Plongements de mots statiques Wordembeddings

Traitement Automatique des Langues
Master IAAA
Aix Marseille Université

Outline

Outline

Représentations

- Objet mathématique permettant de représenter un objet du monde concret ou abstrait
- Pourquoi en a-t-on besoin ?
  - Pour servir d'entrée à un modèle mathématique
  - Pour analyser, comparer, visualiser et parfois comprendre des objets complexes

Représentations de mots

- Quelles caractéristiques doit posséder une bonne représentation de mots ?
  - mathématiques :
    - numérique
    - de dimension raisonnable
  - linguistiques :
    - permet de calculer une distance entre mots qui ait du sens :

      $$d(vélo, bicyclette) < d(vélo, cornichon)$$

    - prise en compte de la polysémie :

      $$rep(vol_1) \neq rep(vol_2)$$

- La représentation orthographique des mots n'est donc pas un très bon candidat !

Vocabulaire

- Etant donné un texte T, on distingue :
  - le nombre de mots différents (quelquefois appelés *types*) que comporte T.
  - les occurrences d'un mot : le nombre de fois qu'un mot donné apparaît dans T.
- Le **vocabulaire** est un ensemble de types.
  - Le vocabulaire peut être fourni indépendamment de T
  - Souvent il est extrait de T
- Un vocabulaire V contient $|V|$ mots distincts
- V peut être très grand, souvent $10k \leq |V| \leq 500k$

Vocabulaire

- On attribue en général, de manière arbitraire un entier à tout mot du vocabulaire, que l'on appelle **index** du mot.

```
vélo        1267
...
cornichon   1568
...
bicyclette 4534
```

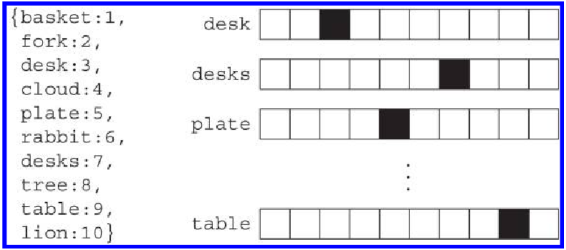
- L'index est une représentation possible pour un mot.
- Est-ce un bon candidat pour une représentation ?

Outline

Vecteurs *one-hot*

- Appelé aussi *encodage 1 parmi n*
- Tout mot m d'index i est représenté par un vecteur binaire $\mathbf{v}$ dont la dimension est $|V|$
- $\mathbf{v}$ possède un 1 en position i et des 0 partout ailleurs :

$$\mathbf{v}_i = 1 \textit{ et } \forall j \in \{1 \ldots |V|\} - \{i\} \; \mathbf{v}_j = 0$$



Source : Source : Pilehvar & Camacho-Collados (2020)

Sacs de mots

- On peut combiner des vecteurs one-hot à l'aide de fonctions :
  - **Ou logique** pour représenter un ensemble de mots
    $x_i = 1$ si le mot d'indice i est présent dans l'ensemble
  - **Somme** : nombre d'occurrences d'un mot
    $x_i = c$ si le mot d'indice i possède c occurrences
  - **Moyenne** : pas très pertinent pour des vecteurs one-hot, utile pour les représentations denses
- La position des mots n'est pas représentée !
- Représentation possible de textes pour des tâches simples de TAL.

Bilan

- Quelles caractéristiques doit posséder une bonne représentation de mots ?
  - mathématiques :
    - numérique
    - de dimension raisonnable
  - linguistiques :
    - permet de calculer une distance entre mots qui ait du sens :

$$d(vélo, bicyclette) < d(vélo, cornichon)$$

    - prise en compte de la polysémie :

$$rep(vol_1) \neq rep(vol_2)$$

Bilan

- Quelles caractéristiques doit posséder une bonne représentation de mots ?
  - mathématiques :
    - numérique
    - de dimension raisonnable
  - linguistiques :
    - permet de calculer une distance entre mots qui ait du sens :

$$d(vélo, bicyclette) < d(vélo, cornichon)$$

    - prise en compte de la polysémie :

$$rep(vol_1) \neq rep(vol_2)$$

- Pas terrible !

Outline

L’hypothèse distributionnelle

- Deux mots ayant tendance à apparaître dans le même contexte ont tendance à avoir le même sens (Harris 1954)
- “You shall know a word by the company it keeps" (Firth 1957)

L'hypothèse distributionnelle : exemple

Qu'est-ce qu'un *winyby* ?

L’hypothèse distributionnelle : exemple

Qu’est-ce qu’un *winyby* ?

