Vectorisation de Documents Texte

Pr T.RACHAD
UM6P – 2024/2025

Objectifs:

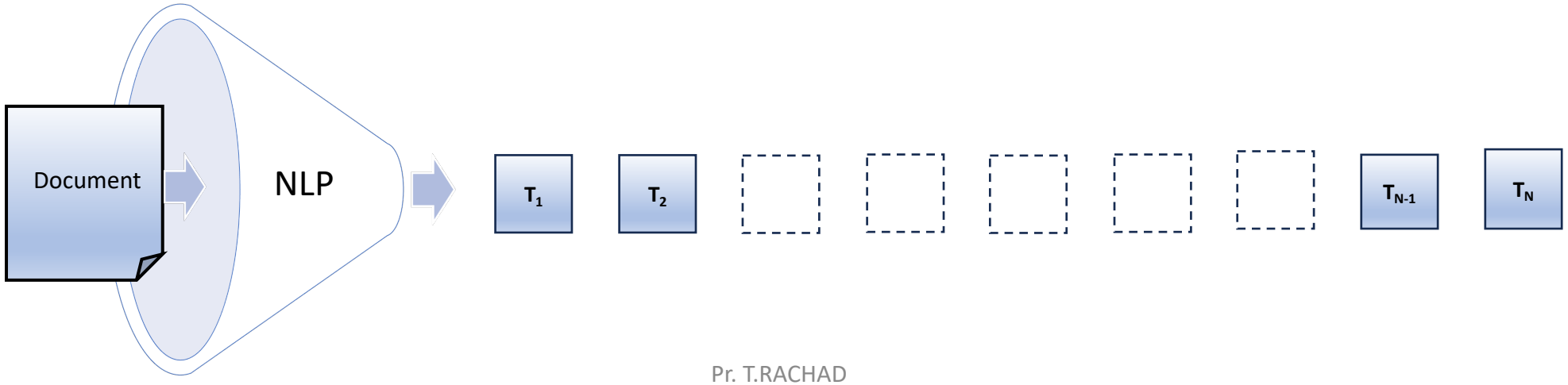
- Découvrir les différentes techniques permettant d'avoir une représentation vectorielle de documents texte.
- Découvrir les différentes façons pour calculer les similarités entre des documents texte.

Model Représentatif de documents texte

Model Représentatif d'un Document (1/5)

- Les opérations de prétraitement (Segmentation, Nettoyage, lemmatisation, PoS-Tag, Analyse syntaxique, Analyse sémantique…) visent principalement à obtenir une représentation plus structurée d'un document texte.

- Chaque document du corpus peut produire un nombre diffèrent de termes clés



Model Représentatif d'un Document (2/5)

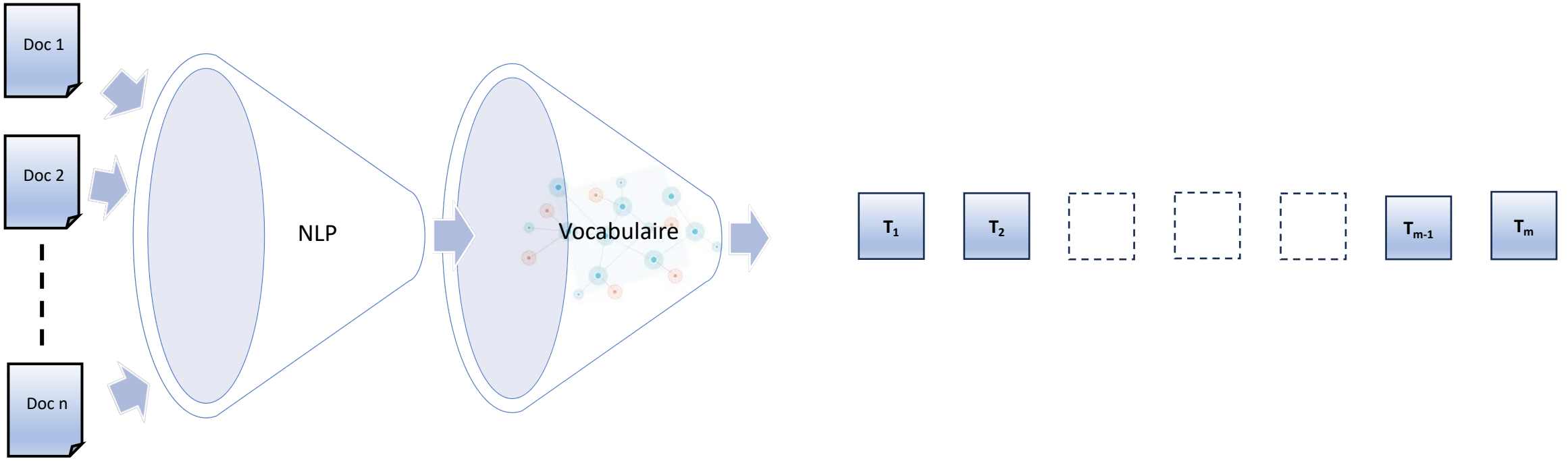
- Le nombre important (des centaines, voire de milliers) d'éléments-clés décrivant les documents d'un corpus affecte considérablement les performances d'un système de text mining.

- Il faut avoir un équilibre entre la taille et le niveau sémantique des éléments-clés choisis pour décrire un corpus de documents texte.

- Les éléments-clés à sélectionner doivent faciliter la découverte de patterns dans une collection de documents.

- Cela peut être réalisé en se basant sur des vocabulaires contrôlés, des dictionnaires, des thésaurus, des ontologies, etc.

Model Représentatif d'un Document (3/5)



Model Représentatif d'un Document (4/5)

- L'ensemble des éléments clés obtenus représentent ce qu'on appelle le model représentatif qui fournit une représentation simplifiée des connaissances existantes dans tous les documents d'un corpus.

