








Word-Embedding et sentiments des ménages avec Twitter

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

Introduction (1/2)

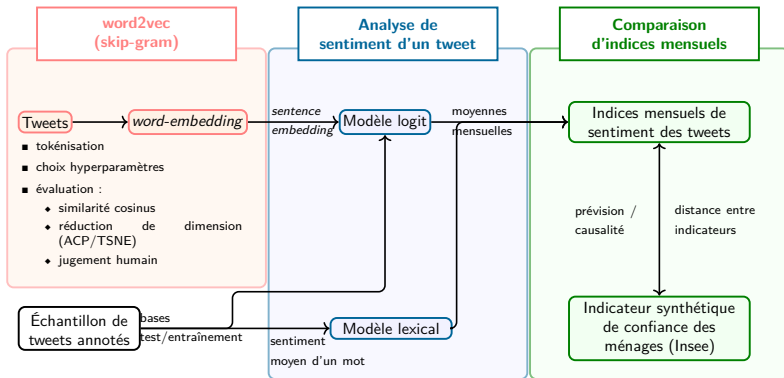
- *word2vec* = modèle de NLP développé par Google (Mikolov et al (2013)).
- **Objectif** = *word-embedding* : donner une représentation vectorielle aux mots.
 - o  vecteurs difficilement interprétables
 - o  tâches d'apprentissage facilitées

Introduction (1/2)

- *word2vec* = modèle de NLP développé par Google (Mikolov et al (2013)).
- **Objectif** = *word-embedding* : donner une représentation vectorielle aux mots.
 - o  vecteurs difficilement interprétables
 - o  tâches d'apprentissage facilitées
-  Réseau de neurones à 2 couches permettant de traiter des grandes bases de données.
 - o mots apparaissant dans un même contexte = représentations vectorielles proches
 - o

$$\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italie} = \overrightarrow{Rome}$$

Introduction (2/2)



Sommaire

1. Le modèle *word2vec*

1.1 L'approche Skip-gram

1.2 Construction de la base d'entraînement

1.3 *softmax* et *negative sampling*

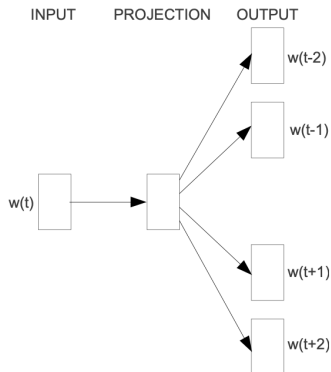
2. Évaluation du modèle

3. Indice de sentiments

L'approche Skip-gram

Approche retenue : Skip-gram

- étant donné un mot *focus* quels pourraient être ses voisins (*contextes*) ?
- les *contextes* dépendent d'un paramètre : la *fenêtre* w



L'approche Skip-gram

Approche retenue : Skip-gram

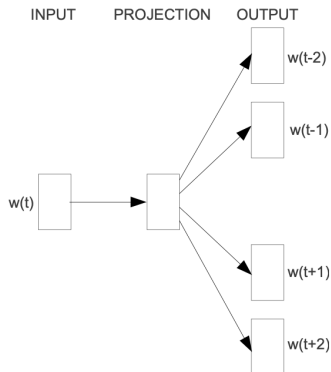
- étant donné un mot *focus* quels pourraient être ses voisins (*contextes*) ?
- les *contextes* dépendent d'un paramètre : la *fenêtre* w

Exemple :

“*Espérons que la présentation sous Teams se passe bien*”

Voisins(passe, $w = 1$) = [se, bien]

Voisins(passe, $w = 2$) = [Teams, se, bien]



Construction de la base d'entraînement (1/2)



À partir de couples [focus, contexte], on met itérativement à jour deux matrices W_e et W_s . Représentation vectorielle finale :

$$\frac{W_e + W_s}{2} = \underbrace{\begin{pmatrix} \text{représentation mot 1} \\ \vdots \\ \text{représentation mot } n \end{pmatrix}}_{\text{dimension } [n \times \text{dim}]}$$

