

Sémantique (lexicale) distributionnelle

Chapitre 6, Jurafsky & Martin (2019)

P. Amsili

nov. 2020

Programme

Plusieurs types d'embeddings :

- Tf-Idf
- ▶ Une baseline courante
 - ▶ Vecteurs dispersés (*sparse*)
 - ▶ Les mots sont représentés par une fonction simple des fréquences des mots voisins

- PPMI
- ▶ mesure de l'informativité des contextes par rapport à un mot

- Word2vec
- ▶ Vecteurs denses
 - ▶ créés par l'entraînement d'un classifieur à distinguer les mots proches et lointains

Embeddings + récents

Tf-Idf

Term-frequency / Inverse document frequency

- ▶ Méthode de pondération (de la relation entre un mot-clé (*terme*) et un document)
- ▶ Courante en *Recherche d'information* (et Fouille de Textes)
- ▶ Une des plus anciennes utilisations des modèles par espace vectoriel (1971)
- ▶ *terme* = token ou lemme ou mot-clé ou mwe ou n-gramme...

Matrice terme-document

	QuatreVT 119 Kw	Voyage Bal 82 kw	Bête Hum. 128 kw	Mme Bovary 117 kw
bataille	35	4	6	2
clair	105	26	96	52
facile	12	19	6	10
politique	11	0	9	5
voyage	17	196	94	44
idiot	2	1	2	6
amour	19	0	47	94

Quatrevingt-treize (Hugo)

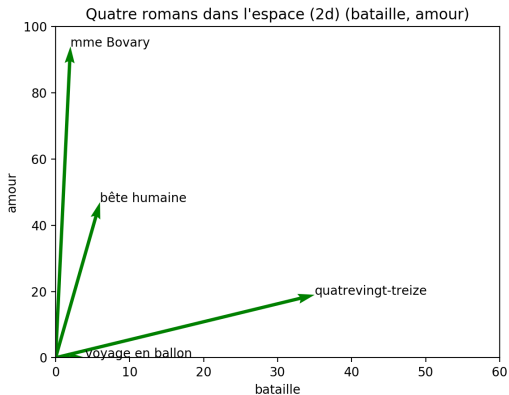
Le voyage en ballon (Verne)

La bête humaine (Zola)

Mme Bovary (Flaubert)

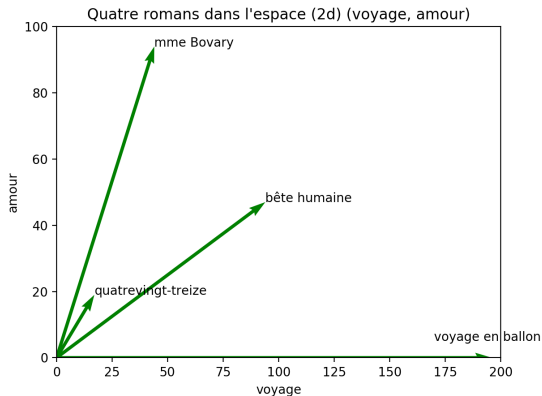
Documents comme vecteurs

	QuatreVT	Voyage Bal	Bête Hum.	Mme Bovary
bataille	35	4	6	2
amour	19	0	47	94



Documents comme vecteurs

	QuatreVT	Voyage Bal	Bête Hum.	Mme Bovary
voyage	17	196	94	44
amour	19	0	47	94



Méthode IR

Recherche d'information : identification du document d dans la collection D qui correspond le mieux à une requête q .

La requête q peut être représentée par un vecteur (de taille $|V|$)

On doit trouver une mesure de similarité entre chaque (vecteur de) document et la requête.

Vecteurs terme-document

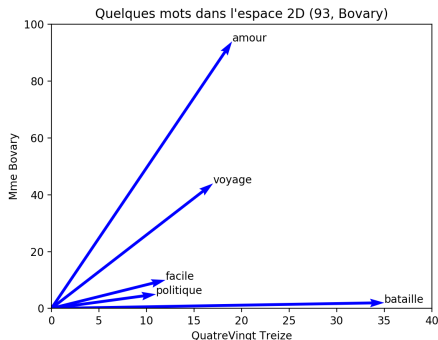
On peut inverser la représentation : les dimensions sont maintenant les documents, les vecteurs permettent de décrire des mots.

