Les embeddings et la sémantique vectorielle

TALN Semaine 5

Merci à Dan Jurafsky pour les diapos et l'inspiration !

Plan pour aujourd'hui

1. **Le sens des mots**
2. **Les embeddings et leurs propriétés**
3. **Trois types:**
    1. **Embeddings épars (TF-IDF)**
    2. **Embeddings denses (Word2Vec/skipgram)**
    3. **Embeddings contextualisés (LM)**
4. ***Exercices de groupe***

Le sens des mots

Qu’est ce que le sens d’un mot?

Les modèle N-gram que nous avons vues jusqu'à présent:

- Les mots ne sont que des strings (ou des indices w_i dans une liste de vocabulaire)
- Ce n'est pas très satisfaisant !

Cours d’introduction à la logique:

- Le sense de "dog" est DOG; cat est CAT

$$\forall x \; DOG(x) \longrightarrow MAMMAL(x)$$

Vieille blague linguistique par Barbara Partee en 1967:

- Q: What's the meaning of life?
- A: LIFE

Desiderata

Qu'est-ce qu'une théorie du sens des mots devrait faire pour nous ?

Regardons quelques desiderata

De la sémantique lexicale, l'étude linguistique du sens des mots:

Lemme et sens

Lemme

mouse (N)

sens

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Modified from the online thesaurus WordNet

Un sens ou « concept » est la composante de sens d'un mot

Les lemmes peuvent être polysémiques (avoir plusieurs sens)

Relations entre les sens : Synonyme

Les synonymes ont le même sens dans certains ou tous les contextes.

- filbert / hazelnut
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- water / H_20

Relations entre les sens : Synonyme

Notez qu'il n'y a probablement pas d'exemples de synonymie parfaite.

- Même si de nombreux aspects du sens sont identiques
- Il peut encore différer en fonction de la politesse, du slang, du registre, du genre, etc.

Relation: Synonyme?

water/H_20

"H_20" dans un guide de surfing?

grand/large

ma grande soeur != ma soeur large

je ne crois pas qu'il y ait de-
mot ſynonime dans aucune
Langue. Je le dis par con-

Abbé Gabriel Girard, 1718

LA JUSTESSE
DE LA
LANGUE FRANÇOISE,
OU
LES DIFFERENTES SIGNIFICATIONS
DES MOTS QUI PASSENT
POUR
SYNONIMES.

Par M. l'Abbé GIRARD C. D. M. D. D. F.

A PARIS,
Chez LAURENT D'HOURY, Imprimeur-Libraire, au bas de la rue de la Harpe, vis-à vis la rue S. Severin, au Saint Eſprit.

M. DCC. XVIII.
Avec Approbation & Privilege du Roy.

Principle de contraste en linguistiques

Difference de forme → difference de sens

Relation : Similitude

Des mots avec des significations similaires. Pas des synonymes, mais partageant un élément de sens

`car, bicycle`

`cow, horse`

Demandez aux gens à quel point 2 mots sont similaires

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

Relation: L'association

Les mots peuvent être associés de n'importe quelle manière, comme via un cadre ou un champ sémantique

- `coffee, tea:` **similaires**
- `coffee, cup:` **associés**, pas similaires

Champs sémantiques

Des mots qui ensemble

- Couvrent un domaine sémantique particulier
- Entretiennent des relations structurées les uns avec les autres.

hospitals

surgeon, scalpel, nurse, anaesthetic, hospital

restaurants

waiter, menu, plate, food, menu, chef

houses

door, roof, kitchen, family, bed

Relation: Antonymie

Sens qui sont opposés par rapport à une seule caractéristique du sens

Sinon, ils sont très similaires !

`dark/light` `short/long` `fast/slow` `rise/fall`
`hot/cold` `up/down` `in/out`

Plus formellement : les antonymes peuvent

- Définir une opposition binaire ou être aux extrémités opposées d'une échelle
  - `long/short, fast/slow`
- Être *réversibles*:
  - `rise/fall, up/down`

Connotation (sentiment)

- Les mots ont des significations affectives
  - Connotations positives (heureux)
  - Connotations négatives (tristes)

- Les connotations peuvent être subtiles :
  - Connotation positive : copie, répliqua, reproduction
  - Connotation négative : faux, imitation

- Évaluation (sentiment !)
  - Évaluation positive (super, amour)
  - Évaluation négative (terrible, haine)

Sens des mots

Pour résumer

Les Concepts ou sens

ont des associations complexes avec plusieurs **mots (ex. Homonymes)**

ont des relations les uns avec les autres sens

- Synonyme
- Antonyme
- Similarité
- Association
- Connotation

Modèles informatiques du sens des mots

Pouvons-nous construire une théorie sur la façon de représenter le sens des mots, qui représente au moins certaines des desiderata ?

Nous introduirons la **sémantique vectorielle**

- Le standard en TALN pour representer le sens!
- Gère beaucoup de nos objectifs !

Ludwig Wittgenstein, PI #43:
"The meaning of a word is its use in the language"

Définissons les mots par leurs usages

Une façon de définir "l'usages" :

Les mots sont définis par leur environnement (les mots qui les entourent)

Zellig Harris (1954):

Si A et B ont des environnements presque identiques, nous disons qu'ils sont synonymes.

Que signifie le récent emprunt anglais *ong choi* ?