Construction de la base d'entraînement (1/2)



À partir de couples [focus, contexte], on met itérativement à jour deux matrices W_e et W_s . Représentation vectorielle finale :

$$\frac{W_e + W_s}{2} = \underbrace{\begin{pmatrix} \text{représentation mot 1} \\ \vdots \\ \text{représentation mot } n \end{pmatrix}}_{\text{dimension } [n \times \text{dim}]}$$

Pour chaque phrase on :

- supprime la ponctuation, met tout en minuscule
 - effectue un sous-échantillonnage des mots (*subsampling*)
 - tire au hasard un mot *focus* et un mot *contexte* associé
- ➔ on parcourt la base *epochs* fois

Construction de la base d'entraînement (2/2)

Exemple avec $w = 2$:

“Espérons que la présentation sous Teams se passe bien !!!”

– on supprime la ponctuation, met tout en minuscule

➔ [espérons, que, la, présentation, sous, teams, se, passe, bien]

Construction de la base d'entraînement (2/2)

Exemple avec $w = 2$:

“*Espérons que la présentation sous Teams se passe bien !!!*”

- on supprime la ponctuation, met tout en minuscule
➔ [espérons, que, la, présentation, sous, teams, se, passe, bien]
- on effectue un sous-échantillonnage des mots (*subsampling*)
➔ [espérons, X, X, présentation, X, teams, se, passe, X]

Construction de la base d'entraînement (2/2)

Exemple avec $w = 2$:

“*Espérons que la présentation sous Teams se passe bien !!!*”

- on supprime la ponctuation, met tout en minuscule
➔ [espérons, que, la, présentation, sous, teams, se, passe, bien]
- on effectue un sous-échantillonnage des mots (*subsampling*)
➔ [espérons, X, X, présentation, X, teams, se, passe, X]
- on tire au hasard un mot *focus* et un mot *contexte* associé
➔ On tire un couple au hasard parmi [présentation, teams], [teams, présentation], [teams, se], [teams, passe], [se, teams], ...

Actualisation de W_e et W_s

Pour chaque couple [focus, contexte] : actualisation de W_e et W_s par descente de gradient :

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \text{Loss}(\theta^{(t)})$$

η *taux d'apprentissage* et $\text{Loss}(\theta)$ fonction de perte

Actualisation de W_e et W_s

Pour chaque couple [focus, contexte] : actualisation de W_e et W_s par descente de gradient :

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \text{Loss}(\theta^{(t)})$$

η *taux d'apprentissage* et $\text{Loss}(\theta)$ fonction de perte

Deux approches :

- 1 *softmax* : pour un mot focus on estime la probabilité que les autres mots soient voisins (classification multiclasse)

$$\mathbb{P}(w_{\text{contexte}} | w_{\text{focus}}) = ?$$

Actualisation de W_e et W_s

Pour chaque couple [focus, contexte] : actualisation de W_e et W_s par descente de gradient :

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \text{Loss}(\theta^{(t)})$$

η *taux d'apprentissage* et $\text{Loss}(\theta)$ fonction de perte

Deux approches :

- 1 *softmax* : pour un mot focus on estime la probabilité que les autres mots soient voisins (classification multiclasse)

$$\mathbb{P}(w_{\text{contexte}} | w_{\text{focus}}) = ?$$

- 2 *negative sampling* : pour chaque couple [focus, mot2] on estime la probabilité que mot2 soit voisin de focus (classification binaire)

$$\mathbb{P}(D = 1 | w_{\text{focus}}, w_{\text{mot2}}) = ?$$

softmax et negative sampling

Pour chaque couple [focus, contexte] :

1 *softmax* : on maximise

$$\mathbb{P}(w_{\text{contexte}} | w_{\text{focus}}) = \frac{\exp(W_{e, w_{\text{focus}}} \times {}^t W_{s, w_{\text{contexte}}})}{\sum_{i=1}^n \exp(W_{e, w_{\text{focus}}} \times {}^t W_{s, w_i})}$$