- Formellement, pour un corpus $D = \{d_i, i=1, 2,...,n\}$ de documents texte, chaque document d_i sera considéré comme un point dans un espace numérique $T= \{t_i, i=1, 2,...,m\}$.

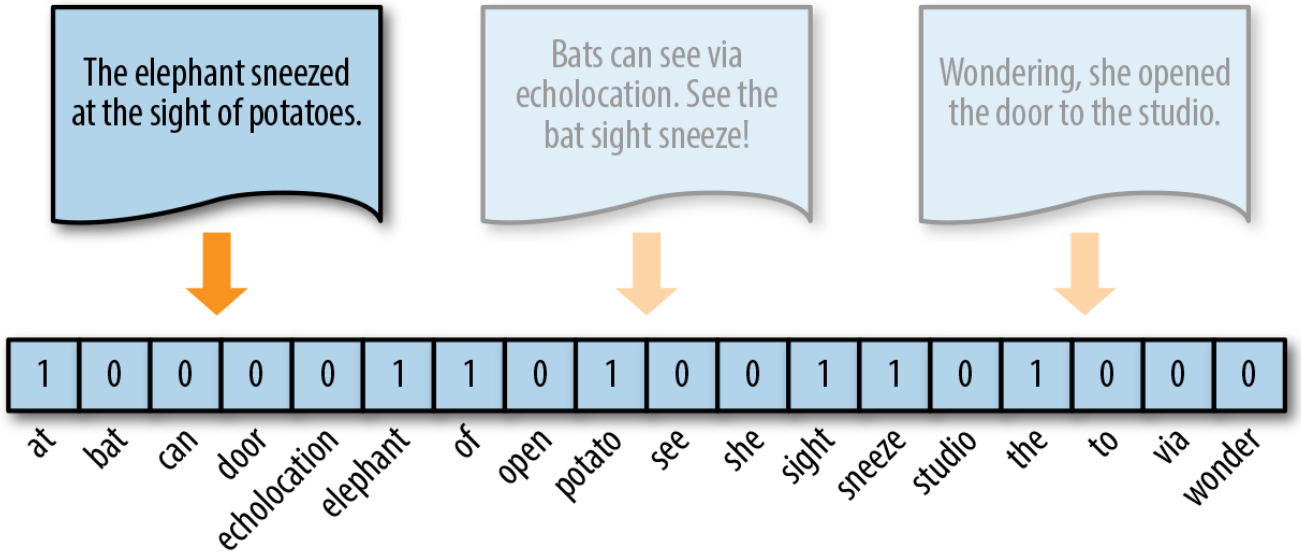
- Cette représentation (numérique) permettra d'automatiser le traitement de documents texte avec des méthodes statiques supervisées et non supervisées.

Model Représentatif d'un Document (5/5)

- Il existe deux classes de techniques pour récupérer la représentation vectorielle numérique d'un document texte:
  - Techniques classiques qui utilisent un vocabulaire constitué d'un nombre bien déterminé de mots : **One Hot Vector, Bag of Words, Tf-idf**.
  - Techniques non supervisées permettant d'induire les représentations vectorielles des termes à partir d'une grande quantité de données textuelles: **Word2Vect, Doc2Vect, Glove, Bert…**

One Hot Vector (OHV)

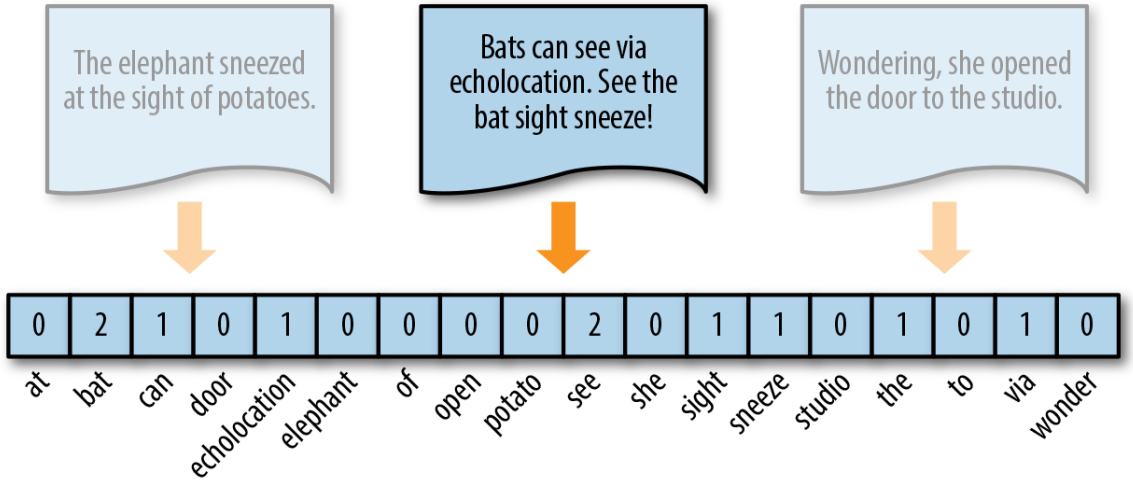
- OHV est un modèle représentatif naïf qui nous informe sur la présence ou non des termes d'un vocabulaire dans un document texte.



Bag-of-Words (BOW)

- OHV fourni un modèle représentatif très simple qui néglige l'importance ou le poids des termes du vocabulaire dans les documents du corpus.

- Le modèle BOW est une amélioration du OHV qui calcule pour chaque document le nombre d'occurrences de chaque terme du vocabulaire.



Bag of n-grams

- Dans les représentations vectorielles à base de BOW l'ordre des termes dans les documents est ignoré.