- *C’était le meilleur winyby de la région*

L'hypothèse distributionnelle : exemple

Qu'est-ce qu'un *winyby* ?

- *C'était le meilleur winyby de la région*
- *J'ai apporté du winyby pour le déjeuner*

L'hypothèse distributionnelle : exemple

Qu'est-ce qu'un *winyby* ?

- *C'était le meilleur winyby de la région*
- *J'ai apporté du winyby pour le déjeuner*
- *Un verre de winyby, s'il vous plaît*

L'hypothèse distributionnelle : exemple

Qu'est-ce qu'un *winyby* ?

- *C'était le meilleur winyby de la région*
- *J'ai apporté du winyby pour le déjeuner*
- *Un verre de winyby, s'il vous plaît*
- *J'ai mal à la tête, j'ai bu trop de winyby hier soir*

L'hypothèse distributionnelle : exemple

Qu'est-ce qu'un *winyby* ?

- *C'était le meilleur winyby de la région*
- *J'ai apporté du winyby pour le déjeuner*
- *Un verre de winyby, s'il vous plaît*
- *J'ai mal à la tête, j'ai bu trop de winyby hier soir*
- *Cette distillerie de winyby a été fondée au 14ème siècle*

Le modèle de la fenêtre glissante

- Etant donné une grande quantité de texte brut, on compte combien de fois les mots apparaissent dans un contexte donné.
- Qu'est ce que le contexte d'un mot ?
- Les L mots qui précèdent et les L mots qui suivent.
- Modèle de la fenêtre glissante.
  - Un segment de $2 \times L + 1$ mots
  - Le mot du milieu est appelé mot **cible**
  - Les autres mots sont appelés mots du **contexte**
  - Le contexte n'est plus vu comme une unité, il est décomposé en mots
- Le mot cible à l'instant t sera mot du contexte à l'instant $t + 1$
- Quelle doit être la valeur de L ?
  - L petit : relations syntagmatiques (déterminant, nom)
  - L grand : relations paradigmatiques (synonymes, antonymes ...)

La fenêtre glissante : exemple

Source: `https://openclassrooms.com/en/courses/6532301-introduction-to-natural-language-processing/6980971-compare-embedding-models`

Matrice d'occurrence

- On compte le nombre de fois que le mot m_j fait partie du contexte du mot cible m_i
- Ce nombre constitue l'entrée $\mathbf{M}_{i,j}$ de la matrice $\mathbf{M}$

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Jurafsky & Martin (chap. 6)

- La ligne $\mathbf{m}_i$ de $\mathbf{M}$ représente le mot m_i
- Comment comparer deux mots ?

Similarité cosinus

Etant donné deux vecteurs $\mathbf{m}_i$ et $\mathbf{m}_j$, on définit le cosinus des deux vecteurs de la façon suivante :

$$\cos(\mathbf{m}_i, \mathbf{m}_j) = \frac{\mathbf{m}_i \cdot \mathbf{m}_j}{||\mathbf{m}_i|| \times ||\mathbf{m}_j||} = \frac{\mathbf{m}_i}{||\mathbf{m}_i||} \cdot \frac{\mathbf{m}_j}{||\mathbf{m}_j||}$$

- Numerateur : produit scalaires de $\mathbf{m}_i$ et $\mathbf{m}_j$:

$$\mathbf{m}_i \cdot \mathbf{m}_j = \sum_{k=1}^{|V|} M_{i,k} \times M_{j,k}$$

- Dénominateur : Produit des normes de $\mathbf{m}_i$ et $\mathbf{m}_j$:

$$||\mathbf{m}_i|| = \sqrt{\mathbf{m}_i \cdot \mathbf{m}_i} = \sqrt{\sum_{k=1}^{|V|} M_{i,k} \times M_{i,k}}$$

Produit scalaire

- Le produit scalaire de deux vecteurs **a** et **b** est d'autant plus élevé que les composantes correspondantes des deux vecteurs ont des valeurs élevées.

$\mathbf{a}$	$\mathbf{b}$	$\mathbf{a} \cdot \mathbf{b}$
$(2, 4, 10)$	$(2, 4, 10)$	120
$(2, 4, 10)$	$(4, 10, 2)$	68
$(1, 2, 5)$	$(1, 2, 5)$	30
$(0, 2, 0)$	$(1, 0, 5)$	0

- La valeur du produit scalaire de deux vecteurs est très liée à la norme de ces derniers.
- Le cosinus de deux vecteurs peut être vu comme le produit scalaire des vecteurs normalisés
- On note $\mathbf{a}' = \frac{\mathbf{a}}{||\mathbf{a}||}$ et $\mathbf{b}' = \frac{\mathbf{b}}{||\mathbf{b}||}$