Complexité : $\mathcal{O}(n)$ et $n \simeq 70\,000$

softmax et negative sampling

Pour chaque couple [focus, contexte] :

1 *softmax* : on maximise

$$\mathbb{P}(w_{\text{contexte}} | w_{\text{focus}}) = \frac{\exp(W_{e, w_{\text{focus}}} \times {}^t W_{s, w_{\text{contexte}}})}{\sum_{i=1}^n \exp(W_{e, w_{\text{focus}}} \times {}^t W_{s, w_i})}$$



Complexité : $\mathcal{O}(n)$ et $n \simeq 70\,000$

2 *negative sampling* : on tire $K = 5$ mots “négatifs” $(w_{\text{neg}, i})_{i=1..K}$ a priori non liés à [focus, contexte]

On maximise $\mathbb{P}(D = 1 | w_{\text{focus}}, w_{\text{contexte}})$ et $\mathbb{P}(D = 0 | w_{\text{focus}}, w_{\text{neg}, i})$

$$\begin{cases} \mathbb{P}(D = 1 | w_{\text{focus}}, w_{\text{contexte}}) &= \sigma(W_{e, w_{\text{focus}}} {}^t W_{s, w_{\text{contexte}}}) \\ \mathbb{P}(D = 0 | w_{\text{focus}}, w_{\text{neg}, i}) &= \sigma(-W_{e, w_{\text{focus}}} {}^t W_{s, w_{\text{neg}, i}}) \\ \sigma(x) &= \frac{1}{1 + \exp(-x)} \end{cases}$$



Complexité : $\mathcal{O}(K)$

Sommaire

1. Le modèle *word2vec*

2. Évaluation du modèle

- 2.1 Évaluation sur un corpus fictif
- 2.2 Choix des meilleurs hyperparamètres
- 2.3 Évaluation sur le corpus de tweets

3. Indice de sentiments

Comment évaluer le modèle ?

Les vecteurs-mots sont de grande dimension : comment juger de leur qualité et de leurs proximités ?

- **Similarité cosinus** : distance entre vecteurs-mots.
- **ACP et t-SNE** : réduire la dimension et analyser les proximités.
- **Jugement humain** : corrélations entre les proximités de nos vecteurs-mots et une base de proximités de mots construites par le jugement d'individus.

➡ Évaluer sur un corpus fictif puis sur l'ensemble des tweets

Évaluation sur un corpus fictif (1/2)

Idée : construire un corpus fictif pour lequel on connaît le résultat attendu.

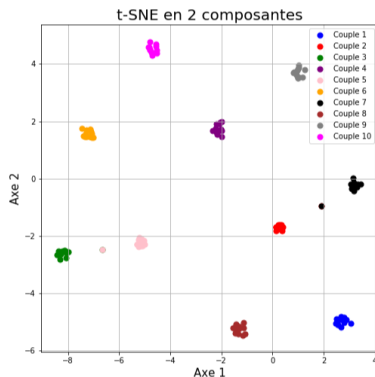
En pratique :

- 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.

Évaluation sur un corpus fictif (2/2)

mot	similarité cosinus avec « grand »
longueur	0,982
petit	0,981
s	0,979
⋮	⋮
allates	-0,784

mot	similarité cosinus avec « petit »
taille	0,987
longueur	0,983
grand	0,981
⋮	⋮
allates	-0,810



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

➡ implémentation semble validée (résultats conformes aux attendus)

