Espaces vectoriels et recherche d'information

Matthieu Constant
Université Paris-Est Marne-la-Vallée, LIGM

Plan

Plan

Rappels sur les vecteurs en géométrie 2D

Un espace à deux dimensions

- Une origine
- Deux axes perpendiculaires avec des unités
- Un point défini par deux coordonnées sur ces axes

Un vecteur dans un espace à deux dimensions

- Définition informelle : une direction, un sens et une distance
- Définition formelle : deux coordonnées sur les axes

Calculs sur les vecteurs

Coordonnées

Soit un vecteur u défini par ses coordonnées (u_x, u_y).

Norme

- Longueur du vecteur
- Formule pour le vecteur u :

$$|u| = \sqrt{u_x^2 + u_y^2}$$

Calculs sur les vecteurs (suite)

Produit scalaire

- Produit de deux vecteurs qui renvoie un nombre (scalaire)
- Formule :

$$u.v = u_x v_x + u_y v_y$$

Cosinus

- La mesure dépend de l'angle formé entre deux vecteurs
- Elle est comprise entre -1 et 1 pour des angles entre 0 et 2π (entre 0 et 1 pour des angles entre 0 et $\frac{\pi}{2}$)
- Formule :

$$cos(u,v) = \frac{u.v}{|u||v|}$$

Généralisation à n dimensions

Vecteur

- Un vecteur est placé dans un espace à n dimensions (n axes)
- Un vecteur u est défini par un n-uplet (u_1,u_2,...,u_n)

Norme

$$|u| = \sqrt{\sum_{i=1}^{n} u_i^2}$$

Généralisation à n dimensions (suite)

Produit scalaire

$$u.v = \sum_{i=1}^{n} u_i v_i$$

Cosinus

$$cos(u,v) = \frac{u.v}{|u||v|}$$

Plan

Modèle des espaces vectoriels

Représentation simple des textes

- Un texte est un sac de mots (il n'y a plus d'ordre !)
- On associe à chaque mot un poids (nombre réel), mesurant son "importance" dans le texte

Application à la géométrie vectorielle

- Un texte est un vecteur dans un espace de grande dimension
- Chaque coordonnée correspond au degré d'importance d'un mot donné dans le texte

Pondération des mots

Pondération naïve

- Poids binaire (1 si terme présent dans le document, 0 sinon)
- Fréquence du mot dans le document

Pondération plus intelligente

- On utilise des fonctions correctrices de la fréquence
- On prend aussi en compte la distribution du mot dans la collection où est plongée le texte

Pondération binaire - exemple

Collection de documents

d1 we were anchored off an island in the bahamas

d2 the couple traveled from island to island throughout the bahamas

Représentation vectorielle

	an	anchored	bahamas	couple	from	in	island	off	the	throughout	to	traveled	we	were
d1	1	1	1	0	0	1	1	1	1	0	0	0	1	1
d2	0	0	1	1	1	0	1	0	1	1	1	1	0	0

Nombre d'occurrences - exemple

Collection de documents

d1 we were anchored off an island in the bahamas

d2 the couple traveled from island to island throughout the bahamas

Représentation vectorielle

	an	anchored	bahamas	couple	from	in	island	off	the	throughout	to	traveled	we	were
d1	1	1	1	0	0	1	1	1	1	0	0	0	1	1
d2	0	0	1	1	1	0	2	0	2	1	1	1	0	0

Mesure TF.IDF

Principe

- On suppose que le texte traité est plongé dans une collection de documents
- Un mot pertinent d'un document apparaîtra plusieurs fois dans ce document
- Les mots non-pertinents sont distribués de manière homogène dans la collection

Exemple (statistiques sur des articles du New York Times)

terme	cf	df
insurance	10440	3997
try	10422	8760

tiré de Manning et Schütze, 1999

Mesure TF.IDF (suite)

Fréquence des termes ou mots (TF)

