



## *Word-Embedding* et sentiments des ménages avec Twitter

KIM ANTUNEZ, ROMAIN LESAUVAGE ET ALAIN  
QUARTIER-LA-TENTE  
11/06/2020  
Ensaë — 2019-2020

# Sommaire

---

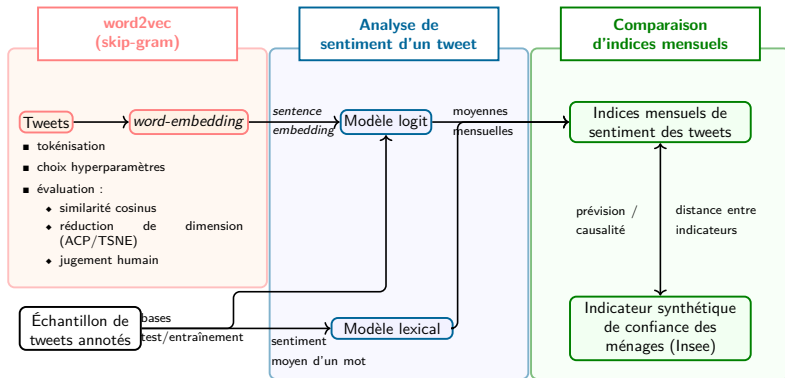
## 1. Introduction

## 2. *word2vec*

## 3. Évaluation du modèle

## 4. Indice de sentiments

# Introduction



# Sommaire

---

1. Introduction

2. *word2vec*

3. Évaluation du modèle

4. Indice de sentiments

# Sommaire

---

## 1. Introduction

## 2. *word2vec*

## 3. Évaluation du modèle

### 3.1 Évaluation sur un corpus fictif

### 3.2 Choix des meilleurs hyperparamètres pour le modèle

### 3.3 Évaluation sur le corpus de tweets

## 4. Indice de sentiments

# Comment évaluer le modèle ?

---

**Problème** : utilisation généralisée des *word-embeddings* mais peu de travaux théoriques expliquent ce qui est capturé, comment évaluer le modèle ?

➡ Utiliser un corpus fictif.

Les vecteurs-mots sont de grande dimension : comment juger de leur proximité ?

Distance entre mots mesurée par distance euclidienne ou **similarité cosinus**.

**ACP** : réduire la dimension et analyser les proximités.

**t-SNE** : algorithme stochastique qui favorise l'apparition de clusters.

**Jugement humain** : corrélations entre mots faites par l'homme, utilisées comme références.

# Évaluation sur un corpus fictif

---

**Idée :** construire un corpus fictif pour lesquels on connaît le résultat attendu en contrôlant les contextes.

**En pratique :**

On génère 10 groupes de mots composés d'un couple de référence et de 10 autres mots contexte.

On construit 10 000 phrases en tirant au hasard :

- 1 des groupes de mots ;

- 1 des 2 mots « références » du groupe ;

- 5 mots contextes ;

- 3 mots bruits parmi une liste de 100 mots.

On mélange les 9 mots de chaque phrase.

## Résultats de l'évaluation

mot	similarité cosinus avec « grand »	mot	similarité cosinus avec « petit »
longueur	0,982	taille	0,987
petit	0,981	longueur	0,983
s	0,979	grand	0,981
⋮	⋮	⋮	⋮
susiens	-0,735	alesiez	-0,745
allates	-0,784	allates	-0,810

*Note : Paramètres utilisés :  $ep = 50$  /  $lr = 0,01$  /  $w = 5$  /  $dim = 10$ .*

On obtient de très bon résultats conformes à ce qui était attendu : l'implémentation du modèle semble validée.



## Choix des meilleurs hyperparamètres pour le modèle

---

*word2vec* se base sur différents choix d'hyperparamètres : taille de la fenêtre ( $w$ ), nombre d'epochs ( $ep$ ), taux d'apprentissage ( $lr$ ), dimension des *word-embeddings* ( $dim$ ).