- La fréquence d'apparition d'une séquence de termes (séquence de n termes) peut informer également sur les connaissances cachées dans les documents et peut être exploitée dans une représentation vectorielle des documents.

- BOW est un cas particulier du Bag of n-grams dans lequel n=1.

TFIDF (1/3)

- BOW considère la fréquence d'apparition des termes du vocabulaire dans chaque document et néglige l'importance du terme par rapport au corpus tout en entier.

- TF-IDF (term-frequency - inverse document frequency) permet de calculer la fréquence de présence d'un terme dans un document en considérant également son occurrence dans tout le corpus.

- Cela permettra de diminuer l'importance des termes les plus fréquents dans les documents texte et qui sont souvent non significatifs.

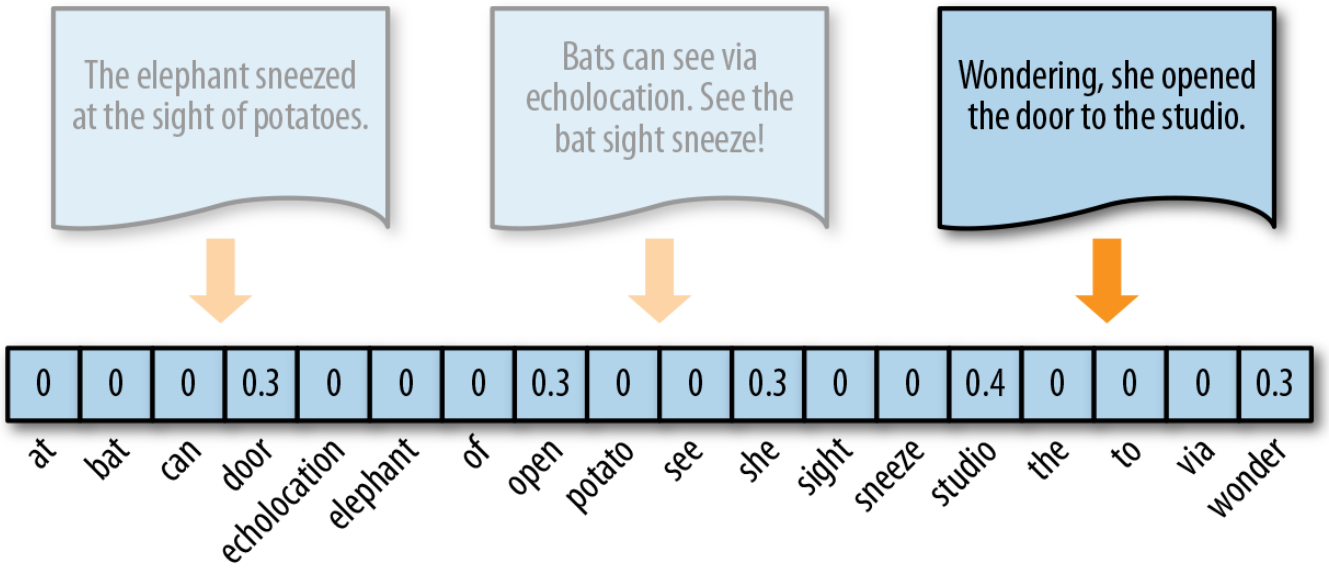
TFIDF (2/3)

- Le calcul du score TFIDF des termes du vocabulaire dans un document donné repose sur les fréquences (TF) des différents termes dans le document et sur leurs fréquence inverse (IDF).

- Le produit des fréquences TF et IDF d'un terme est appelé le poids TFIDF de ce terme. plus le score TFIDF est élevé, plus le terme est rare dans le document.

- TFIDF(t, d, D) = TF(t, d)*IDF(t , D)
  - TF(t, d)= log(1+f(t,d))
  - IDF(t, D)= log(N/ f(t,D)) avec N est le nombre de documents dans le corpus

TFIDF (3/3)


The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
0 0 0 0.3 0 0 0 0.3 0 0 0.3 0 0 0.4 0 0 0 0.3
at bat can door echolocation elephant of open potato see she sight sneeze studio the to via wonder

(OHV, BOW, TFIDF) Inconvénients

- La taille importante du vocabulaire pour un corpus très large.
  - Nécessité de réduire la la taille du vocabulaire(LSA, LDA…).

- Toutes les techniques Ignorent la dimension sémantique.

- Elles ignorent l'aspect structurel (enchainement des termes).

Techniques non Supervisées

- L'objectif des techniques de vectorisation non supervisées est d'avoir une représentation vectorielle (numérique) qui prendra en considération les éventuelles similarités entres les mots selon les contextes de leur utilisation.
  - Les mots similaires auront des représentations vectorielles très proches.

- Les représentations vectorielles sont obtenues en se basant sur les cooccurrences des termes dans les documents du corpus.

- Deux approches:
  - Approches par décomposition vectorielle (SVD) à base de matrices de cooccurrences.
  - Approches Itératives à base des réseaux de neurones (word2vec, GloVe…)

SVD (Singular value Decomposition) (1/4)

1. Nous parcourons le corpus et accumulons les cooccurrences de mots sous forme de matrice X.
2. Ensuite, nous effectuons une décomposition de valeurs singulières sur X pour obtenir une décomposition UDV^T.
3. Nous utilisons ensuite les lignes de U comme représentation vectorielle des mots du vocabulaire.

	w_1	w_2	…	w_m
d_1				
d_2				
…				
d_n				

X =Corpus vectorisé

	E_1	E_2	…	E_k
w_1				
w_2				
…				
w_m				

U=Les mots représentés dans un nouvel espace dimension k<n

SVD (Singular value Decomposition) (2/4)

- Plusieurs façons pour créer la matrice X:
  - X est une matrice des occurrences des termes du vocabulaire dans les documents du corpus. X_{ij} représente le nombre des occurrences du mot w_i dans le document d_j.
  - X est une matrice carrée et symétrique des cooccurrences (à base d'une fenêtre) de termes. X_{ij} représente le nombre des cooccurrences du terme w_i avec le terme w_j dans les documents du corpus selon une fenêtre de taille prédéfinie.
  - Pondérer le nombre des cooccurrences en fonction de la distance entre les termes du document.