Suppose you see these sentences:

- Ong choi est **délicieux sauté à l'ail.**
- Ong choi est superbe **sur le riz**
- **Feuilles** de choi Ong avec sauces **salées**

And you've also seen these:

- …Épinards **sautés à l'ail sur du riz**
- Les tiges et les **feuilles** de blette sont **délicieuses**
- Choux verts et autres légumes aux feuilles **salées**

Conclusion:

- L'ongchoi est légume vert à feuilles comme les épinards, les blettes ou les choux verts
  - Nous pourrions conclure cela sur la base de mots comme "feuilles" et "délicieux" et "sauté"

Représentations distribuées

Définissons le sens d'un mot par sa distribution dans l'usage de la langue, c'est-à-dire ses mots voisins ou ses environnements grammaticaux.

Sémantique vectorielle

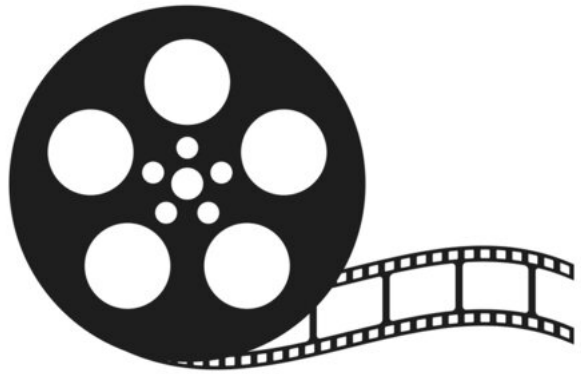
Chaque mot = un vecteur (pas juste "good" ou "w_{45}")

Les mots similaires sont "proche dans **l'espace sémantique**"

Nous construisons cet espace automatiquement utilisant les mots à proximité dans le texte

Analyse des sentiments

Critique de film positive ou négative ?



- 👎 unbelievably disappointing
- 👍 Full of zany characters and richly applied satire, and some great plot twists
- 👍 this is the greatest screwball comedy ever filmed
- 👎 It was pathetic. The worst part about it was the boxing scenes.

Embeddings

Les représentations vectorielles du sens des mots sont appelées "embedding" parce qu'elles sont “embedded” dans un espace.

Les algorithmes de TALN moderne utilise des embeddings comme représentation du sens des mots

Modèle du sens basé sur la similitude

Intuition : pourquoi les vecteurs ?

Considérez l'analyse des sentiments :

- avec les mots, une caractéristique est l'identité du mot

  Ex. Caractéristique 5 : Le mot précédent était « terrible »

  Nécessite exactement **le même mot** en entraînement ou en test

- avec les embeddings, une caractéristique est un vecteur de mot

  Ex. Caractéristique 5 : Le mot précédent était vecteur [35,22,17]

  Maintenant, dans l'ensemble de tests, nous pourrions voir un vecteur similaire [34,21,14]

  Nous pouvons généraliser à des mots similaires mais non vus !!!

Propriétés des embeddings

La taille de fenêtres considérées détermine les mots voisinants

Petites fenêtres (C= +/- 2) : les mots les plus proches sont des mots syntaxiquement similaires dans la même taxonomie

- Les voisins les plus proches de *Hogwarts* sont d'autres écoles fictives
  - *Sunnydale, Evernight, Blandings*

Grandes fenêtres (C= +/- 5) : les mots les plus proches sont des mots apparentés dans le même champ sémantique

- Les voisins les plus proches de *Hogwarts* sont dans le monde de Harry Potter:
  - *Dumbledore, half-blood, Malfoy*

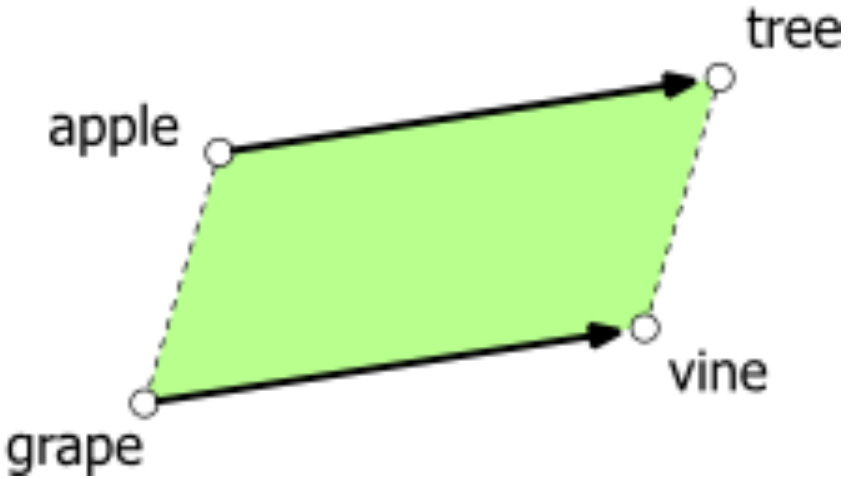
Propriétés des embeddings

Relations analogiques

Le modèle de parallélogramme classique du raisonnement analogique (Rumelhart and Abrahamson 1973)

To solve: *"apple is to tree as grape is to _____"*

Add $\overrightarrow{tree}$ – $\overrightarrow{apple}$ *to* $\overrightarrow{grape}$ *to get* $\overrightarrow{vine}$



Relations analogiques par parallélogramme

La méthode du parallélogramme peut résoudre des analogies avec des embeddings à la fois épars et denses (Turney and Littman 2005, Mikolov et al. 2013b)