$\mathbf{a}'$	$\mathbf{b}'$	$\mathbf{a}' \cdot \mathbf{b}'$
$(0.18, 0.36, 0.91)$	$(0.18, 0.36, 0.91)$	1
$(0.18, 0.36, 0.91)$	$(0.36, 0.91, 0.18)$	0.55
$(0.18, 0.36, 0.91)$	$(0.18, 0.36, 0.91)$	1
$(0, 1, 0)$	$(0.16, 0, 0.84)$	0

Exemple

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry}, \text{information}) = \frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .018$$

$$\cos(\text{digital}, \text{information}) = \frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

Jurafsky & Martin (chap. 6)

Bilan

- Quelles caractéristiques doit posséder une bonne représentation de mots ?
  - mathématiques :
    - numérique
    - de dimension raisonnable
  - linguistiques :
    - permet de calculer une distance entre mots qui ait du sens :

      $$d(\textit{vélo}, \textit{bicyclette}) < d(\textit{vélo}, \textit{cornichon})$$

    - prise en compte de la polysémie :

      $$rep(vol_1) \neq rep(vol_2)$$

Bilan

- Quelles caractéristiques doit posséder une bonne représentation de mots ?
  - mathématiques :
    - numérique
    - de dimension raisonnable
  - linguistiques :
    - permet de calculer une distance entre mots qui ait du sens :

      $$d(\textit{vélo}, \textit{bicyclette}) < d(\textit{vélo}, \textit{cornichon})$$

    - prise en compte de la polysémie :

      $$rep(vol_1) \neq rep(vol_2)$$

- C'est mieux !

Outline

Plongements de mots

- Les méthodes précédentes avaient l'inconvénient de représenter les mots par des vecteurs de **dimensions importantes** et **très creux** (beaucoup de zéros).
- Il est possible d'apprendre des représentations **denses** de dimensions réduites et arbitraires, appelées **plongements** ou **embeddings**
- On définit une tâche de **prédiction** binaire : est-ce que le mot *m* est susceptible d'apparaître dans le contexte du mot cible, par exemple *cornichon* ?
- On n'a pas besoin de données annotées, du texte brut suffit, il nous permet de savoir quels mots apparaissent ou n'apparaissent pas dans le contexte de *cornichon* (auto-supervision).
- Les paramètres du modèle de prédiction constituent les représentations des mots.

Plongements de mots

- De très bons résutlats sur toutes les tâches de TAL.
- Modèle opaque : on ne peut pas attribuer un sens aux dimensions des vecteurs appris.
- Plusieurs algorithmes fondés sur cette idée
- Ici l'algorithme Word2veq, plus précisément Skip-gram with Negative Sampling (SGNS)
- Les étapes :
  - Construire des exemples positifs et négatifs à l'aide d'un corpus brut
    - (mot cible, mot du contexte) $\rightarrow$ exemple positif
    - (mot cible, mot hors du contexte) $\rightarrow$ exemple négatif
  - Entraîner un classifieur à distinguer ces deux cas
  - Utiliser les paramètres du modèle comme plongements

Des plongements aux probabilités

- Etant donné un mot m et un mot c et leurs plongements $\mathbf{m}$ et $\mathbf{c}$
- Comment estimer la probabilité que le mot c apparaisse dans le contexte du mot m :

$$P(+|m, c)$$

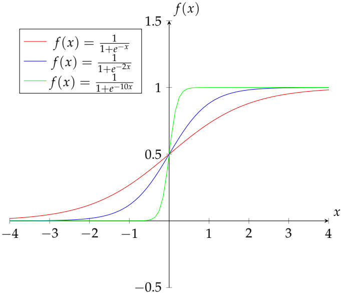
- Intuition :
  - Le plongement d'un mot est une représentation de son contexte.
  - Deux mots ont une probabilité élevée d'apparaître dans le même contexte si leurs plongements sont proches.

Des plongements aux probabilités

- On utilise le produit scalaire $\mathbf{m} \cdot \mathbf{c}$ comme mesure de proximité des vecteurs $\mathbf{m}$ et $\mathbf{c}$.
- Le résultat du produit scalaire n'est pas une probabilité
- On le transforme en probabilité à l'aide de la fonction sigmoïde (aussi appelée fonction logistique) :

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

Fonction sigmoïde $f(x) = \frac{1}{1+exp(-ax)}$



- Aussi appelée fonction écrasante ou *smashing function*
- Elle projette l'intervalle $]-\infty, +\infty[$ sur l'intervalle $]0, 1[$.