SVD (Singular value Decomposition) (3/4)

- Soit $C_{(mxn)}$ la transposé de la matrice rectangulaire $X_{(nxm)}$ représentant le corpus.
  - Les lignes de C donnent les représentations vectorielles des mots du vocabulaire dans l'espace des documents.

- On réalise la décomposition de la matrice sous la forme $C=UDV^T$ avec:
  - *U* est une matrice orthogonale ($UU^T=I$) de dimensions mxk. C'est le vocabulaire représenté dans le nouvel espace de dimension k.
  - *D* est une matrice représentant les valeurs singulières de la décomposition. C'est une matrice diagonale de dimension kxk représentant les poids des différents concepts latents.
  - *V* est une matrice orthogonale ($V^TV=I$) de dimensions nxk.

La matrice C

	d_1	d_2	…	d_n
w_1				
w_2				
w_m				

Corpus vectorisé

La matrice U

	E_1	E_2	…	E_k
w_1	π_{11}	π_{12}		π_{1k}
w_2	π_{21}	π_{22}		π_{2k}
…				
w_m	π_{n1}	π_{n2}		π_{nk}

La matrice V

	E_1	E_2	…	E_k
d_1	$P(d_1/E_1)$	$P(d_1/E_2)$		$P(d_1/E_k)$
d_2	$P(d_2/E_1)$	$P(d_2/E_2)$		$P(d_2/E_k)$
…				
d_n	$P(d_n/E_1)$	$P(d_n/E_2)$		$P(d_n/E_k)$

La matrice D

E_1	E_2	…	E_k
V_1	V_2		V_k

SVD (Singular value Decomposition) (4/4)

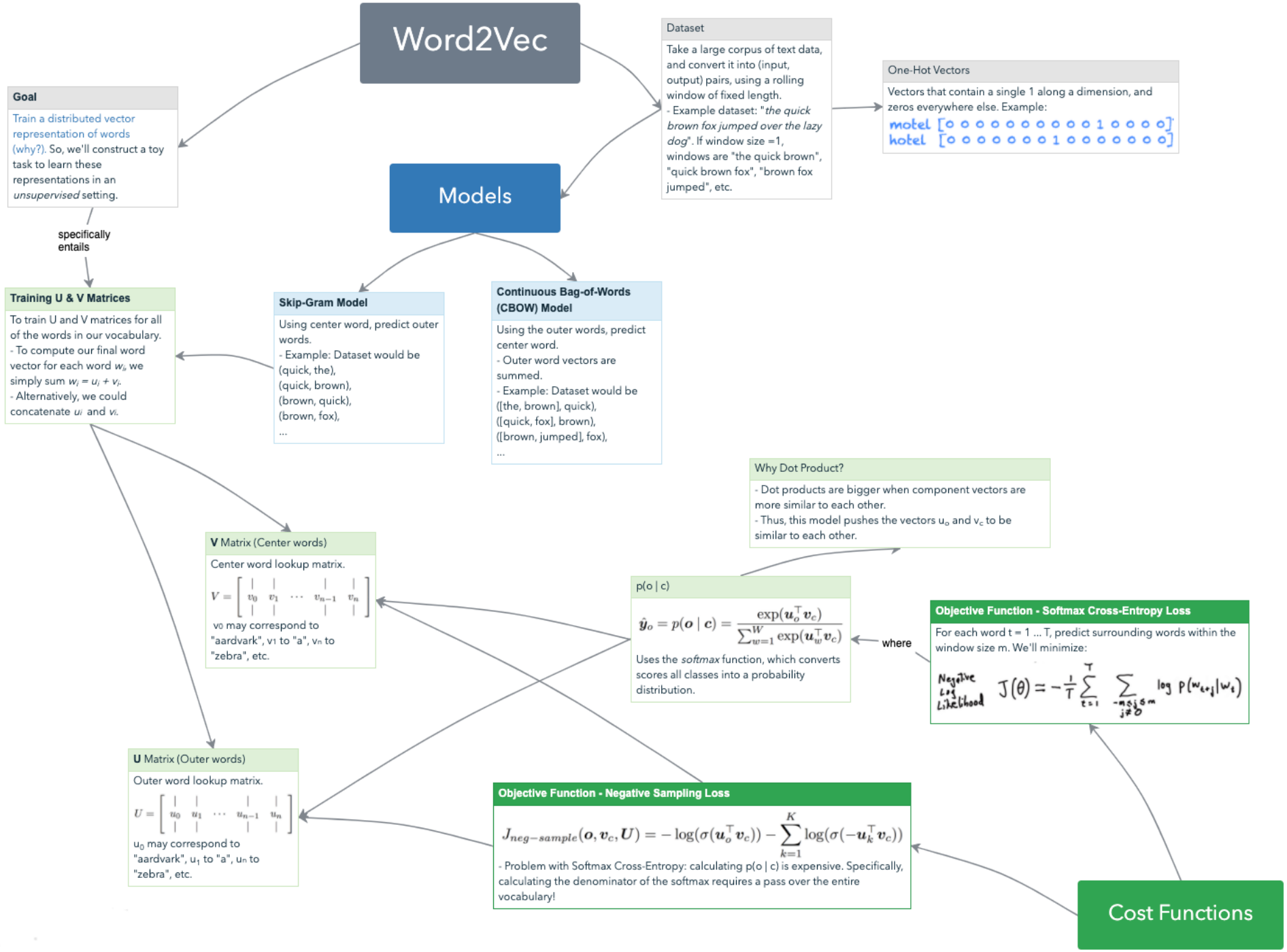
- Inconvénients:
  - La matrice X est extrêmement creuse car la plupart des mots ne coexistent pas.
  - La matrice possède des dimensions très grandes.
  - Les méthodes basées sur SVD ne s'adaptent pas bien aux grandes matrices et il est difficile d'incorporer de nouveaux mots ou documents.
  - Le coût de calcul pour une matrice $m \times n$ est $O(mn^2)$