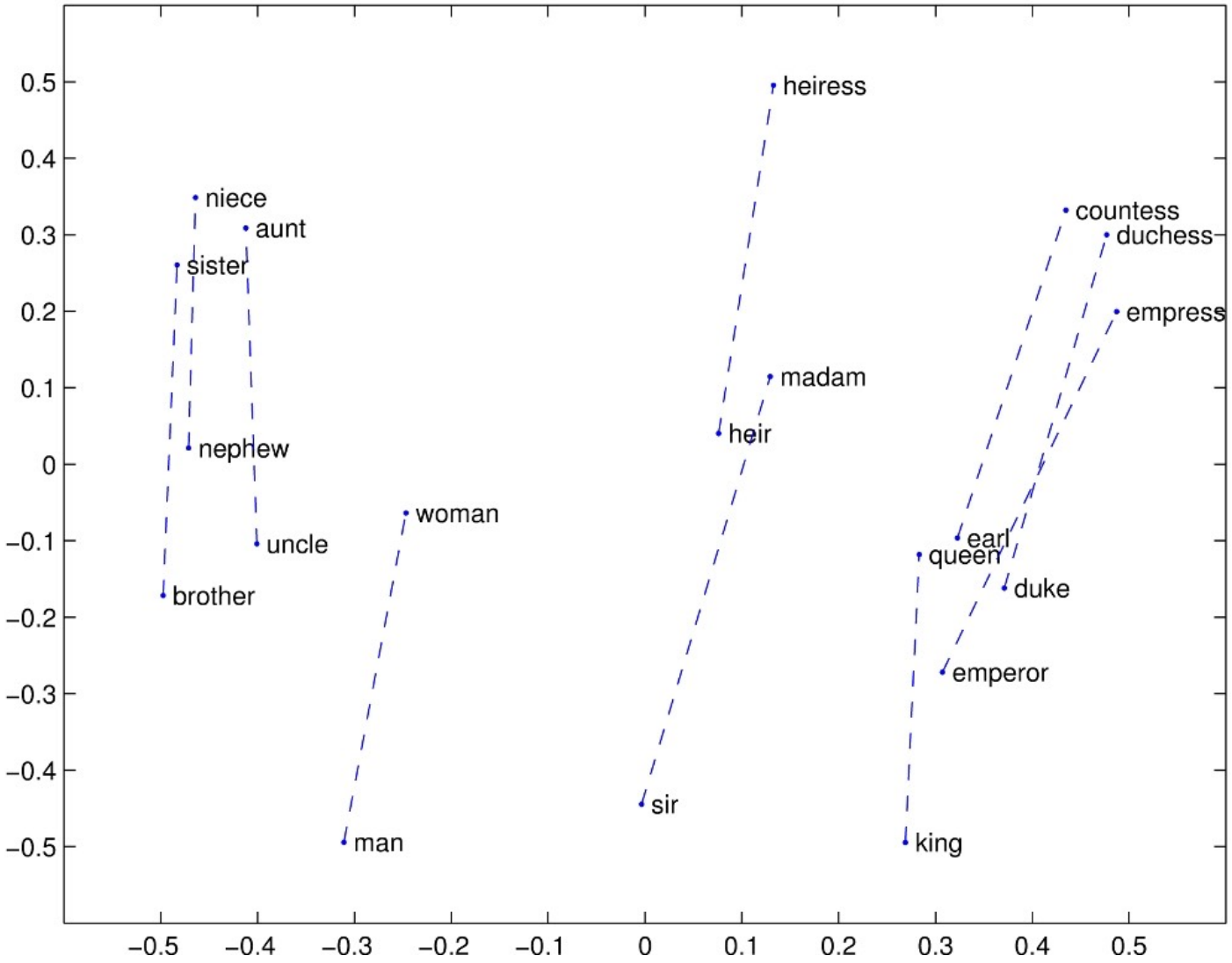
$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$

$\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}}$ is close to $\overrightarrow{\text{Rome}}$

Pour un problème a:a*::b:b*, la méthode du parallélogramme est :

$$\hat{b}^* = \underset{x}{\text{argmax}}\ \text{distance}(x, a^* - a + b)$$

Espace sémantique GloVE



Mises en garde avec la méthode du parallélogramme

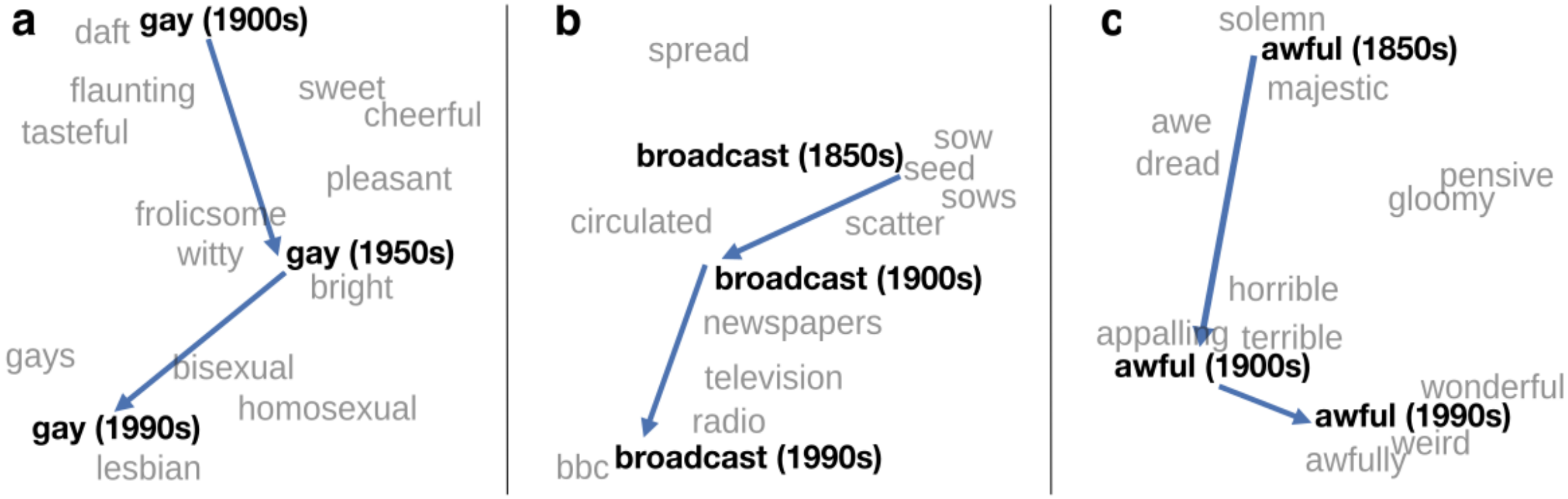
Cela ne semble fonctionner que pour les mots fréquents, les petites distances et certaines relations (reliant les pays aux capitales ou aux parties du discours), mais pas à d'autres.(Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)