- $tf_{i,j}$: fréquence du mot i dans le document j de la collection
- $tf_{i,j}$: nombre d'occurrences du mot i dans le document j normalisé par le nombre total de mots dans le document j

Fréquence inverse de document (IDF)

- idf_i mesure l'importance d'un terme dans l'ensemble de la collection
- $idf_i = log \frac{m}{D(i)}$
  avec m le nombre total de documents de la collection
  et $D(i)$ le nombre de documents de collection où le mot i apparaît

Formule TF.IDF

- Le poids $d_{i,j}$ d'un mot i dans un document j est :

$$d_{i,j} = tf_{i,j}.idf_i$$

Représentation des documents en Python

Utilisation de dictionnaires

- clés : mots (dont le poids est non nul)
- valeurs : poids des mots

Classe *Text* et ses méthodes

- def getName(self)
- def getWordTokens(self)
- def getWeight(self,w) #w : un mot
- def cosine(self,t) #t : un Text

Exemple de code Python

Calcul de la norme

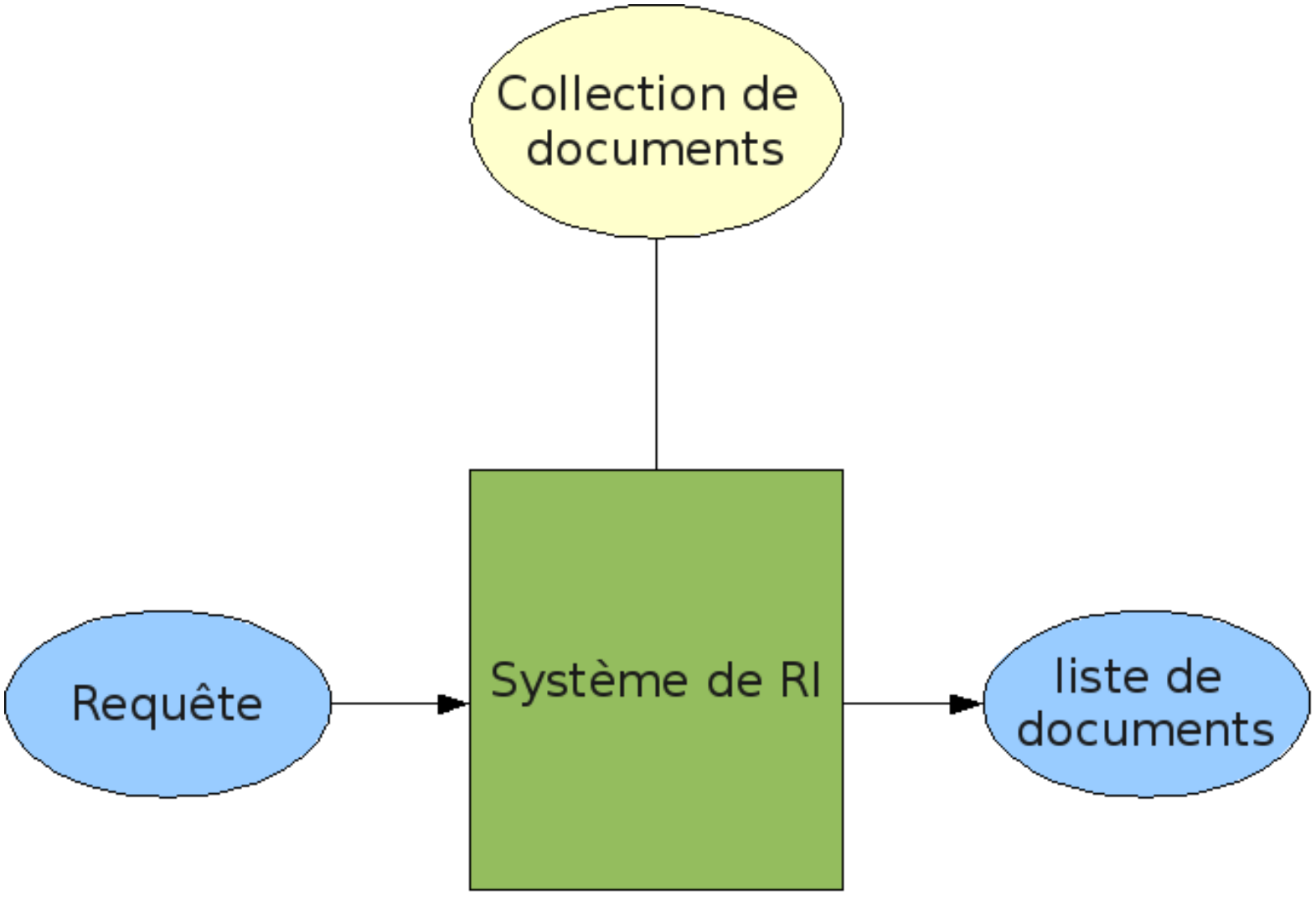
```
import math


class Text:

  def norm(self):
    n = 0
    for w in self.getWordTokens():
      n += self.getWeight(w)*self.getWeight(w)
    return math.sqrt(n)
```