	QuatreVT 119 Kw	Voyage Bal 82 kw	Bête Hum. 128 kw	Mme Bovary 117 kw
bataille	35	4	6	2
clair	105	26	96	52
facile	12	19	6	10
politique	11	0	9	5
voyage	17	196	94	44
idiot	2	1	2	6
amour	19	0	47	94

amour (comme *politique*)

est le genre de mot qui n'apparaît pas dans "Le voyage en ballon".

On peut visualiser les mots dans l'espace (Quatrevingt-treize, Mme Bovary) :



bataille	(35,2)
politique	(11,5)
amour	(19,94)
voyage	(17,44)

Matrice terme-terme

More common: word-word matrix
(or "term-context matrix")

Two **words** are similar in meaning if their context vectors are similar

sugar, a sliced lemon, a tablespoonful of
their enjoyment. Cautiously she sampled her first
well suited to programming on the digital
for the purpose of gathering data and

apricot
pineapple
computer.
information

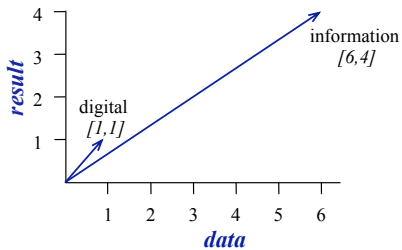
jam, a pinch each of,
and another fruit whose taste she likened
In finding the optimal R-stage policy from
necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Embeddings

└ Vecteurs dispersés

└ Tf-Idf



Reminders from linear algebra

$$\text{dot-product}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$$

$$\text{vector length } |\vec{v}| = \sqrt{\sum_{i=1}^N v_i^2}$$

Cosine for computing similarity Sec 8.3

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

v_i is the count for word v in context i

w_i is the count for word w in context i .

→ →

Cos(v, w) is the cosine similarity of v and w

→ →

$$\vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos \theta$$

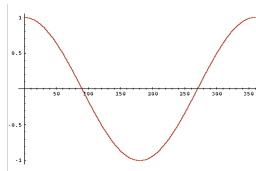
$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} = \cos \theta$$

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



Frequency is non-negative, so cosine range 0-1

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\vec{v} \cdot \vec{w}}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

Which pair of words is more similar?

cosine(apricot, information) =

$$\frac{1+0+0}{\sqrt{1+0+0} \sqrt{1+36+1}} = \frac{1}{\sqrt{38}} = .16$$

cosine(digital, information) =

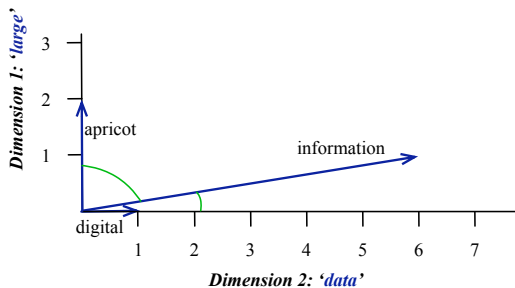
$$\frac{0+6+2}{\sqrt{0+1+4} \sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

cosine(apricot, digital) =

$$\frac{0+0+0}{\sqrt{1+0+0} \sqrt{0+1+4}} = 0$$

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Visualizing cosines (well, angles)



Discussion : fréquence brute

Les fréquences brutes sont problématiques

- ▶ La fréquence est utile :
Si *sugar* apparaît plus que *apricot* c'est une information utile
- ▶ Mais les mots trop fréquents (comme *it*, *the*) ne sont pas informatifs
- ▶ La normalisation (par la longueur du texte) peut aider, mais elle ne change rien à ce paradoxe.