Comprendre l'analogie est un domaine de recherche ouvert (Peterson et al. 2020)

Embeddings comme fenêtre sur la sémantique historique

Entraîner des embeddings sur différentes décennies de texte historique pour voir les sens changer

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



William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

Les embeddings reflètent les biais culturels !

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 *NeurIPS*, pp. 4349-4357. 2016.

“Paris : France :: Tokyo : x”

- x = Japan

“father : doctor :: mother : x”

- x = nurse

“man : computer programmer :: woman : x”

- x = homemaker

Les algorithmes qui utilisent des enbeddings dans le cadre, par exemple, des recherches d'embauche pour les programmeurs, pourraient conduire à des préjugés dans l’embauche

Les types d’embeddings

- **Il existe trois principaux types d'embeddings utilisés en TALN et ils sont créés en utilisant trois méthodes différentes :**

    1. Embeddings épars et “Term Frequency-Inverse Document Frequency (TF-IDF)

    2. Embeddings denses et Word2Vec et l’algorithme skip-gram

    3. Embeddings contextualisés et les LLMs

Embeddings éparses et TF-IDF

Term-document matrix

Chaque document est représenté par un vecteur de mots

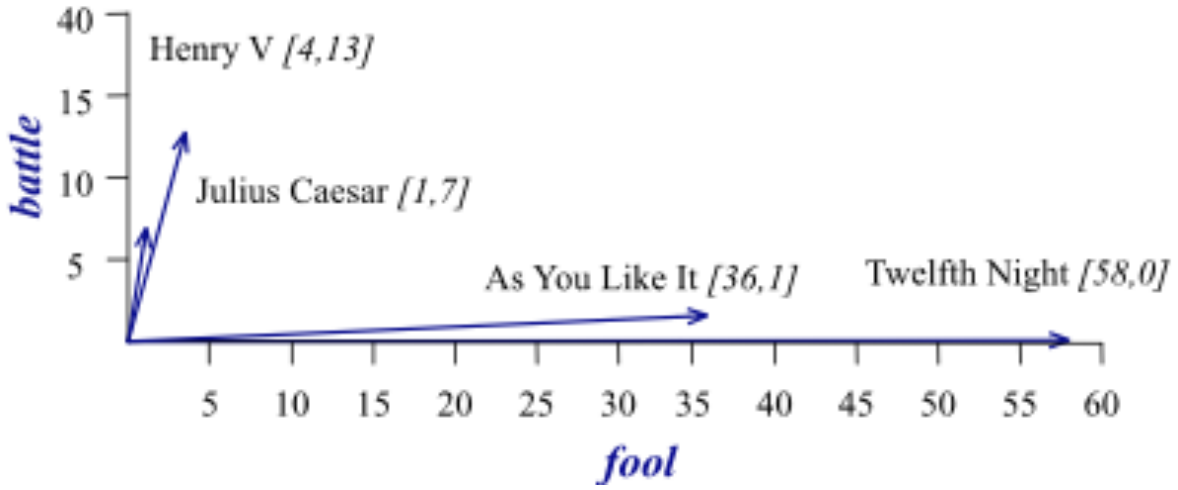
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Les vecteurs sont la base de la recherche sémantique de documents

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Les vecteurs sont similaires pour les deux comédies

Les comedies ont plusieurs occurrences de *fool* et *wit* et peu de *battles*.



Les mots peuvent aussi être des vecteurs !

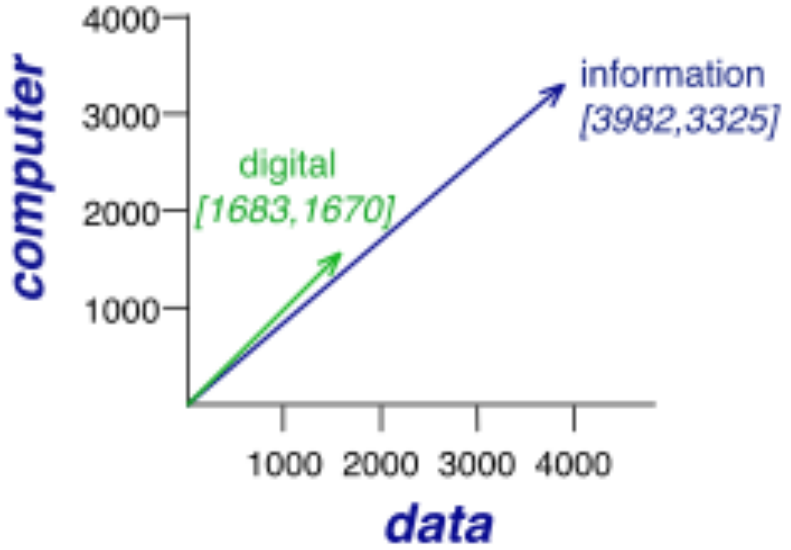
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle est “le type de mot qui apparait dans *Julius Caesar* et *Henry V*"

fool est "le type de mot qui apparait dans des comedies, come *Twelfth Night*”