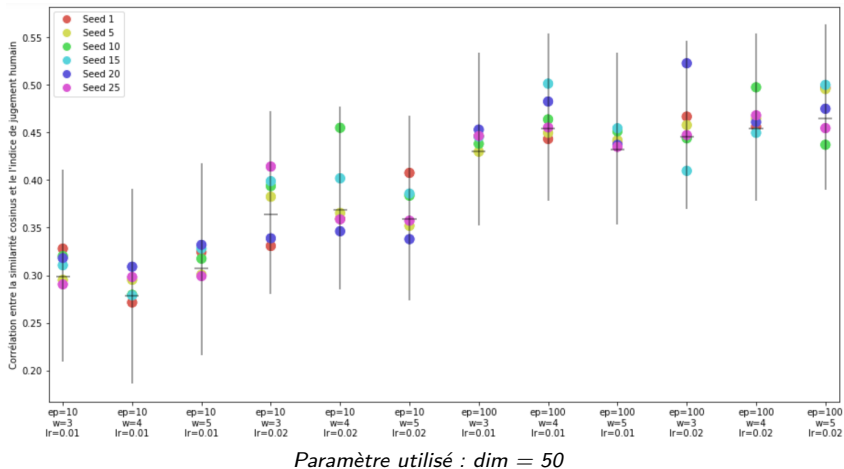
Déterminer les hyperparamètres

- *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)

Déterminer les hyperparamètres

- *Word2vec* se base sur différents choix d'hyperparamètres :
 - o taille de la fenêtre (w)
 - o nombre d'epochs (ep)
 - o taux d'apprentissage (lr)
 - o dimension des *word-embeddings* (dim)
 - Détermination empirique des hyperparamètres :
 - o corrélation de Spearman entre nos vecteurs-mots et une base de jugement humain
 - o chronophage (il faut relancer le modèle à chaque fois).
- ➔ Utilisation complémentaire de Gensim puis validation avec notre implémentation.

Exemple : epochs, fenêtre et taux d'apprentissage



Valeurs retenues pour les hyperparamètres

- **Nombre d'epochs** : qualité des résultats croît avec le nombre d'epochs
➔ $ep = 100$.
- **Taille de fenêtre** : capte des informations sémantiques différentes selon sa valeur
➔ $w = 4$.
- **Taux d'apprentissage** : 0,02 donne de meilleurs résultats
➔ $lr = 0,02$.
- **Dimension** : qualité des résultats croît avec la dimension jusqu'à 300 puis décroît. Peu de différences entre 100 et 300.
➔ $dim = 100$.

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 %)

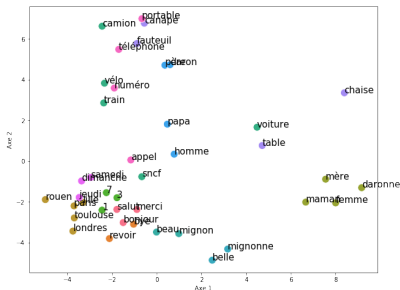
➡ **très bons résultats**

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)

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

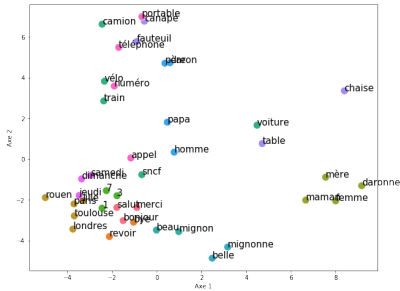
10 plus proches voisins par similarité cosinus

Évaluation sur le corpus de tweets (2/2)

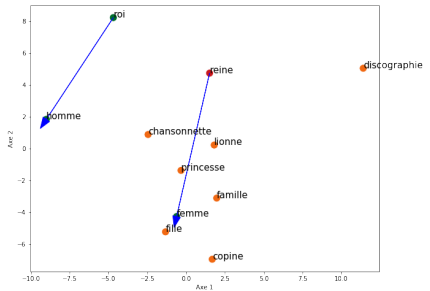


ACP sur un corpus réduit de mots.

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. Le modèle *word2vec*

2. Évaluation du modèle

3. Indice de sentiments

3.1 Prédire le sentiment d'un tweet

3.2 Sentiments des tweets et enquête de conjoncture auprès des ménages

Prédire le sentiment d'un tweet

- **Idée** : associer à chaque tweet un sentiment
 - 1 s'il est positif
 - 0 s'il est négatif
- Base de 23 000 tweets annotés sur les transports urbains :
 - **base d'entraînement** : 16 000 tweets
 - **base de test** : 7 000 tweets
- 2 approches :
 - **Modèle lexical** : utiliser l'information des tweets annotés pour construire un sentiment moyen par mot.
 - **Modèle logit** : utiliser les *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 :

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

- $\gamma \in [-1, 1]$ 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.
- ➡ $Accuracy^1 = 89,1 \%$ ($\gamma^* = -0,14$).