⇒ Tf-Idf

tf-idf: combine two factors

tf: term frequency. frequency count (usually log-transformed):

$$tf_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Idf: inverse document frequency: tf-

$$idf_i = \log \left(\frac{N}{df_i} \right)$$

Total # of docs in collection

of docs that have word i

Words like "the" or "good" have very low idf

tf-idf value for word t in document d:

$$w_{t,d} = tf_{t,d} \times idf_t$$

tf-idf : synthèse

- ▶ similarité de deux mots (en prenant le cosinus)
- ▶ similarité de deux documents (en prenant le baricentre de tous les mots du document)

Une alternative à tf-idf

Chercher à mesurer si un mot du contexte est **particulièrement informatif** à propos du mot cible.

- ▶ Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X, Y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
 - Things are co-occurring **less than** we expect by chance
 - Unreliable without enormous corpora
 - Imagine w_1 and w_2 whose probability is each 10^{-6}
 - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}
 - Plus it's not clear people are good at "unrelatedness"
- So we just replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:

$$\text{PPMI}(\text{word}_1, \text{word}_2) = \max\left(\log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}, 0\right)$$

Computing PPMI on a term-context matrix

Matrix F with W rows (words) and C columns (contexts)

f_{ij} is # of times w_i occurs in context c_j

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

$$P_{ij} = \frac{f_{ij}}{W \cdot C}$$

$$P_{i*} = \frac{\sum_{j=1}^C f_{ij}}{W \cdot C}$$

$$P_{*j} = \frac{\sum_{i=1}^W f_{ij}}{W \cdot C}$$

$$pmi_{ij} = \log_2 \frac{P_{ij}}{P_{i*} P_{*j}}$$

$$ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

apricot
pineapple
digital
information

Count(w,context)					
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$p(w=\text{information},c=\text{data}) = 6/19 = .32$

$p(w=\text{information}) = 11/19 = .58$

$p(c=\text{data}) = 7/19 = .37$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N}$$

$$p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$pmi_{ij} = \log_2 \frac{P_{ij}}{P_i P_j}$$

		p(w,context)					p(w)
		computer	data	pinch	result	sugar	
apricot		0.00	0.00	0.05	0.00	0.05	0.11
pineapple		0.00	0.00	0.05	0.00	0.05	0.11
digital		0.11	0.05	0.00	0.05	0.00	0.21
information		0.05	0.32	0.00	0.21	0.00	0.58
	p(context)	0.16	0.37	0.11	0.26	0.11	

$$pmi(\text{information}, \text{data}) = \log_2 \left(\frac{.32}{(.37 * .58)} \right) = .58$$

(.57 using full precision)

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

Weighting PMI

PMI is biased toward infrequent events

- Very rare words have very high PMI values

Two solutions:

- Give rare words slightly higher probabilities
- Use add-one smoothing (which has a similar effect)

Weighting PMI: Giving rare context words slightly higher probability

Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_{\alpha}(w, c) = \max\left(\log_2 \frac{P(w, c)}{P(w)P_{\alpha}(c)}, 0\right)$$

$$P_{\alpha}(c) = \frac{\text{count}(c)^{\alpha}}{\sum_c \text{count}(c)^{\alpha}}$$

This helps because $P_{\alpha}(c) > P(c)$ for rare c

Consider two events, $P(a) = .99$ and $P(b) = .01$

$$P_{\alpha}(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97 \quad P_{\alpha}(b) = \frac{.01^{.75}}{.01^{.75} + .99^{.75}} = .03$$

Use Laplace smoothing (add -1)



	Add-2 Smoothed Count(w,context)				
	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

	p(w,context) [add-2]					p(w)
	computer	data	pinch	result	sugar	
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(context)	0.19	0.25	0.17	0.22	0.17	

PPMI versus add-2 smoothed PPMI

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

	PPMI(w,context) [add-2]				
	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

Plan

Sémantique lexicale

Vecteurs dispersés

Sémantique distributionnelle : principe

Tf-Idf

PPMI

Vecteurs denses

Word2vec

Propriétés des embeddings

Densité

Les vecteurs construits avec tf-idf et PPMI sont longs (entre 20k et 50k), et creux.

Les vecteurs dont nous allons parler maintenant sont courts (entre 50 et 1000 dimensions), et denses (presque pas de zéros).