Plan

Recherche d'informations



Recherche d'informations (suite)

Principe

- L'utilisateur entre une requête décrivant une information qu'il cherche
- Le système renvoie une liste de documents pertinents par rapport à la requête

Deux approches

- Recherche exacte (ex. systèmes booléens)
- Recherche floue (ex. modèles à espaces vectoriels)

Recherche d'informations et modèles à espaces vectoriels

Représentation

Requêtes (des séquences de mots) et documents de la collection sont représentés sous la forme de vecteurs

Métaphore entre `proximité spatiale` et `proximité sémantique`

- Les documents les plus pertinents sont ceux qui ont les vecteurs les plus proches de celui de la requête
- Les documents les plus pertinents contiennent des mots similaires à ceux de la requête

=> La mesure de la pertinence d'une requête par rapport à un document consiste à comparer leurs vecteurs respectifs : ex. cosinus de leur angle

Requête - exemple

- Requête q : "island couple"

	an	anchored	bahamas	couple	from	in	island	off	the	throughout	to	traveled	we	were
q	0	0	0	1	0	0	1	0	0	0	0	0	0	0
d1	1	1	1	0	0	1	1	1	1	0	0	0	1	1
d2	0	0	1	1	1	0	2	0	2	1	1	1	0	0

Requête - exemple

- Requête q : "island couple"

	an	anchored	bahamas	couple	from	in	island	off	the	throughout	to	traveled	we	were
q	0	0	0	1	0	0	1	0	0	0	0	0	0	0
d1	1	1	1	0	0	1	1	1	1	0	0	0	1	1
d2	0	0	1	1	1	0	2	0	2	1	1	1	0	0

- Pertinence des documents de la collection :

  cos(q,d1)= 1/(1.41 * 3) = 0.2

  cos(q,d2)= 3/(1.41 * 3.74) = 0.6

Exemple de code Python

Moteur de recherche

```
# collection.lst est un fichier
# contenant la liste des fichiers de la collection

c = TextCollection('collection.lst')
while True:
  query = raw_input('Enter a query:')
  print 'RESULT:'
  print c.search(query,10)
```

Exemple de code Python

Recherche des meilleurs documents

```
class TextCollection:

  def search(self,query,N):
    q = Text(text=query)
    scores = {}
    for t in self.getTexts():
      score[t.getName()] = t.cosine(q)
    return sortScores(scores,N)
```

Evaluation des systèmes

Qualité d'un système de RI

Dans quelle mesure les documents pertinents sont retournés avant les documents non-pertinents ?

Mesures traditionnelles

- **précision** : proportion de documents pertinents dans la liste retournée
- **rappel** : proportion de documents pertinents dans la collection qui sont dans la liste retournée (difficile à évaluer !)