Word2vec (1/3)

- Au lieu de traiter à la fois toutes les valeurs de la matrice des cooccurrences comme en SVD, Word2vec connu également sous le nom **Neural Probabilistic Language Models** permet d'apprendre les représentations vectorielles des différents termes en suivant une approche itérative.

- A chaque nouvelle itération, nous récupérons la probabilité d'un mot selon son contexte actuel (ou l'inverse).

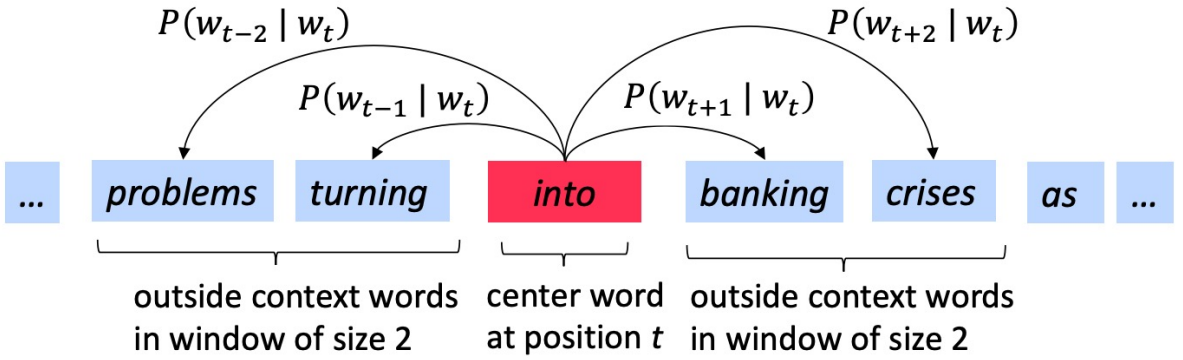
- Bénéfices:
  - Plus rapide
  - Possibilité de traiter de nouveaux mots/documents


Word2Vec
Goal
Train a distributed vector representation of words (why?). So, we'll construct a toy task to learn these representations in an *unsupervised* setting.
specifically entails
Dataset
Take a large corpus of text data, and convert it into (input, output) pairs, using a rolling window of fixed length.
- Example dataset: "*the quick brown fox jumped over the lazy dog*". If window size =1, windows are "the quick brown", "quick brown fox", "brown fox jumped", etc.
One-Hot Vectors
Vectors that contain a single 1 along a dimension, and zeros everywhere else. Example:
motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
Models
Training U & V Matrices
To train U and V matrices for all of the words in our vocabulary.
- To compute our final word vector for each word w_i, we simply sum $w_i = u_i + v_i$.
- Alternatively, we could concatenate u_i and v_i.
Skip-Gram Model
Using center word, predict outer words.
- Example: Dataset would be
(quick, the),
(quick, brown),
(brown, quick),
(brown, fox),
...
Continuous Bag-of-Words (CBOW) Model
Using the outer words, predict center word.
- Outer word vectors are summed.
- Example: Dataset would be
([the, brown], quick),
([quick, fox], brown),
([brown, jumped], fox),
...
Why Dot Product?
- Dot products are bigger when component vectors are more similar to each other.
- Thus, this model pushes the vectors u_o and v_c to be similar to each other.
V Matrix (Center words)
Center word lookup matrix.
$V = \begin{bmatrix} | & | & & | & | \\ v_0 & v_1 & \cdots & v_{n-1} & v_n \\ | & | & & | & | \end{bmatrix}$
v_0 may correspond to "aardvark", v_1 to "a", v_n to "zebra", etc.
p(o | c)
$\hat{y}_o = p(o \mid c) = \frac{\exp(u_o^\top v_c)}{\sum_{w=1}^{W} \exp(u_w^\top v_c)}$
Uses the *softmax* function, which converts scores all classes into a probability distribution.
where
Objective Function - Softmax Cross-Entropy Loss
For each word t = 1 ... T, predict surrounding words within the window size m. We'll minimize:
Negative Log Likelihood $J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j} \mid w_t)$
U Matrix (Outer words)
Outer word lookup matrix.
$U = \begin{bmatrix} | & | & & | & | \\ u_0 & u_1 & \cdots & u_{n-1} & u_n \\ | & | & & | & | \end{bmatrix}$
u_0 may correspond to "aardvark", u_1 to "a", u_n to "zebra", etc.
Objective Function - Negative Sampling Loss
$J_{neg-sample}(o, v_c, U) = -\log(\sigma(u_o^\top v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^\top v_c))$
- Problem with Softmax Cross-Entropy: calculating p(o | c) is expensive. Specifically, calculating the denominator of the softmax requires a pass over the entire vocabulary!
Cost Functions

Word2vec (2/3)

- Le processus général du word2vec consiste à parcourir les différents documents d_i en extrayant à chaque **itération t** une expression texte d'une taille fixe **k** (**fenêtre de taille k**). Cette expression possède un **mot central w_t** et un **contexte de sortie O** (qui représente les autres mots de l'expression).

- Les représentations vectorielles des différents mots sont obtenues en se basant sur les probabilités $P(w_{t+j}/w_t)$.