Avantage des vecteurs denses :

- ▶ les vecteurs de petite taille peuvent être plus faciles à utiliser pour l'apprentissage (moins de poids à régler)
- ▶ les vecteurs de petite taille peuvent généraliser mieux que les vecteurs de fréquence
- ▶ ils peuvent aussi mieux représenter la synonymie :
 - ▶ *voiture* et *automobile* sont des synonymes, mais s'ils sont pris comme deux dimensions distinctes, les voisins de ces deux mots ne sont pas considérés comme similaires
- ▶ **en pratique, ils fonctionnent mieux**

Word2vec

- ▶ Première proposition vraiment à large couverture,
- ▶ très répandue encore aujourd'hui,
- ▶ rapide à entraîner,
- ▶ le code est accessible sur Internet.
- ▶ Idée : faire des **prédictions** plutôt que des **comptages**.

Principe général

- ▶ Au lieu de compter combien de fois chaque mot w apparaît dans le voisinage du mot « abricot »
- ▶ on entraîne un classifieur sur une tâche de prédiction binaire :
 - ▶ Est-ce que w a des chances d'apparaître dans le voisinage d'« abricot » ?
- ▶ La tâche en elle-même n'est pas le but de l'opération
- ▶ mais elle va fournir des poids appris qui vont représenter le mot

Avancée théorique

Avec cette méthode, on *utilise du texte courant comme données d'entraînement supervisées implicitement*

- ▶ Tout mot s dans le voisinage de « abricot »
 - ▶ fonctionne comme une réponse correcte “gold” à la question :
 - ▶ Est-ce que w a des chances d'apparaître dans le voisinage d'« abricot » ?
- ▶ Aucune supervision manuelle n'est requise
- ▶ L'idée vient du domaine de la modélisation de langage neuronale
 - ▶ Bengio et al (2003)
 - ▶ Collobert et (2011)

Les différents algorithmes

Deux grands types d'algorithmes dans l'article de Mikolov :

- ▶ Continuous Bag of Words (CBOW)
- ▶ Skip-gram

On présente dans ce qui suit l'architecture SGNS (skip-gram with negative sampling)

Algorithme skip-gram

1. On considère le mot cible et le voisinage comme des exemples positifs
2. On tire au hasard d'autres mots du lexique pour obtenir des exemples négatifs
3. On utilise la régression logistique pour entraîner un classifieur pour distinguer ces deux cas
4. Les poids appris sont utilisés comme plongement.

Données d'entraînement

Phrase d'entraînement :

... lemon, **tablespoon** of **apricot** jam a pinch...
 c1 c2 **target** c3 c4

Objectif du skip-gram : étant donné un tuple (t,c) (target, context-word)

- ▶ (apricot, jam)
- ▶ (apricot, aardvark)

→ retourner la probabilité que c soit un vrai mot de contexte.

- ▶ $P(+|t, c)$
- ▶ $P(-|t, c) = 1 - P(+|t, c)$

Calcul de la probabilité

Intuition :

- ▶ Les mots ont des chances d'apparaître près de mots similaires
- ▶ On modélise la similarité avec le produit scalaire
- ▶ $\text{Similarity}(t,c) \propto t \cdot c$

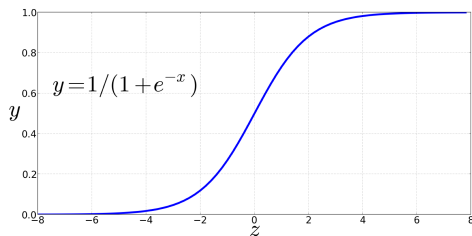
Problème :

- ▶ Le produit scalaire n'est pas une probabilité!
(le cosinus non plus)

Transformation d'un produit scalaire en probabilité

La courbe sigmoïde évolue entre 0 et 1 :

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



Transformation d'un produit scalaire en probabilité (suite)