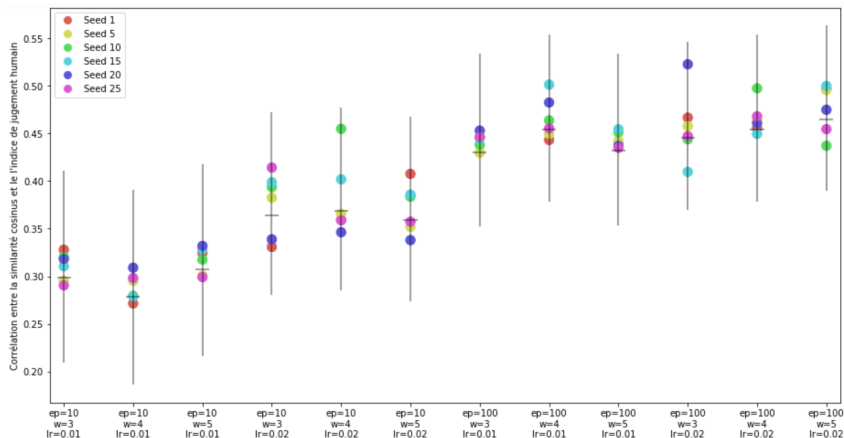
Meilleures valeurs des hyperparamètres déterminées de manière empirique.

Mesure des résultats : utilisation d'un corpus de jugement humain et étude des corrélations de Spearman entre ce corpus et notre modèle.

Problème de temps de compilation de notre algorithme : il faut relancer le modèle à chaque fois.

Utilisation complémentaire de **Gensim** puis résultats validés avec notre implémentation.

# Nombre d'epochs, taille de fenêtre et taux d'apprentissage



Note : Paramètre utilisé :  $\text{dim} = 50$

## Valeurs des hyperparamètres retenus

---

**Nombre d'epochs** : augmentation du nombre d'epochs améliore les résultats

➡ **ep** = 100.

**Taille de fenêtre** : capture différentes informations selon valeur

➡ **w** = 4.

**Taux d'apprentissage** : 0,02 donne de meilleurs résultats

➡ **lr** = 0,02.

Pour la dimension des vecteurs-mots, on observe une amélioration des résultats en augmentant la dimension jusqu'à atteindre 300. En réalité, peu de différences ici entre 100 et 300 ➡ **dim** = 100 par soucis de temps de compilation.

# Évaluation sur le corpus de tweets (1/2)

« Notre » modèle

**Spearman** : 0,57 (p-v : 4,1 %)

➡ **bons résultats**

bonjour (669)	femme (264)	1 (765)	samedi (203)
😊 (0,59)	quelle (0,49)	5 (0,55)	soir (0,57)
😊 (0,59)	cette (0,46)	mois (0,51)	vivement (0,51)
merci (0,54)	une (0,44)	10 (0,49)	demain (0,50)
nuit (0,48)	vie (0,44)	2 (0,48)	end (0,48)
bisous (0,47)	grippe (0,44)	top (0,48)	weekend (0,47)
bonne (0,47)	belle (0,43)	depuis (0,47)	matin (0,45)
😞 (0,46)	ma (0,43)	saison (0,46)	jeudi (0,45)
vous (0,46)	magnifique (0,43)	ans (0,44)	prochain (0,43)
plaisir (0,44)	nouvelle (0,43)	jours (0,43)	week (0,43)
allez (0,43)	vidéo (0,39)	3 (0,43)	🌸 (0,42)

$ep = 80 / w = 4 / lr = 0,02 / dim = 100 / base : 100\ 000\ tweets$

# Évaluation sur le corpus de tweets (1/2)

« Notre » modèle

**Spearman** : 0,57 (p-v : 4,1 %)

➡ **bons résultats**

bonjour (669)	femme (264)	1 (765)	samedi (203)
😊 (0,59)	quelle (0,49)	5 (0,55)	soir (0,57)
😊 (0,59)	cette (0,46)	mois (0,51)	vivement (0,51)
merci (0,54)	une (0,44)	10 (0,49)	demain (0,50)
nuit (0,48)	vie (0,44)	2 (0,48)	end (0,48)
bisous (0,47)	grippe (0,44)	top (0,48)	weekend (0,47)
bonne (0,47)	belle (0,43)	depuis (0,47)	matin (0,45)
😞 (0,46)	ma (0,43)	saison (0,46)	jeudi (0,45)
vous (0,46)	magnifique (0,43)	ans (0,44)	prochain (0,43)
plaisir (0,44)	nouvelle (0,43)	jours (0,43)	week (0,43)
allez (0,43)	vidéo (0,39)	3 (0,43)	🌸 (0,42)