Word2vec (3/3)

- Deux modèles d'implémentation à base des réseaux de neurones:
  - **Continuous bag-of-words (CBOW):** Prédire un mot central à partir du contexte.
  - **Skip-gram**: Prédire la distribution (probabilité) des mots de contexte à partir d'un mot central.

- Deux méthodes d'apprentissage:
  - Classificateur Softmax.
  - Negative sampling.

Word2vec (CBOW) (1/5)

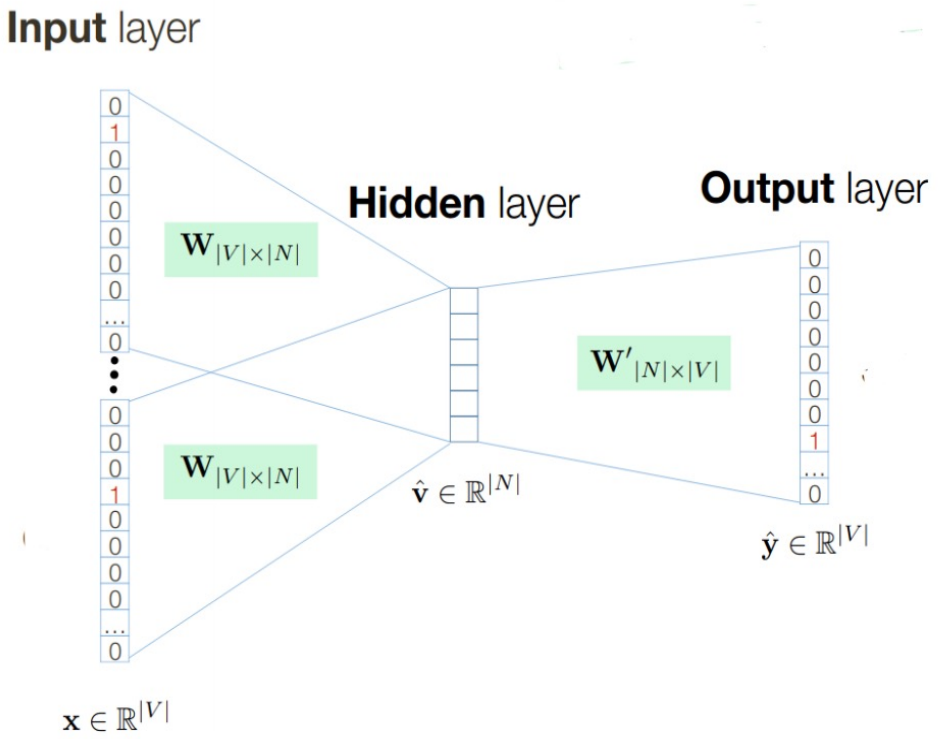
- Dans cette approche on traite une séquence de mots comme un contexte qui va nous permettre de prédire un mot central.

- Pour chaque mot w_i nous voulons découvrir(prédire) deux vecteurs:
  - U_i est un vecteur représentant le mot dans son contexte lorsque le mot est central.
  - V_i est un vecteur représentant le contexte d'un mot lorsque le mot est dans le contexte.

- Paramètres du réseau de neurones :
  - **Entrée**: Une phrase (le contexte) représentée par une matrice ***O*** qui correspond aux vecteurs de ses différents mots (one-hot vectors).
  - **Sortie**: y (one hot vector) un vecteur représentant le mot central.
  - Les vecteurs U et V (inconnus)

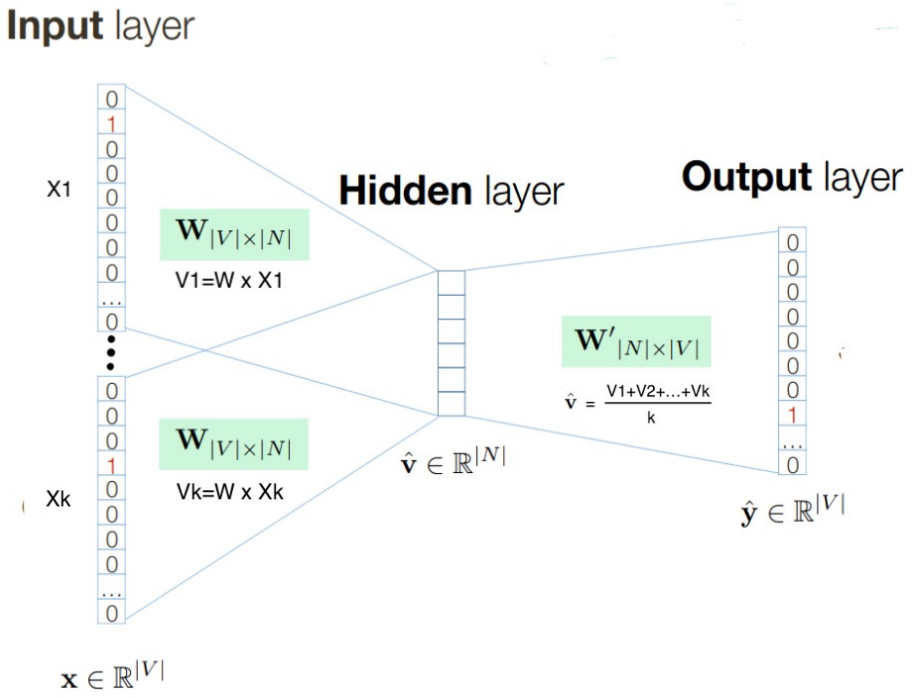
Word2vec (CBOW) (2/5)

- **La couche d'entrée**: C'est une représentation vectorielle des mots du contexte O={X_{t-k} …,X_{t-2}, X_{t-1}, X_{t+1}, X_{t+2},…,X_{t+k}} tel que chaque mot X_i (vecteur ligne) est une représentation vectorielle (one hot vector) dans l'espace du vocabulaire V de dimension n.
- **La couche de sortie**: C'est une représentation (approximative) vectorielle du mot central y dans l'espace du vocabulaire V de dimension n.
- **La couche cachée**: Elle reçoit en entrée la matrice des poids $W_{(nxm)}$(les vecteurs V_i) et renvoi en sortie la matrice des poids $W'_{(mxn)}$ (les vecteurs U_i)
  - Les poids W et W' seront découverts durant l'étape d'apprentissage. Ils seront initialisés par des valeurs aléatoires au début de l'opération d'apprentissage.
  - W et W' sont les mêmes pour tout le vocabulaire.