Plus courant : matrice mot-mot (Ou "matrice terme-contexte")

Deux **mots** ont un sens similaire si leurs vecteurs de contexte sont similaires



is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

	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	...

la fréquence seule est une mauvaise représentation

- Les matrices de co-occurrence que nous avons vues représentent chaque cellule par des fréquences de mots.
- La fréquence est clairement utile ; si *sucre* apparaît beaucoup près de *abricot*, c'est une information utile.
- Mais les mots trop fréquents comme *le*, *il* ou *de* ne sont pas très instructifs sur le contexte.
- C'est un paradoxe ! Comment pouvons-nous équilibrer ces deux contraintes contradictoires ?

Deux solutions courantes pour la pondération des mots

tf-idf: (term frequency x inverse document frequency)

Valeur tf-idf pour mot *t* dans le document *d* :

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Les mots comme "le" ou "il" ont un idf très faible

PMI: (Pointwise mutual information)

- $\mathrm{PMI}(w_1, w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$

Vérifier si des mots comme "bon" apparaissent plus souvent avec "super" que le hasard.

Term frequency (tf) dans l'algorithme tf-idf

Nous pourrions imaginer utiliser le compte brut des mots dans un document:

$$\text{tf}_{t,d} = \text{count}(t,d)$$

Mais généralement nous écrasons un peu les comptes :

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

Document frequency (df)

df_t est le nombre de documents dans lesquels le mot *t* apparait.

(Notez qu'il ne s'agit pas de la fréquence collective : nombre total sur tous les documents)

"*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

N est le nombre total de documents dans la collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Qu'est-ce qu'un document ?

Pourrait être une pièce de théâtre ou un article de Wikipédia

Mais aux fins de tf-idf, les documents peuvent être n'importe quoi ; nous appelons souvent chaque paragraphe un document !

Valeur pondérée tf-idf finale pour chaque mot

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Comptes brute:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.246	0	0.454	0.520
good	0	0	0	0
fool	0.030	0.033	0.0012	0.0019
wit	0.085	0.081	0.048	0.054

TF-IDF

- L’outil par excellence de la recherche de document!
- Un modèle de base commun
- Vecteurs épars
- Les mots sont représentés par (une fonction simple donnant) les comptes des mots voisininants

Les types d’embeddings

- **Il existe trois principaux types d'embeddings utilisés en TALN et ils sont créés en utilisant trois méthodes différentes :**

  1. **Embeddings épars et “Term Frequency-Inverse Document Frequency (TF-IDF)**

  2. Embeddings denses et Word2Vec et l’algorithme skip-gram

  3. Embeddings contextualisés et les LLMs

Embeddings denses

Vecteurs épars par rapport aux vecteurs denses

tf-idf (or PMI) : vecteurs sont

- **longs** (|V|= 20,000 to 50,000)
- **épars** (La plupart des valeurs sont zéros)

Alternative: Apprendre des vecteurs qui sont

- **courts** (50-1000)
- **denses** (La plupart des valeurs sont non-zéros)

Vecteurs épars versus denses

Pourquoi des vecteurs denses ?

- Les vecteurs courts peuvent être plus faciles à utiliser en tant que **caractéristiques** dans un modèle neuronal (moins de poids à régler)
- Les vecteurs denses peuvent mieux **généraliser** que les comptes brutes
- Les vecteurs denses peuvent mieux **capturer la synonymie** :
  - *voiture* et *automobile* sont synonymes ; mais sont des dimensions distinctes . Un mot avec *voiture* comme voisin et un mot avec *automobile* comme voisin devraient être similaires, mais ne le sont pas
- **En pratique, ils fonctionnent mieux**

Similitude dans l'espace vectoriel

Produit scalaire et similarité cosinus

Le produit scalaire de deux vecteurs :

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

Le produit scalaire a tendance à être élevé lorsque les deux vecteurs ont de grandes valeurs dans les mêmes dimensions.

Il peut donc être une mesure de similitude utile entre les vecteurs.

Problème avec le produit scalaire

- Il est biaisé vers les longs vecteurs: le produit scalaire est plus élevé si un vecteur est plus long (des valeurs plus élevées dans de nombreuses dimensions)
- Longueur du vecteur:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

- Les mots fréquents (de, le, vous) ont de longs vecteurs (puisqu'ils apparaissent plusieurs fois avec d'autres mots).
- Donc, le produit scalaire favorise excessivement les mots fréquents