Calcul de $P(+|m,c)$ et $P(-|m,c)$

- On a donc :

$$P(+|m,c) = \sigma(\mathbf{m} \cdot \mathbf{c}) = \frac{1}{1 + \exp(-\mathbf{m} \cdot \mathbf{c})}$$

- Pour obtenir une distribution de probabilité, il faut que :

$$P(+|m,c) + P(-|m,c) = 1$$

- Donc :

$$P(-|m,c) = 1 - P(+|m,c) = 1 - \sigma(\mathbf{m} \cdot \mathbf{c}) = \sigma(-\mathbf{m} \cdot \mathbf{c})$$

- On verra en TD pourquoi $\sigma(-x) = 1 - \sigma(x)$
- On obtient :

$$P(-|m,c) = \sigma(-\mathbf{m} \cdot \mathbf{c}) = \frac{1}{1 + \exp(\mathbf{m} \cdot \mathbf{c})}$$

Simplification

- On sait calculer $P(+|m, c)$ où c est un mot du contexte de m.
- Mais dans le modèle de la fenêtre glissante centrée sur m_i, il y a plusieurs mots de contexte :
  - L avant : m_{i-L}^{i-1}
  - et L après : m_{i+1}^{i+L}
- On fait l'hypothèse d'indépendance suivante :

$$P(+|m_i, m_{i-L}^{i-1}, m_{i+1}^{i+L}) = \prod_{j=1}^{L} P(+|m_i, m_{i-j})P(+|m_i, m_{i+j})$$

- On néglige :
  - la position relative des mots dans le contexte
  - les dépendances entre mots du contexte

Exemples négatifs

- Etant donné la fenêtre glissante centrée sur m, il est facile de trouver de exemples négatifs.
- Il suffit pour cela de choisir des mots au hasard dans le lexique qui n'apparaissent pas de le contexte de m.
- SGNS combine :
  - un exemple positif (m, c_{pos})
  - et k exemples négatifs $(m, c_{neg_1}) \ldots (m, c_{neg_k})$

Echantillonnage des exemples négatifs

- On echantillonne les mots pour constituer des exemples négatifs à partir de leur probabilité unigramme.
- Problème : étant donné la distribution du lexique les mots rares seront très rarement choisis
- On pondère la probabilité unigramme par un paramètre α :

$$p_\alpha(w) = \frac{count(w)^\alpha}{\sum_{i=1}^{|V|} count(w_i)^\alpha}$$

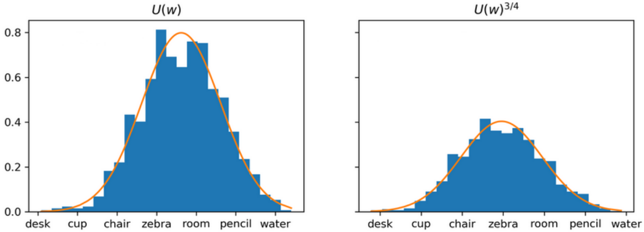
- Ce qui permet d'augmenter la probabilité des mots rares

Illustration

- $V = \{a, b\}$

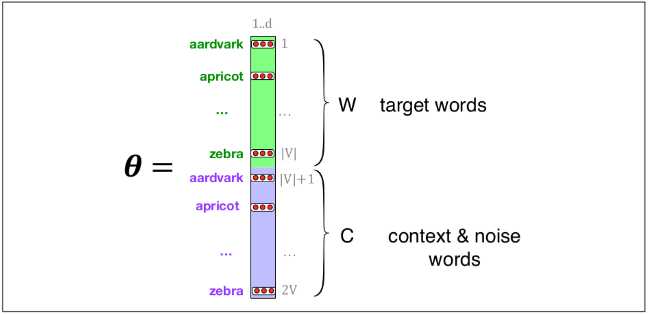
	$\alpha = 1$	$\alpha = 0.75$
$p_\alpha(a)$	0.99	0.97
$p_\alpha(b)$	0.01	0.3

- Exemple plus réaliste :



Source: `https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling`

Les paramètres du classifieur



- Deux matrices de plongements
  - Plongement des mots cible **M** (**W** dans la figure)
  - Plongement des mots de contexte (positifs et négatifs) **C**
- Les matrices sont initialisées aléatoirement et optimisée par descente stochastique du gradient.
- Disposer de deux matrices de plongements permet de simplifier les calculs

<u>Source</u>: Jurafsky & Martin (chp. 6)