$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

$$P(-|t, c) = 1 - P(+|t, c) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

Pour les mots de contexte
(sous l'hypothèse qu'ils sont indépendants)

$$P(+|t, c_{1:k}) = \prod_{i=1}^k \frac{1}{1 + e^{-t \cdot c_i}}$$

$$\log P(+|t, c_{1:k}) = \sum_{i=1}^k \log \frac{1}{1 + e^{-t \cdot c_i}}$$

Exemples positifs et négatifs

... lemon, **tablespoon** of **apricot** jam a pinch...
 c1 c2 **target** c3 c4

positifs		négatifs			
t	c	t	c	t	c
apricot	tablespoon	apricot	aardvark	apricot	twelve
apricot	of	apricot	puddle	apricot	hello
apricot	jam	apricot	where	apricot	dear
apricot	a	apricot	coaxial	apricot	forever

Pour chaque exemple positif, on crée k exemples négatifs, en utilisant des mots au hasard ($\neq t$)

Choix des mots « négatifs (*noise words*) »

Le tirage au sort pourrait se faire selon la fréquence des unigrammes (le mot *le* serait choisi avec une probabilité de $\frac{\text{count}('le')}{N}$), mais on préfère utiliser une fréquence pondérée $P_\alpha(w)$:

$$P_\alpha(w) = \frac{\text{count}(w)^\alpha}{\sum_{w^j} \text{count}(w^j)^\alpha}$$

On choisit fréquemment $\alpha = \frac{3}{4}$.

Utiliser une pondération augmente la probabilité des mots rares.

Mise en place

On démarre avec chaque mot représenté par un vecteur d'une taille donnée (p.ex. 300), initialisé au hasard.

On a donc au départ $300 \times V$ paramètres aléatoires

Sur la totalité du jeu d'entraînement, on va chercher à ajuster les vecteurs de mots de telle sorte que

- ▶ la similarité des paires (t, c) avec c dans les exemples positifs soit maximale
- ▶ la similarité des paires (t, c) avec c dans les exemples négatifs soit minimale

On veut donc maximiser :

$$\sum_{(t,c) \in +} \log P(+|t, c) + \sum_{(t,c) \in -} \log P(-|t, c)$$

Si on choisit un mot cible t

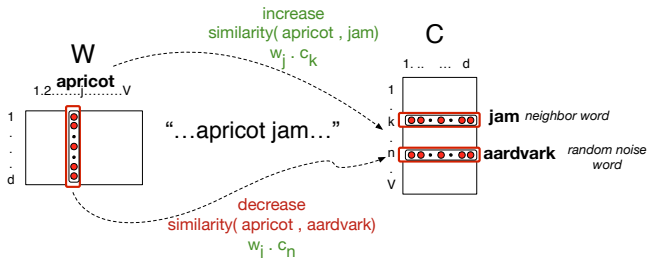
$$L(\theta) = \log P(+|t, c) + \sum_{i=1}^k \log P(-|t, n_i)$$

$$= \log \sigma(t \cdot c) + \sum_{i=1}^k \log \sigma(-n_i \cdot t)$$

$$= \log \frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^k \log \frac{1}{1+e^{n_i \cdot t}}$$

Entraînement

On réalise l'entraînement avec une descente de gradient.



En fait, cette méthode apprend en même temps deux embeddings pour chaque mot.

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random 300-dimensional vectors as initial embeddings

Use logistic regression, the second most basic classifier used in machine learning after naïve bayes

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random 300-dimensional vectors as initial embeddings

Use logistic regression, the second most basic classifier used in machine learning after naïve bayes

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

Evaluating embeddings

Compare to human scores on word similarity-type tasks:

- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Properties of embeddings

Similarity depends on window size C

$C = \pm 2$ The nearest words to *Hogwarts*:

- *Sunnydale*
- *Evernight*

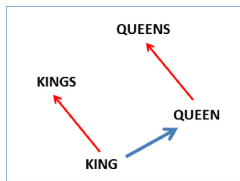
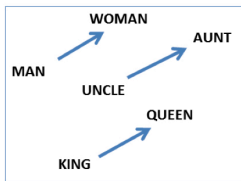
$C = \pm 5$ The nearest words to *Hogwarts*:

- *Dumbledore*
- *Malfoy*
- *halfblood*

Analogy: Embeddings capture relational meaning!