1. Taux de tweets dont le sentiment est bien prédit.

Modèle logit : prédiction grâce aux *word-embeddings*

Le sentiment prédit d'un tweet t sera :

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

Avec :

$$Y_i = \mathbb{1} \left\{ \sum_{j=1}^n \beta_j X_{i,j} + \varepsilon_i \geq 0 \right\} \quad \mathbb{P}(Y_i = 1 | X_i) = F_\varepsilon \left(\sum_{j=1}^n \beta_j X_{i,j} \right)$$

- Y_i le sentiment du tweet i ;
- $X_{i,1}, \dots, X_{i,n}$ les coordonnées de la *sentence-embedding* du tweet i ;
- ε_i le résidu de notre modèle, de fonction de répartition F_ε 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 points à traiter :

- Doit-on inclure les *stop-words*? OUI
- Comment traiter les mots inconnus? AFFECTER LE VECTEUR-MOT LOWFREQUENCY
- Modèle probit ou logit? LOGIT

➡ $Accuracy = 69,8 \%$ ($\gamma^* \simeq 0,5$).

Limites des modèles utilisés

Modèle lexical ici meilleur que le modèle logit car ...

- 1 Davantage de mots inconnus dans le modèle logit (4,6 % des mots contre 1,4 % dans le modèle lexical).

Limites des modèles utilisés

Modèle lexical ici meilleur que le modèle logit car ...

- 1 Davantage de mots inconnus dans le modèle logit (4,6 % des mots contre 1,4 % dans le modèle lexical).
- 2 Le processus d'annotation utilisé pour les tweets sur les transports urbains reproduit en partie par le modèle lexical ?

Limites des modèles utilisés

Modèle lexical ici meilleur que le modèle logit car ...

- 1 Davantage de mots inconnus dans le modèle logit (4,6 % des mots contre 1,4 % dans le modèle lexical).
- 2 Le processus d'annotation utilisé pour les tweets sur les transports urbains reproduit en partie par le modèle lexical ?
- 3 Le *domain shift*.

Limites des modèles utilisés

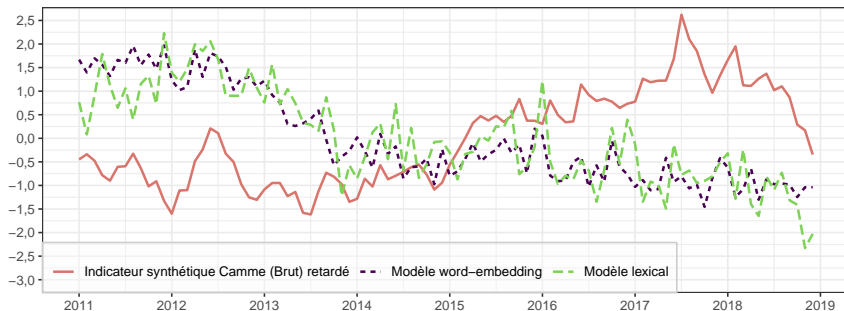
Modèle lexical ici meilleur que le modèle logit car ...

- 1 Davantage de mots inconnus dans le modèle logit (4,6 % des mots contre 1,4 % dans le modèle lexical).
- 2 Le processus d'annotation utilisé pour les tweets sur les transports urbains reproduit en partie par le modèle lexical ?
- 3 Le *domain shift*.

