Enoncé

Rendez-vous sur https://classroom.github.com/a/ 80uuQMs, téléchargez votre projet, lisez attentivement les sections **Distance de document** et **Explications**, et réalisez le travail demandé dans la section **Objectifs**. Un fonctionnement correct ne peut pas vous assurer une note parfaite. La qualité de votre code, performance et lisibilité sont également pris en compte ! Pensez également à commenter votre code si vous le jugez complexe.

Pour finir, n'oubliez pas de *pusher* votre code sur votre *repository* git – vous avez jusqu'au 22 décembre 2019 à 23h59.

La distance de document

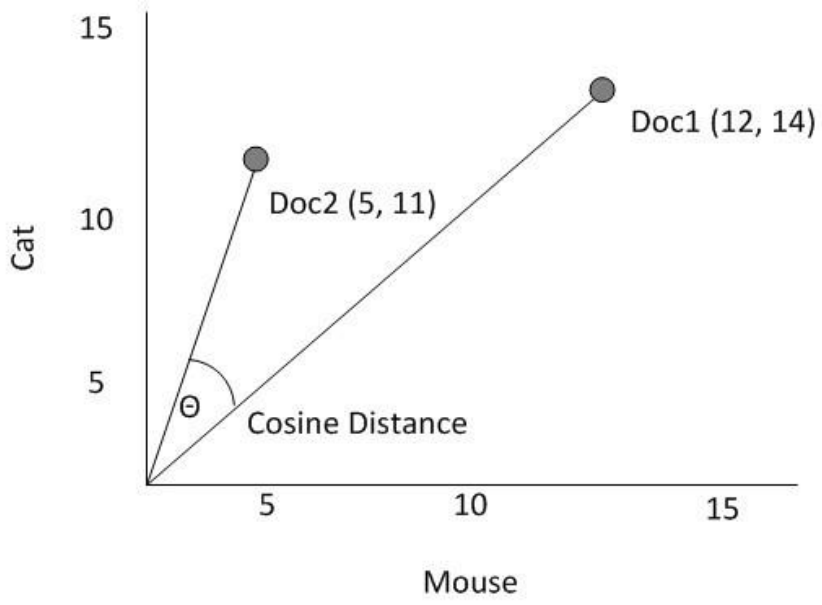
Les similitudes entre documents sont mesurées en fonction du chevauchement du contenu entre les documents. En raison du grand nombre de documents texte dans notre vie, il est nécessaire de traiter automatiquement ces documents pour les applications d'extraction d'informations, de mise en cluster de similarité et de recherche. C'est en effet ce qu'une recherche Google ou autre utilise comme algorithme pour retourner les meilleures réponses.

Il existe un grand nombre d'algorithmes complexes pour résoudre ce problème. L'un de ces algorithmes est une similarité en cosinus - une mesure de similarité basée sur un vecteur. La distance cosinusoïdale de deux documents est définie par l'angle entre leurs vecteurs caractéristiques qui sont, dans notre cas, des vecteurs fréquence-mots. La distribution de la fréquence des mots d'un document est une correspondance entre les mots et leur fréquence dans le texte.

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

où « · » désigne le produit scalaire des deux vecteurs de fréquence A et B, et $\|A\|$ dénote la longueur (ou la norme) d'un vecteur.

Figure 1 – Visualisation de la distance de cosinus



Vous pouvez visualiser, grâce à la Figure 1, comment un document est représenté sous forme vectorielle. Par exemple, le document **Doc1** contient 12 fois le mot « **Mouse** » et 14 fois le mot « **Cat** ». Nous pouvons compter combien de fois ces mots apparaissent dans plusieurs documents et garder seulement les n documents plus proches en utilisant la distance de cosinus (θ).

Explications mathématiques

Utilisons la phrase suivante comme exemple :

« To be or not to be »

Pour avoir un vecteur de fréquences des mots, vous devez parcourir la phrase et stocker chaque mot et son nombre d'occurrences dans un dictionnaire.

$$C = \{be: 2, not: 1, or: 1, to: 2\}$$

Sans les mots, cela devient un vecteur à 4 dimensions :

$$\{2, 1, 1, 2\}$$

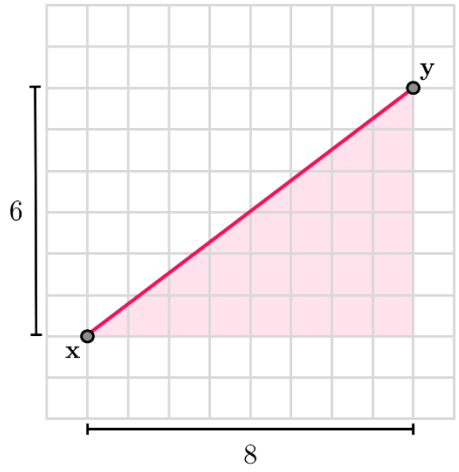
La norme euclidienne

La norme euclidienne du vecteur de fréquence est définie par

$$\|C\| = \sqrt{x_1^2 + x_2^2 \cdots + x_n^2} = \sqrt{2^2 + 1^2 + 1^2 + 2^2} = 3.16228$$

où x_k représente les fréquences pour chaque mot du texte.

Figure 2 - Pythagore



Vous pouvez voir cette formule comme une utilisation du théorème de Pythagore.

Le produit scalaire

Imaginons un autre vecteur D

$$D = \{3, 4, 1, 0\}$$

Un produit scalaire est définie comme suit :

$$C \cdot D = C_1 \cdot D_1 + C_2 \cdot D_2 + \cdots C_4 \cdot D_4 = 2 \cdot 3 + 1 \cdot 4 + 1 \cdot 1 + 2 \cdot 0 = 11$$

La distance de cosinus

Pour calculer la distance entre deux documents D_1et D_2, nous pouvons utiliser la mesure de similarité de cosinus :

$$dist(D_1, D_2) = \arccos\left(\frac{freq(D_1) \cdot freq(D_2)}{\|freq(D_1)\| \cdot \|freq(D_2)\|}\right)$$

Notez que la distance entre deux documents identiques est 0 et pour deux documents distincts leur distance est de $\frac{\pi}{2} = 1.57$ ...

Objectifs

- Acquérir de l'expérience en travaillant avec les interfaces List, Set et Map
- Acquérir de l'expérience en travaillant avec des problèmes plus réalistes et ouverts, avec des spécifications à respecter

Spécifications

- Ne considérez que les mots de longueur supérieure à 0 ! Aussi, un mot est une séquence de lettres [a…zA…Z] pouvant inclure des chiffres [0…9]. Tous les délimiteurs sont jetés et ne sont pas conservés dans le mot.
- Le programme doit être *case insensitive*, c'est-à-dire, il doit considérer les majuscules comme des minuscules. HEG = heg.
- L'implémentation de la méthode du produit scalaire doit respecter une performance en **O(n)**.

Toutes les méthodes que vous pouvez implémenter sont commentées dans les classes **Distance**, **Vector**, **Corpus**, **Document** et **Loader**.

Dans la classe Main, vous avez un code qui devrait pouvoir être exécuté sans être modifié. Utilisez-le pour vérifier votre implémentation finale. Pensez à faire vos propres tests également.