$\text{vector}('king') - \text{vector}('man') + \text{vector}('woman') \approx \text{vector}('queen')$

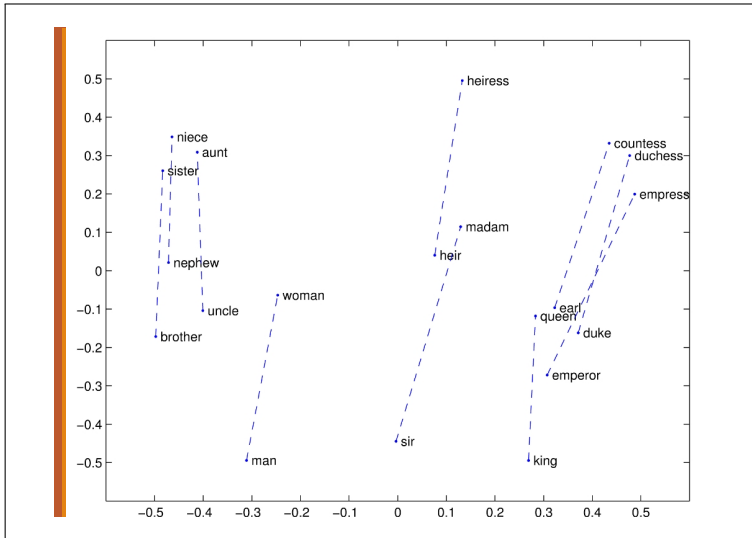
$\text{vector}('Paris') - \text{vector}('France') + \text{vector}('Italy') \approx \text{vector}('Rome')$



Embeddings

↳ Vecteurs denses

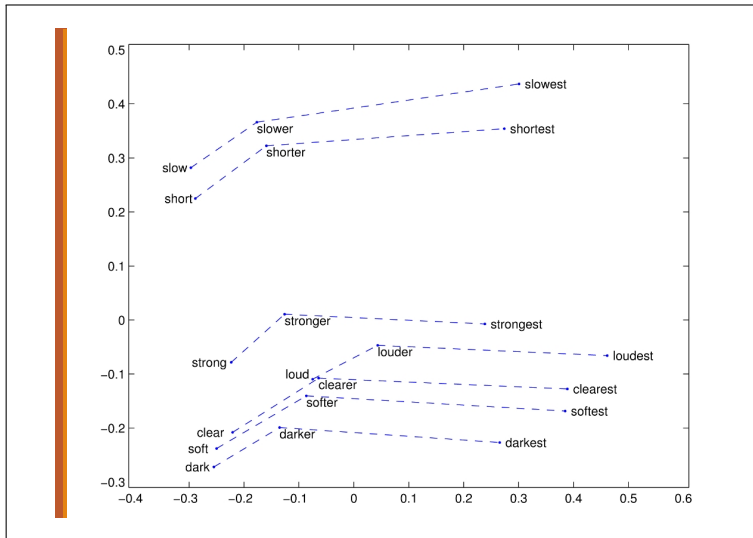
↳ Propriétés des embeddings




Embeddings

↳ Vecteurs denses

↳ Propriétés des embeddings





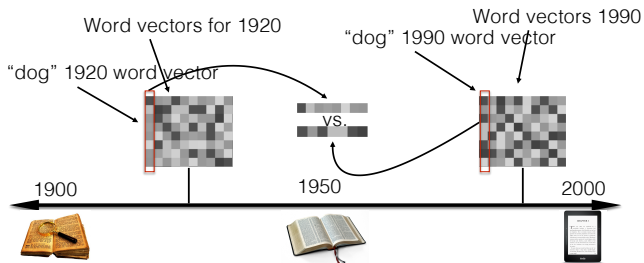
Embeddings can help study
word history!

Train embeddings on old books to study
changes in word meaning!!



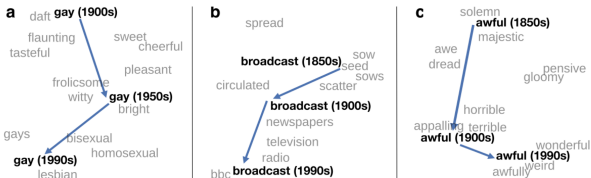
Will Hamilton

Diachronic word embeddings for studying language change!