Word2vec (CBOW) (3/5)

- Au début Les vecteurs X_i sont multipliés par la matrice des poids W pour obtenir les différents vecteurs v_i.
- Le vecteur en sortie de la couche cachée sera obtenu en appliquant une moyenne sur l ’ensemble des vecteur v_i.



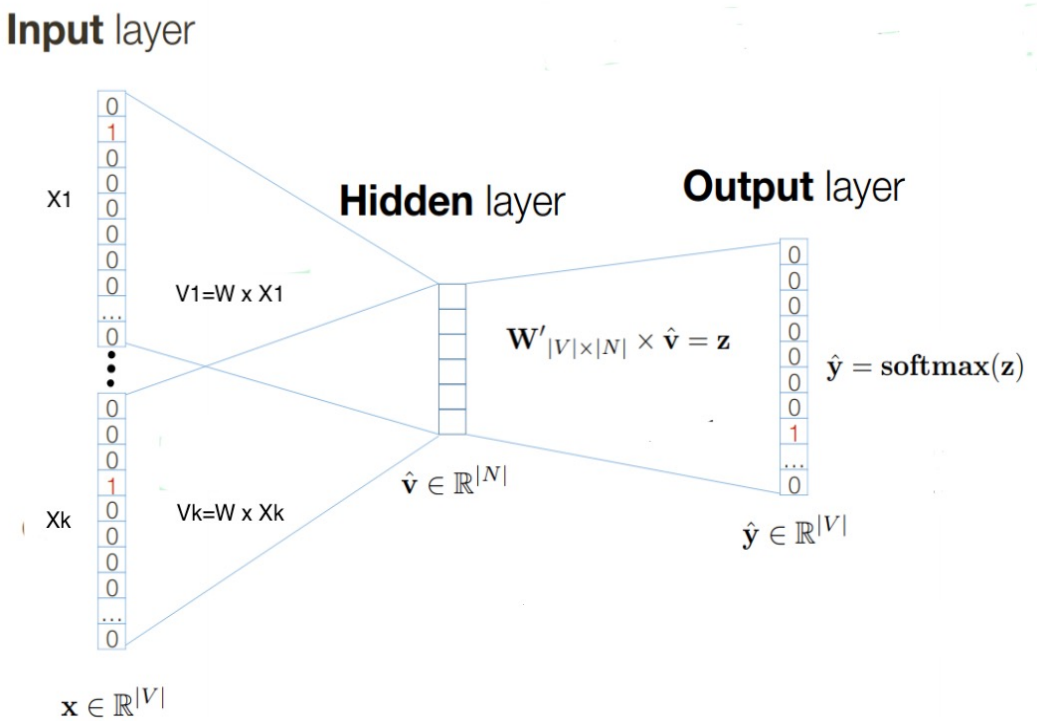
Word2vec (CBOW) (4/5)

- Le vecteur caché est par la suite multiplié par la matrice des poids W' pour obtenir le vecteur Z de sortie.
- La sortie finale est générée en se basant sur une normalisation softmax du vecteur z. tel que, pour C un mot central et O un mot de contexte:

Pour rendre tous positif

Le produit scalaire compare la similitude de o et c
Produit scalaire plus grand = probabilité plus grande

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$



Word2vec (CBOW) (5/5)

- La perte en matière de données est calculée par suite comme étant la différence entre y et $\hat{y}$: $|y - \hat{y}|$

- Par la suite un processus d'optimisation (à base de SGD: Stochastic Gradient Descent) est exécuté pour minimiser les pertes et mettre à jours les matrices des poids.

- Finalement la matrice V est W et la matrice U est W'.

SVD vs Word2vec

SVD

- Basée sur les cooccurrences (count based). La matrice possède des dimensions très grandes.
- Le coût de calcul pour une matrice m × n est O (mn^2)
- Les méthodes basées sur SVD ne s'adaptent pas bien aux grandes matrices et il est difficile d'incorporer de nouveaux mots ou documents.
- La matrice X est extrêmement creuse car la plupart des mots ne coexistent pas.
- Utilisation efficace des statistiques.
- Bénéfique pour capturer les similarités entre les mots

Word2vec

- Basée sur des probabilités
- Vitesse du traitement augment extrêmement en fonction de la taille du corpus .
- Méthode adaptative. De nouveaux mots/documents peuvent être ajoutés aisément au modèle.
- Utilisation inefficace des statistiques
- Peut être utilisée pour récupérer des relations plus complexes que les similarités???

Autres Techniques (Exposées)

- GloVe
- BERT
- ELMo
- n-gram embeddings
- Averaging word embeddings
- Sent2Vec
- FastText

Operations Vectorielles de Base

Notion de Similarité (1/2)

- Dans plusieurs applications du Text Mining on aura besoin de calculer la similarité ou la distance entre des documents texte:
  - Classification
  - Clustering
  - Substitution de documents
  - …

- La similarité Sim(d1,d2) entre deux documents d1 et d2 correspond au degré de **ressemblance** entre les deux documents.
  - Sim(d1,d2) =0 signifie que d1 et d2 sont complètement différents
  - Sim(d1,d2)=1 signifie que d1 et d2 sont totalement identiques

- La distance dist(d1,d2) entre deux documents d1 et d2 correspond au degré de **dissimilarité** entre les deux documents avec dist(d1,d2)= 1- Sim(d1,d2).