Exemple

Evaluation	Ranking 1	Ranking 2	Ranking 3
	d0 : v	d9 : x	d5 : x
	d1 : v	d8 : x	d0 : v
	d2 : v	d7 : x	d1 : v
	d3 : v	d6 : x	d9 : x
	d4 : v	d5 : x	d8 : x
	d5 : x	d0 : v	d2 : v
	d6 : x	d1 : v	d4 : v
	d7 : x	d2 : v	d3 : v
	d8 : x	d3 : v	d6 : x
	d9 : x	d4 : v	d7 : x
Précision :	**0.5**	**0.5**	**0.5**

tiré de (Manning et Shütze, 1999)

Précision et cutoff

Précision traditionnelle pas suffisante

- car ne tient pas compte du rang du document
- ex. Ranking 1 est clairement meilleur que Ranking 2 !

Solution alternative : le cutoff

- On regarde la précision de segments initiaux plus petits
- Ex. on peut calculer la précision au rang 5

Exemple

Evaluation	Ranking 1	Ranking 2	Ranking 3
	d0 : v	d9 : x	d5 : x
	d1 : v	d8 : x	d0 : v
	d2 : v	d7 : x	d1 : v
	d3 : v	d6 : x	d9 : x
	d4 : v	d5 : x	d8 : x
	d5 : x	d0 : v	d2 : v
	d6 : x	d1 : v	d4 : v
	d7 : x	d2 : v	d3 : v
	d8 : x	d3 : v	d6 : x
	d9 : x	d4 : v	d7 : x
Précision à 10 :	0.5	0.5	0.5
Précision à 5 :	**1**	**0**	**0.4**

tiré de (Manning et Shütze, 1999)

Précision moyenne

Principe

- Précision calculée pour chaque point de la liste où l'on trouve un document pertinent
- Puis on fait la moyenne

Dans l' exemple

- points pertinents : d1, d2, d3, d4 et d5
- précisions dans le Ranking 3 :
  1/2 (d1), 2/3 (d2), 3/6 (d3), 4/7 (d5), 5/8 (d4)
- moyenne : 0.5726

Exemple

Evaluation	Ranking 1	Ranking 2	Ranking 3
	d0 : v 1/1	d9 : x	d5 : x
	d1 : v 2/2	d8 : x	d0 : v 1/2
	d2 : v 3/3	d7 : x	d1 : v 2/3
	d3 : v 4/4	d6 : x	d9 : x
	d4 : v 5/5	d5 : x	d8 : x
	d5 : x	d0 : v 1/6	d2 : v 3/6
	d6 : x	d1 : v 2/7	d4 : v 4/7
	d7 : x	d2 : v 3/8	d3 : v 5/8
	d8 : x	d3 : v 4/9	d6 : x
	d9 : x	d4 : v 5/10	d7 : x
Précision à 10 :	0.5	0.5	0.5
Précision à 5 :	1	0	0.4
Précision moyenne :	**1**	**0.35**	**0.57**

tiré de (Manning et Shütze, 1999)

Quelques techniques complémentaires

Filtrage

- Parcours de l'ensemble des documents de la collection coûteux
- Filtrage des documents non-pertinents par un index

Réduction de l'espace

- Racinisation des mots (ex. algorithme de Porter avec nltk)
- Filtrage des mots grammaticaux (ex. *le, la, un, à, de*, ...) (?)

Quelques techniques complémentaires (suite)

Extension de la requête

- Ajout de synonymes à l'aide de ressources linguistiques
- Précision des requêtes avec de nouveaux mots calculés à partir de statistiques de cooccurrence)

Algorithme `pseudo-feedback`

Quelques techniques complémentaires (suite)

Autres critères de recherche

- PageRank (Google)
- Positionnement des mots de la requête dans le document (ex. titre)
- Distance entre les mots de la requête dans le document
- ...