Visualizing changes

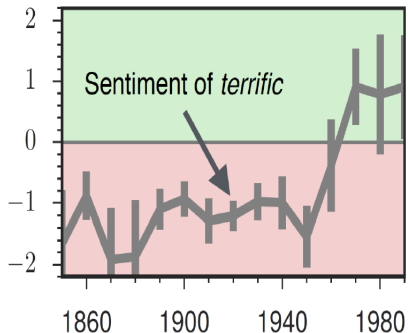
Project 300 dimensions down into 2



~30 million books, 1850-1990, Google Books data

The evolution of sentiment words

Negative words change faster than positive words





Embeddings and bias

Embeddings reflect cultural bias

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

Ask "Paris : France :: Tokyo : x"

- x = Japan

Ask "father : doctor :: mother : x"

- x = nurse

Ask "man : computer programmer :: woman : x"

- x = homemaker

Embeddings reflect cultural bias

Caliskan, Aylin, Joanna J. Bruson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356:6334, 183-186.

Implicit Association test (Greenwald et al 1998): How associated are

- concepts (*flowers, insects*) & attributes (*pleasantness, unpleasantness*)?
- Studied by measuring timing latencies for categorization.

Psychological findings on US participants:

- African-American names are associated with unpleasant words (more than European-American names)
- Male names associated more with math, female names with arts
- Old people's names with unpleasant words, young people with pleasant words.

Caliskan et al. replication with embeddings:

- African-American names (*Leroy, Shaniqua*) had a higher GloVe cosine with unpleasant words (*abuse, stink, ugly*)
- European American names (*Brad, Greg, Courtney*) had a higher cosine with pleasant words (*love, peace, miracle*)

Embeddings reflect and replicate all sorts of pernicious biases.

Directions

Debiasing algorithms for embeddings

- Bolukbasi, Tolga, Chang, Kai-Wei, Zou, James Y., Saligrama, Venkatesh, and Kalai, Adam T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pp. 4349–4357.

Use embeddings as a historical tool to study bias

Embeddings as a window onto history

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

Use the Hamilton historical embeddings

The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names

- Is correlated with the actual percentage of women teachers in decade X

History of biased framings of women

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

Embeddings for competence adjectives are biased toward men

- *Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.*

This bias is slowly decreasing

Embeddings reflect ethnic stereotypes over time

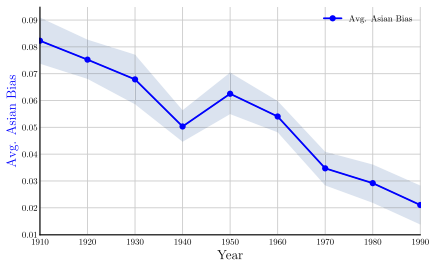
Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
 - *industrious, superstitious, nationalistic*, etc
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

Change in linguistic framing 1910-1990

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

Change in association of Chinese names with adjectives framed as "othering" (*barbaric, monstrous, bizarre*)



Changes in framing: adjectives associated with Chinese

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

1910	1950	1990
Irresponsible	Disorganized	Inhibited
Envious	Outrageous	Passive
Barbaric	Pompous	Dissolute
Aggressive	Unstable	Haughty
Transparent	Effeminate	Complacent
Monstrous	Unprincipled	Forceful
Hateful	Venomous	Fixed
Cruel	Disobedient	Active
Greedy	Predatory	Sensitive
Bizarre	Boisterous	Hearty

Conclusion

Concepts or word senses

- Have a complex many-to-many association with **words** (homonymy, multiple senses)
- Have relations with each other
 - Synonymy, Antonymy, Superordinate
- But are hard to define formally (necessary & sufficient conditions)

Embeddings = vector models of meaning

- More fine-grained than just a string or index
- Especially good at modeling similarity/analogy
 - Just download them and use cosines!!
- Can use sparse models (tf-idf) or dense models (word2vec, GLoVE)
- Useful in practice but know they encode cultural stereotypes

References

Jurafsky, Daniel, & Martin, James H. 2019. *Speech and Language Processing : An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall. drafts of August 29, 2019.