Alternative : cosinus pour calculer la similitude des mots

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

Basé sur la définition du produit scalaire entre deux vecteurs **a** et **b**

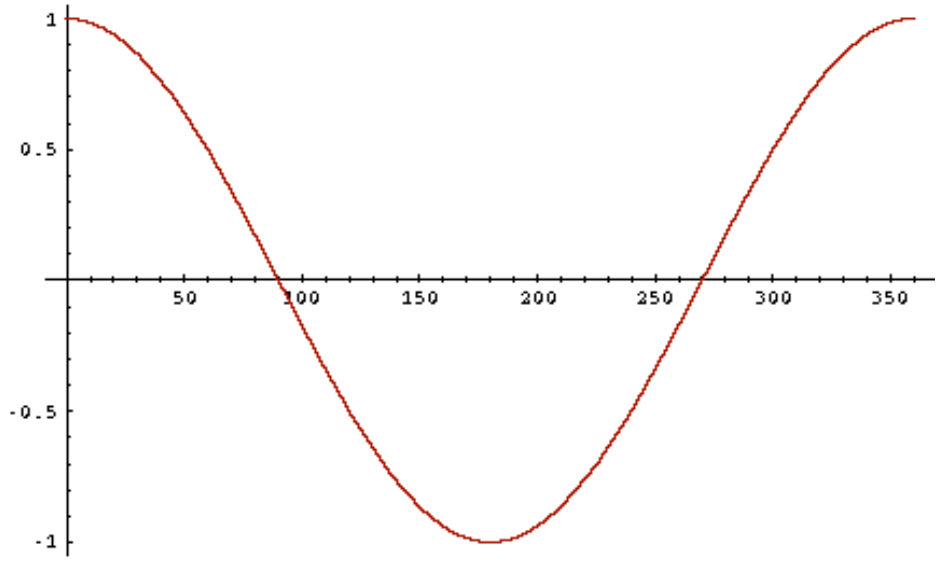
$$\begin{aligned} \mathbf{a}\cdot\mathbf{b} &= |\mathbf{a}||\mathbf{b}|\cos\theta \\ \frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} &= \cos\theta \end{aligned}$$

Cosinus comme métrique de similitude

-1: Les vecteurs pointent dans des directions opposées

+1: Les vecteurs pointent dans la même direction

0: Les vecteurs sont orthogonaux



Mais comme les valeurs de fréquence brutes ne sont pas négatives, le cosinus pour les vecteurs dans tf-idf va de 0 à 1

Exemples de cosinus

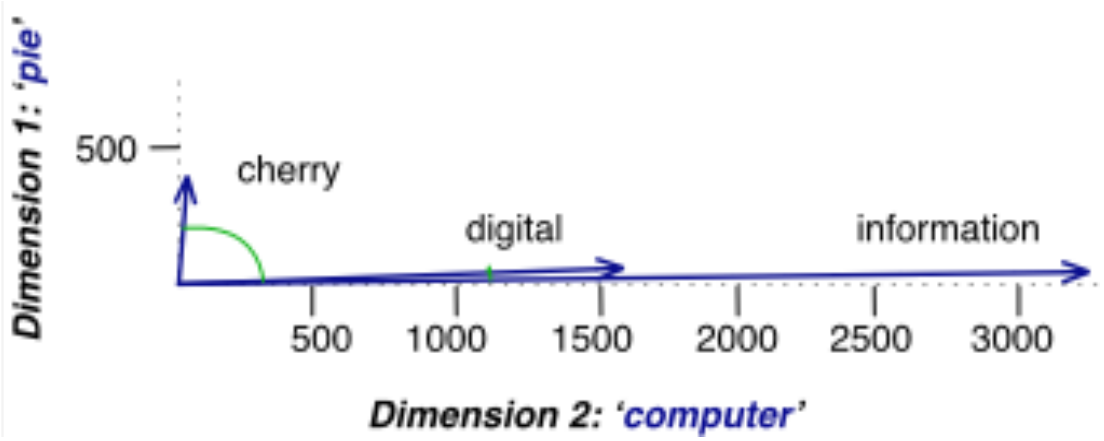
$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

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

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

Visualisez de la distance du cosinus comme les angles



Embeddings denses et Word2Vec

Au lieu de compter la fréquence à laquelle chaque mot ***w*** apparaît près de "*abricot*":

- **Entraîner un classificateur** sur une tâche de prédiction binaire : Est-ce que ***w*** est susceptible de se présenter près de *abricot* ?

Nous ne nous soucions pas vraiment de cette tâche, mais nous prendrons les poids du classificateur une fois entraîné comme embeddings

L'idée: l'**auto-supervision**:

- Pas besoin d'étiquettes humaines
- Bengio et al. (2003); Collobert et al. (2011)

L’approche auto-supervisée

Approche : prédire si le mot ***c*** du mot cible est un "voisin"

1. Traitez le mot cible ***w*** et un mot de contexte voisin ***c*** comme un exemple positif.
2. Échantillonnez au hasard d'autres mots dans le lexique pour obtenir des exemples négatifs
3. Utiliser la régression logistique pour entraîner un classificateur à distinguer les paires positives et négatives
4. Utilisez les poids appris comme embeddings

Classificateur Skip-gram

Classificateur probabiliste, étant donné

- Un mot cible $\boldsymbol{w}$
- Sa fenêtre de contexte de L mots $\boldsymbol{c}_{1:L}$

Estimer la probabilité que $\boldsymbol{w}$ soit dans cette fenêtre en fonction de la similitude de $\boldsymbol{w}$ (embedding) avec $\boldsymbol{c}_{1:L}$ (embeddings).

Pour calculer cela, nous avons besoin d'embeddings pour tous les mots.

Classificateur Skip-gram

(Supposons une fenêtre de +/- 2 mots)

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [cible] c3 c4

Objectif : Entraîner un classificateur à reconnaître les paires positives des mots cibles (**w**ord, **c**ontext)

(apricot, jam)

(apricot, aardvark)

