une fois calculée cette représentation, elle permet d'avoir un vecteur par mot (comme nous avons fait en 2 et 3). Téléchargez le fichier glove.6B.50d.txt.zip sur moodle. Lisez le fichier et calculez le vecteur C = 'Paris'- 'France' + 'Italy', puis triez uniquement les capitales européennes (dans la liste ci-dessous) en ordre descendant en utilisant la similarité cosinus. Notez vous des régularités ? Essayez avec des autres pays au lieu de l'Italie (utilisez les noms en anglais uniquement).

```
Tirana
Vienna
Minsk
Brussels
Sarajevo
Sofia
Zagreb
Nicosia
Prague
Copenhagen
Tallinn
Helsinki
Paris
Berlin
Gibraltar
Athens
Budapest
Reykjavik
Dublin
Douglas
Rome
Riga
Vaduz
Vilnius
Luxembourg
Skopje
Valletta
Monaco
Podgorica
Amsterdam
Oslo
Warsaw
Lisbon
Bucharest
Moscow
Belgrade
Bratislava
Ljubljana
Madrid
Stockholm
Bern
Kiev
London
```