$ep = 80 / w = 4 / lr = 0,02 / dim = 100 / base : 100\ 000\ tweets$

Modèle Gensim

**Spearman** : 0,50 (p-v : 0,0 %)

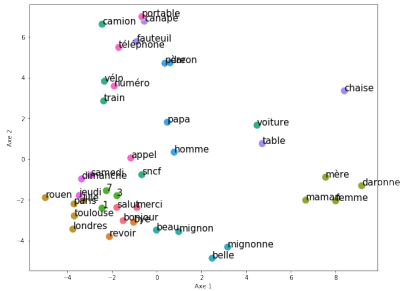
➡ **très bons résultats**

bonjour (17 043)	femme (6 177)	1 (21 055)	samedi (4 917)
bonsoir (0,85)	filles (0,86)	2 (0,65)	vendredi (0,88)
bjr (0,75)	copine (0,74)	3 (0,64)	jeudi (0,86)
hello (0,71)	meuf (0,71)	6 (0,63)	lundi (0,83)
salut (0,66)	demoiselle (0,66)	4 (0,62)	mercredi (0,83)
coucou (0,55)	nana (0,66)	7 (0,60)	dimanche (0,83)
transmets (0,49)	nièce (0,66)	5 (0,58)	mardi (0,76)
désagrement (0,48)	sœur (0,65)	9 (0,58)	demain (0,72)
avezvous (0,48)	barbe (0,65)	8 (0,56)	barathon (0,56)
bettembourg (0,48)	maman (0,64)	1e (0,55)	22h45 (0,55)
hey (0,47)	princesse (0,64)	34 (0,53)	20h (0,54)

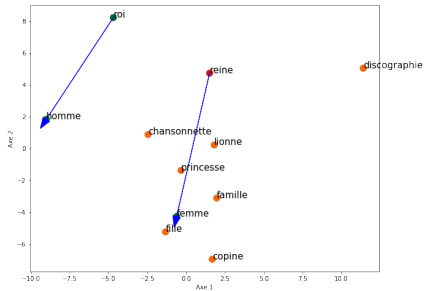
$ep = 100 / w = 4 / lr = 0,02 / dim = 100 / base : ensemble\ des\ tweets$

10 plus proches voisins par similarité cosinus

# Évaluation sur le corpus de tweets (2/2)



ACP sur un corpus réduit de mots.



$$\overrightarrow{Roi} - \overrightarrow{Homme} + \overrightarrow{Femme} = ?$$

➡ Réduction de dimension des vecteurs-mots et (parfois) opérations sur les mots **convaincants**

# Sommaire

---

## 1. Introduction

## 2. *word2vec*

## 3. Évaluation du modèle

## 4. Indice de sentiments

### 4.1 Prédire le sentiment d'un tweet

# Prédire le sentiment d'un tweet

---

**Idée** : associer à chaque tweet un sentiment qui vaut 1 s'il est positif et 0 s'il est négatif.

Pour cela, nous avons à notre disposition une base de tweets annotés sur les transports urbains, contenant 23 000 tweets, séparée en une base de *train* (16 000 tweets) et de *test* (7 000 tweets).

2 approches ici :

Modèle lexical : utiliser l'information des tweets labélisés pour construire un sentiment moyen par mot.

Modèle *logit* : utiliser nos *word-embeddings* comme prédicteurs d'une régression logistique.



## Modèle lexical : sentiment moyen des mots

---

Le sentiment prédit d'un tweet  $t$  composé de  $n$  mots sera :

$$S_{1,\gamma}(t) = 1 \left\{ \frac{1}{n} \sum_{i=1}^n \alpha_i \geq \gamma \right\} \in \{0, 1\}$$

$\gamma \in [-1, 1]$  un seuil fixé ;

$\alpha_i = \frac{nb_+(i) - nb_-(i)}{nb_+(i) + nb_-(i)} \in [-1, 1]$  sentiment moyen du mot  $i$  calculé à partir du nombre de tweets positifs ( $nb_+(i)$ ) et négatifs ( $nb_-(i)$ ) dans lesquels il apparaît.