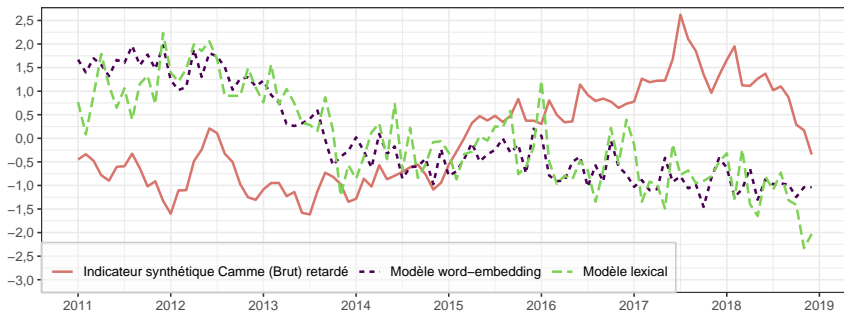
➡ Utilisation d'une nouvelle base de test pour neutraliser certains de ces effets.

Modèle logit alors meilleur que le modèle lexical
(*Accuracy* de 61,9 % contre 55,9 %).

Sentiments des tweets et enquête Camme

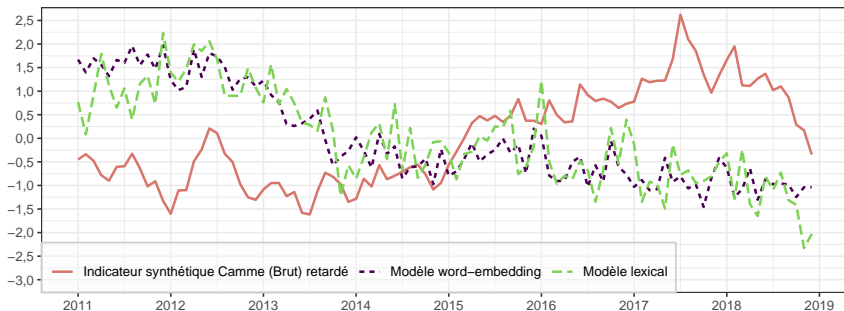


Sentiments des tweets et enquête Camme



- Indicateurs relativement éloignés de l'enquête Camme
- Similarité (DTW) avec indicateur Camme plus proche avec modèle lexical que modèle word-embedding

Sentiments des tweets et enquête Camme



- Indicateurs relativement éloignés de l'enquête Camme
- Similarité (DTW) avec indicateur Camme plus proche avec modèle lexical que modèle word-embedding



- Modèle word-embedding utile pour prévoir indicateur Camme (causalité de Granger) \neq modèle lexical
- Modèle indicateur **avancé** des sentiments des ménages

Conclusion (1/2)

- *Word2vec* ...
 - capture **très bien** la sémantique des mots dans un texte
 - prédit **assez bien** le sentiment d'une phrase
 - est **potentiellement utile** pour prédire l'indicateur synthétique de confiance des ménages de l'Insee ...
 - ... mais demeure **très différent** de cet indicateur (en évolution)

Conclusion (1/2)

- *Word2vec* ...
 - capture **très bien** la sémantique des mots dans un texte
 - prédit **assez bien** le sentiment d'une phrase
 - est **potentiellement utile** pour prédire l'indicateur synthétique de confiance des ménages de l'Insee ...
 - ... mais demeure **très différent** de cet indicateur (en évolution)
- Pourquoi très différent ?
 - Principalement en raison de leurs **différentes philosophies** (sujets spécifiques de Camme VS positivité ou non des tweets pour notre indice) ...
 - ... mais aussi à cause des **limites de la base d'entraînement** de tweets annotés (domain-shift, processus d'annotation, mots inconnus)

Conclusion (2/2)

Pistes d'amélioration ?

- disposer d'une **base de tweets traitant de sujets divers, et bien annotés** (gradation de sentiments, modèles de type BERT, analyse approfondie du contenu et des auteurs des tweets ...)
- améliorer le **prétraitement des tweets** (orthographe des mots, modèle à séquences d'unités de sous-mots type *fasttext* ...)
- utiliser des **modèles d'analyse de sentiment plus élaborés** (type réseaux de neurones récurrents)

Merci pour votre attention

 ARKEnsaë/TweetEmbedding

 Rapport du projet