Fonction de perte

- Etant donné un ensemble formé d'un exemple positif et de k exemples négatifs, l'objectif de l'apprentissage est de
  - Maximiser la probabilité $P(+|m, c_{pos})$
  - Maximiser les probabilités $P(-|m, c_{neg_i})$
- On définit la fonction de perte suivante :

$$
\begin{aligned}
L &= -\log \left[p(+|m, c_{pos}) \prod_{i=1}^{k} p(-|m, c_{neg_i}) \right] \\
&= -\left[\log p(+|m, c_{pos}) + \sum_{i=1}^{k} \log p(-|m, c_{neg_i}) \right] \\
&= -\left[\log \sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) + \sum_{i=1}^{k} \log \left(\sigma(-\mathbf{c}_{neg_i} \cdot \mathbf{m}) \right) \right]
\end{aligned}
$$

Minimisation de la fonction de perte

- On minimise la fonction de perte par descente stochastique du gradient
- Mise à jour des plongements de $\mathbf{m}$, $\mathbf{c}_{pos}$, et $\mathbf{c}_{neg}$ (k plongements) :

$$\mathbf{c}_{pos}^{t+1} = \mathbf{c}_{pos}^{t} - \eta \frac{\partial L}{\partial \mathbf{c}_{pos}^{t}}$$

$$\mathbf{c}_{neg}^{t+1} = \mathbf{c}_{neg}^{t} - \eta \frac{\partial L}{\partial \mathbf{c}_{neg}^{t}}$$

$$\mathbf{m}^{t+1} = \mathbf{m}^{t} - \eta \frac{\partial L}{\partial \mathbf{m}^{t}}$$

Gradient de la fonction de perte

$$L = -\left[\log \sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) + \sum_{i=1}^{k} \log\left(\sigma(-\mathbf{c}_{neg_i} \cdot \mathbf{m})\right)\right]$$

- Il faut dériver L par rapport aux vecteurs $\mathbf{c}_{pos}$, $\mathbf{c}_{neg}$ et $\mathbf{m}$
- Il est assez simple d'obtenir une expression analytique de ces gradients :

$$\frac{\partial L}{\partial \mathbf{c}_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) - 1]\mathbf{m}$$

$$\frac{\partial L}{\partial \mathbf{c}_{neg}} = [\sigma(\mathbf{c}_{neg} \cdot \mathbf{m})]\mathbf{m}$$

$$\frac{\partial L}{\partial \mathbf{m}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) - 1]\mathbf{c}_{pos} + \sum_{i=1}^{k} [\sigma(\mathbf{c}_{neg_i} \cdot \mathbf{m})]\mathbf{c}_{neg_i}$$

Minimisation de la fonction de perte

$$
\begin{aligned}
\mathbf{c}_{pos}^{t+1} &= \mathbf{c}_{pos}^{t} - \eta[\sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) - 1]\mathbf{m} \\
\mathbf{c}_{neg}^{t+1} &= \mathbf{c}_{neg}^{t} - \eta[\sigma(\mathbf{c}_{neg} \cdot \mathbf{m})]\mathbf{m} \\
\mathbf{m}^{t+1} &= \mathbf{m}^{t} - \eta[\sigma(\mathbf{c}_{pos} \cdot \mathbf{m}) - 1]\mathbf{c}_{pos} + \sum_{i=1}^{k}[\sigma(\mathbf{c}_{neg_i} \cdot \mathbf{m})]\mathbf{c}_{neg_i}
\end{aligned}
$$

Production des plongements

- Lorsque l'apprentissage converge, on arrête la mise à jour
- On garde les plongements de matrice **M**
- Que l'on stocke dans un fichier

```
...
garde -0.346829 -0.070150 0.025763 -0.512991 0.486601 -0.058058 0.214757 ...
trois -0.358720 0.004257 0.258529 -0.026849 0.357005 -0.377576 0.162349 ...
pharaon 0.020979 -0.320600 -0.030742 0.072684 0.153152 0.457672 0.256732 ...
venant -0.038000 -0.076984 0.318350 -0.014221 -0.349837 -0.297711 0.044218 ...
smyrne 0.129223 -0.091417 -0.279704 -0.438414 0.614178 0.169431 0.468693 ...
et 0.479522 0.083401 -0.429599 -0.095315 -0.260707 0.219894 -0.109798 ...
naples -0.066515 0.139142 -0.252495 0.170399 -0.834353 0.100872 0.043674 ...
...
```

Quelques références

- le blog de Eric Kim
  $\rightarrow$ https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling
  $\rightarrow$ https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling
- Illustrated word2vec
  $\rightarrow$ https://jalammar.github.io/illustrated-word2vec/
- Le cours de Chris Manning
  $\rightarrow$ https://www.youtube.com/watch?v=ERibwqs9p38
- Le blog de Chris McCormick
  $\rightarrow$ http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/