Notion de Similarité (2/2)

- Les représentations vectorielles de documents texte permettront de calculer les similarités entre les documents texte en exploitant une panoplie de mesures.
  - Distance euclidienne.
  - Similarité Cosinus.
  - indice de Jaccard.
  - …

Distance Euclidienne (1/2)

- La distance euclidienne est une mesure très utilisée en Text Mining pour comparer des expressions texte.

$$d(d_1, d_2) = \sqrt{\sum_{k=1}^{n} (d_{1,k} - d_{2,k})^2}$$

Avec n est la taille du vocabulaire

- NB: La distance euclidienne n'est pas bornée (pas de valeur max).

Distance Euclidienne (2/2)

	connaissance	données	étude	extraction	fouille	text	mining
d1	0	2	1	2	2	1	2
d2	1	1	0	1	0	3	1
d3	0	2	1	1	1	1	1

- d(d1,d2)=3.45
- d(d1,d3)=1.73
- d(d2,d3)=2.82

Similarité Cosinus (1/2)

- La similarité cosinus est également très exploitée en Text Mining.

$$\text{Sim}(d_1, d_2) = \frac{d1.d2}{\|d1\|.\|d2\|} = x = \frac{\sum_{k=1}^{n} d_{1'k} * d_{2'k}}{\sqrt{\sum_{k=1}^{n} {d_{1'k}}^2} * \sqrt{\sum_{k=1}^{n} {d_{2'k}}^2}}$$

- Elle est toujours comprise entre 0 et 1.

- La distance entre d1 et d2 sera: $\text{d}(d_1, d_2) = 1 - \text{Sim}(d_1, d_2)$

Similarité Cosinus (2/2)

	connaissance	données	étude	extraction	fouille	text	mining
d1	0	2	1	2	2	1	2
d2	1	1	0	1	0	3	1
d3	0	2	1	1	1	1	1

- Sim(d1,d2)=0.59
- Sim(d1,d3)=0.94
- Sim(d2,d3)=0.64

Indice de Jaccard (1/2)

- L'indice de Jaccard est également très exploité en Text Mining. Il se base uniquement sur les cooccurrences entre les termes.

$$\text{Sim}(d_1, d_2) = \frac{d1 \cap d2}{d1 \cup d2}$$

- Il est toujours compris entre 0 et 1.

- La distance entre d1 et d1 sera. $\text{d}(d_1, d_2)$=1-$\text{Sim}(d_1, d_2)$

Indice de Jaccard (2/2)

	connaissance	données	étude	extraction	fouille	text	mining
d1	0	1	1	1	1	1	1
d2	1	1	0	1	0	1	1
d3	0	1	1	1	1	1	1

- Sim(d1,d2)=0.57
- Sim(d1,d3)=1.00
- Sim(d2,d3)=0.57

Similarité Sémantique (1/4)

- La similarité sémantique entre les documents texte permet de comparer les documents en considérant tous les termes les composants au lieu de ne considérer que les termes identiques syntaxiquement.

- La similarité sémantique entre deux documents peut être calculée de deux façons différentes:

  - Approche lexicale

  - Approche statistique (Word Embedding)

Similarité Sémantique (2/4)

- L'approche lexicale se base sur l'utilisation d'une base de données lexicale telle que WordNet et qui peut être exploitée dans l'exploration de la sémantique d'un corpus.

- Dans la base de données lexicale WordNet les mots anglais sont déjà rattachés les uns aux autres via des relations lexicales qui définissent les liens sémantiques possibles entres ces mots

- Plusieurs algorithmes sont proposés pour mesurer numériquement les similarités sémantiques entre les mots: ***Leacock-Chodorow***, ***Wu-Palmer***, ***Resnik, Jiang-Conrath***, ***Lin***, etc.

Similarité Sémantique (3/4)

- Les mots contenus dans la base de données lexicale WordNet peuvent avoir plusieurs sens selon le contexte de leur utilisation.

- **Lesk** est un algorithme de disambiguation qui permet de déterminé le sens le plus adapté au contexte d'un mot "l'ensemble de ses voisins"

```
function Simplified_Lesk(word, sentence)
  best-sense←most frequent sense for word
  max-overlap←0
  context←set of words in sentence
  for each sense in senses of word do
    signature←set of words in the gloss_examples of sense
      overlap←Compute_overlap(signature, context)
      if overlap > max-overlap then
                max-overlap←overlap
                best-sense←sense
  end
  return(best-sense){ returns best sense of word}
```

Similarité Sémantique (4/4)

- Pour mesurer la similarité sémantique entre deux documents il faut commencer par déterminer les sens de leurs différents mots.
  - d1={S_1, S_2,…, S_n}
  - d2={S'_1, S'_2,…, S'_m}

- Une méthode naïve pour calculer la similarité sémantique entre d1 et d2 en calculant les similarités entre tous les mots de d1 et de d2 deux à deux.

$$\text{Sim}(d_1, d_2) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} Sim(S_i, S'_j)}{n*m}$$

- D'autre méthode: distance de Hausdorff, indice de Jaccard…

Similarité Sémantique : Inconvénients

- Besoin de réaliser des traitements additionnels pour récupérer le sens exact d'un mot selon son contexte
- Plusieurs nouveaux termes ne figurent pas dans la base de données WordNet.
- WordNet est utile juste pour l'anglais. Pour les autres langues, elle est peu efficace.
- WordNet manque de précision dans le calcul des similarités sémantiques entre les mots.
- ➔ Le Word Embedding est prometteur.