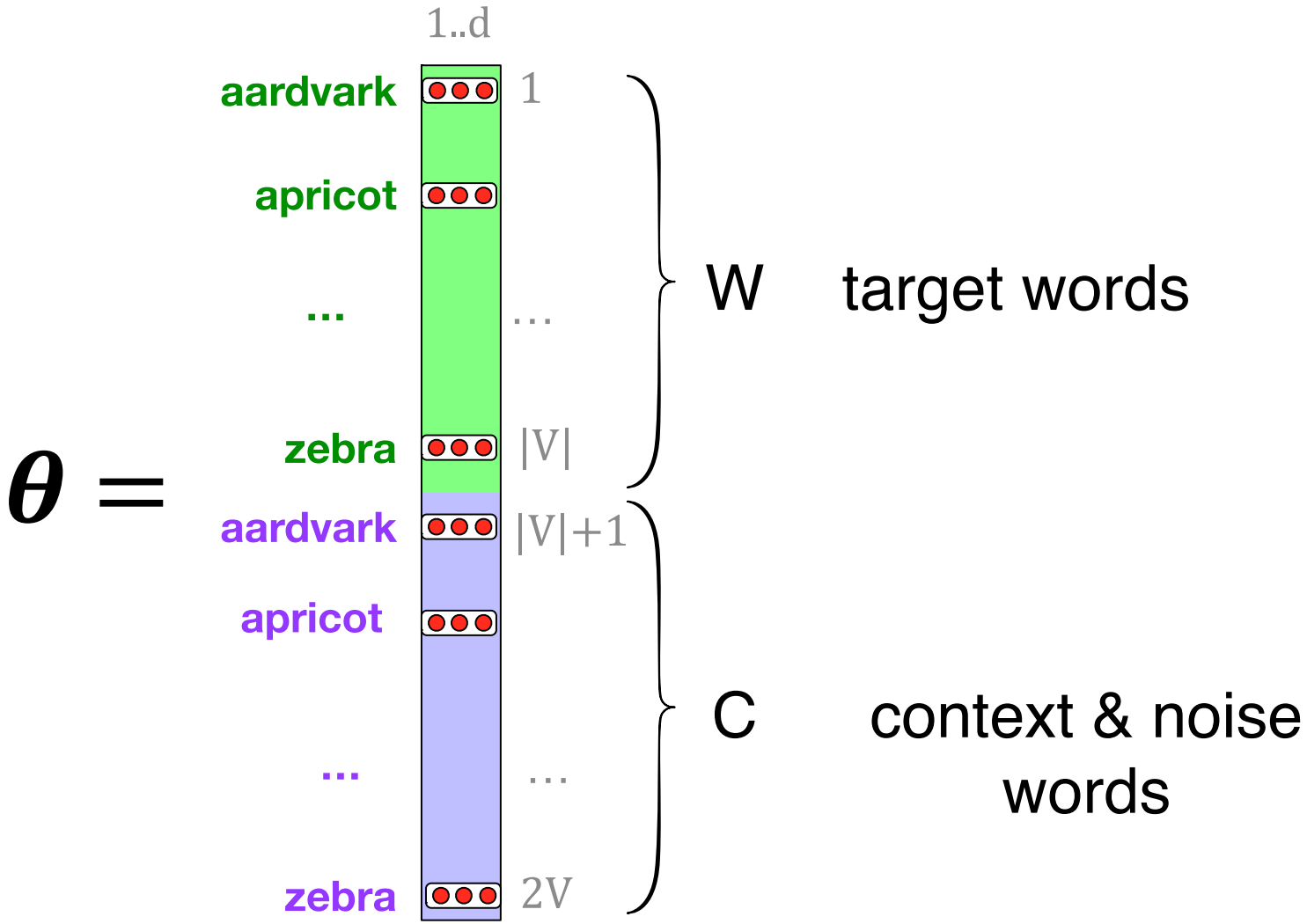
…

Et attribuer à chaque paire une probabilité :

$P(+|w, c)$

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

Embeddings dont nous aurons besoin : un ensemble pour ***w***, un ensemble pour ***c***



Skip-Gram données d'entraînements

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Pour chaque paire positive, nous allons prendre *k* exemples négatifs, échantillonnés selon leur fréquence

Skip-Gram données d'entraînements

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

La similitude est calculée à partir du produit scalaire

Rappelez-vous : deux vecteurs sont similaires s'ils ont un produit scalaire élevé

- Le cosinus n'est qu'un produit scalaire normalisé

Donc:

- Similarity(w,c) $\propto w \cdot c$

Nous devons maintenant normaliser pour obtenir une probabilité

- (Le cosinus n'est pas non plus une probabilité)

Transformer les produits scalaires en probabilités

$\text{sim(w,c)} \approx \text{w} \cdot c$

Pour le transformer en probabilité, nous utiliserons la fonction sigmoïde de la régression logistique :

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

$$\begin{aligned} P(-|w,c) &= 1 - P(+|w,c) \\ &= \sigma(-c \cdot w) = \frac{1}{1+\exp(c \cdot w)} \end{aligned}$$

Traiter des contextes plus larges avec Skip-Gram

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

Ceci est pour un contexte de 1 mot, mais nous avons plusieurs mots dans notre fenêtre de contexte. Nous supposerons l'indépendance et les multiplierons simplement leur probabilités:

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

L’objectif d’apprentissage (fonction de perte)

Supposant un ensemble d'entraînement de paires positives et négatives, et un ensemble initial d’embeddings:

L'objectif de l'apprentissage est d'ajuster ces vecteurs de mots de telle sorte que nous :

- **Maximisons** la similitude du mot cible et des mots de contexte (w , c_{pos}) venant des paires positives
- **Minimisons** la similitude des paires négatives (w , c_{neg})

Fonction de perte pour un w et c_{pos} , c_{neg1} ...c_{negk}

Maximiser la similitude de la cible avec les mots contextuels réels et minimiser la similitude de la cible avec les k mots non-voisins/négatifs échantillonnés.