On détermine le  $\gamma$  optimal en regardant l'*accuracy* sur la base de train : on trouve  $\gamma^* = -0.14$  pour une *accuracy* (sur la base de test) de **89.1%**.

## Modèle logit : prédiction grâce aux *word-embeddings*

$$Y_i = 1 \left\{ \sum_{j=1}^n \beta_j X_{i,j} + \varepsilon_i \geq 0 \right\}$$

$$\mathbb{P}(Y_i = 1 | X_i) = F_\varepsilon \left( \sum_{j=1}^n \beta_j X_{i,j} \right)$$

$Y_i$  le sentiment du tweet  $i$  ;

$X_{i,1}, \dots, X_{i,n}$  les coordonnées de la *sentence-embedding* du tweet  $i$  ;

$\varepsilon_i$  le résidu de notre modèle, de fonction de répartition  $F_\varepsilon$  qui vaudra  $F_\varepsilon(x) = \frac{1}{1+e^{-x}}$  dans le cas d'un modèle logit et  $F_\varepsilon(x) = \Phi(x)$  (fonction de répartition d'une loi  $\mathcal{N}(0,1)$ ) dans le cas d'un modèle probit.

# Spécifications du modèle logit

---

Plusieurs problèmes à traiter :

Doit-on inclure les *stop-words*? ➡ On décide de les garder.

Comment traiter les mots inconnus? ➡ On décide de leur affecter le vecteur du mot *lowfrequency*.

Modèle probit ou logit? ➡ On retient le modèle logit.

Le sentiment du tweet  $t$  est :

$$S_{2,\gamma}(t) = 1 \{ \mathbb{P}(Y_i = 1 | X_i) \geq \gamma \} \quad \in \{0, 1\}$$

On détermine le  $\gamma$  optimal en regardant l'*accuracy* sur la base de train : on trouve  $\gamma^* = 0.5$  pour une *accuracy* (sur la base de test) de **69.8%**.

# Limites des modèles utilisés

---

La bonne performance du modèle lexical par rapport au modèle logit peut s'expliquer par plusieurs facteurs.

1. Les mots inconnus diffèrent selon les modèles.

Modèle lexical : 1,4 % des mots sont inconnus (13,2 % du vocabulaire)

Modèle logit : 4,6 % des mots sont inconnus (36,2 % du vocabulaire)

## Limites des modèles utilisés

---

La bonne performance du modèle lexical par rapport au modèle logit peut s'expliquer par plusieurs facteurs.

1. Les mots inconnus diffèrent selon les modèles.

Modèle lexical : 1,4 % des mots sont inconnus (13,2 % du vocabulaire)

Modèle logit : 4,6 % des mots sont inconnus (36,2 % du vocabulaire)

2. Le processus d'annotation utilisé pour les tweets sur les transports urbains.

## Limites des modèles utilisés

---

La bonne performance du modèle lexical par rapport au modèle logit peut s'expliquer par plusieurs facteurs.

1. Les mots inconnus diffèrent selon les modèles.

Modèle lexical : 1,4 % des mots sont inconnus (13,2 % du vocabulaire)

Modèle logit : 4,6 % des mots sont inconnus (36,2 % du vocabulaire)

2. Le processus d'annotation utilisé pour les tweets sur les transports urbains.
3. Le *domain shift*

## Limites des modèles utilisés

---

La bonne performance du modèle lexical par rapport au modèle logit peut s'expliquer par plusieurs facteurs.

1. Les mots inconnus diffèrent selon les modèles.

Modèle lexical : 1,4 % des mots sont inconnus (13,2 % du vocabulaire)

Modèle logit : 4,6 % des mots sont inconnus (36,2 % du vocabulaire)

2. Le processus d'annotation utilisé pour les tweets sur les transports urbains.

3. Le *domain shift*

Utilisation d'une nouvelle base de test indépendante pour essayer de neutraliser ces effets : **modèle logit meilleur que le modèle lexical** (61,9 % contre 55,9 %).

# Merci pour votre attention

---

 ARKEnsaë/TweetEmbedding

 Rapport du projet