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Apprendre/entraîner le classificateur

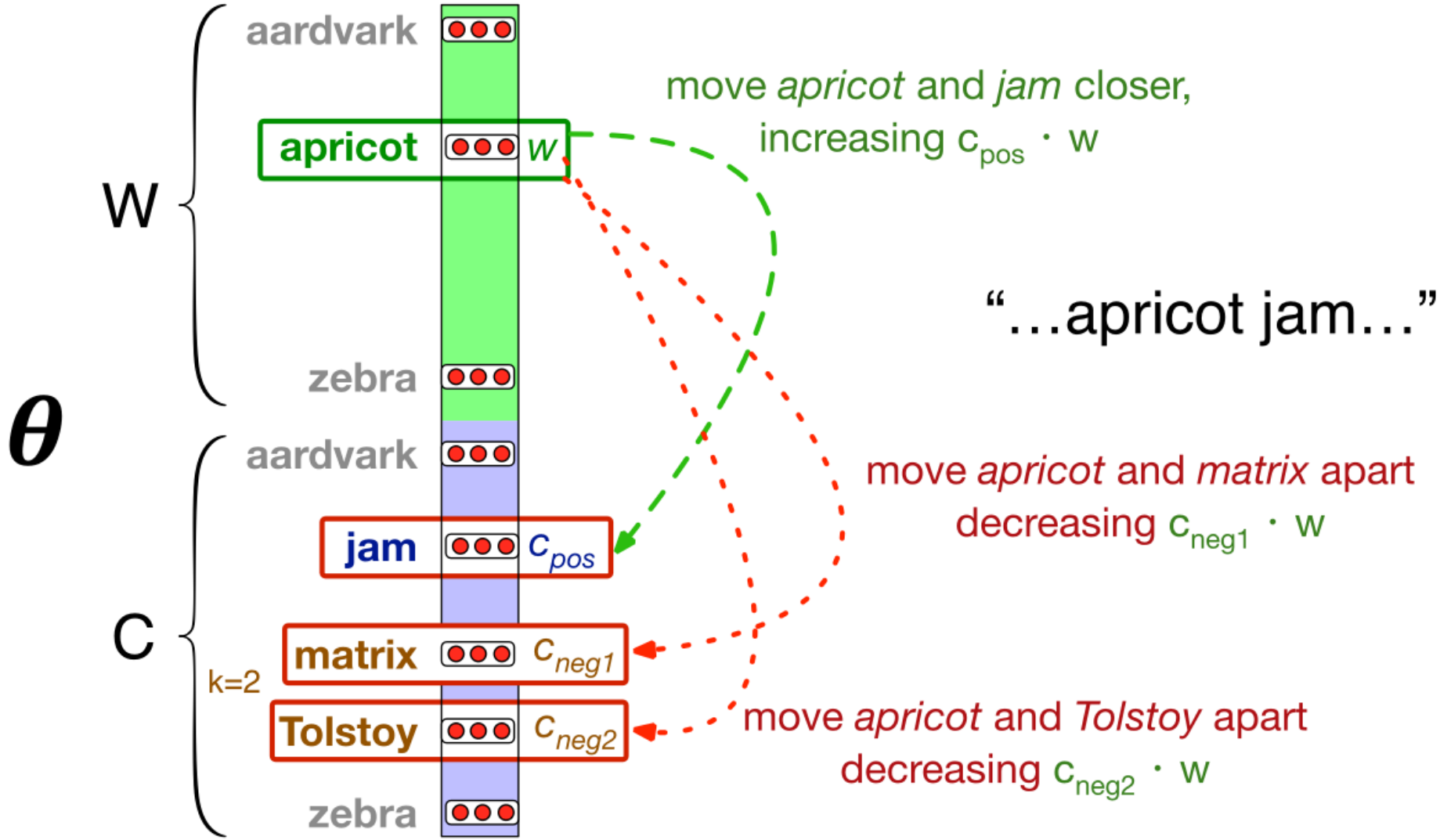
Comment apprendre? Descente de gradient stochastique!

Nous ajusterons les poids pour rendre les paires positives plus probables et les paires négatives moins probables, sur l'ensemble d’entraînement.

Descente de gradient

- À chaque pas de descente:
  - **Direction** : Nous nous déplaçons dans le sens inverse du gradient de la fonction de perte
  - **Magnitude** : nous déplaçons la valeur de ce gradient pondéré par un taux d'apprentissage η
  - Un taux d'apprentissage plus élevé signifie apprendre plus vite

Intuition derrière un pas de descente de gradient



Deux ensembles d'embeddings

Le classificateur apprend deux ensembles d'embeddings

Embeddings cibles matrice **W**

Embeddings de contextes matrice **C**

Il est courant de simplement les ajouter ensemble, représentant le mot *i* avec le vecteur $w_i + c_i$

Résumé : Comment apprendre les embeddings avec skip-gram

Commencez avec |V| vecteurs d-dimensionnels aléatoires comme embeddings initiales

Entraîner le classificateur basé sur la similitude

- Prenez un corpus et prenez des paires de mots voisinant comme exemples positifs
- Prenez des paires de mots non-voisinant comme exemples négatifs
- Entraînez le classificateur à les distinguer en ajustant lentement les poids pour améliorer les performances du classificateur
- Jetez le code classificateur et conservez les embeddings.

Les types d’embeddings

- **Il existe trois principaux types d'embeddings utilisés en TALN et ils sont créés en utilisant trois méthodes différentes :**

    1. Embeddings épars et “Term Frequency-Inverse Document Frequency (TF-IDF)

    2. **Embeddings denses et Word2Vec et l’algorithme skip-gram**

    3. Embeddings contextualisés et les LLMs

Embeddings contextualisés

Les mots sont ambigus

Un sens de mot est une représentation discrète d'un aspect du sens

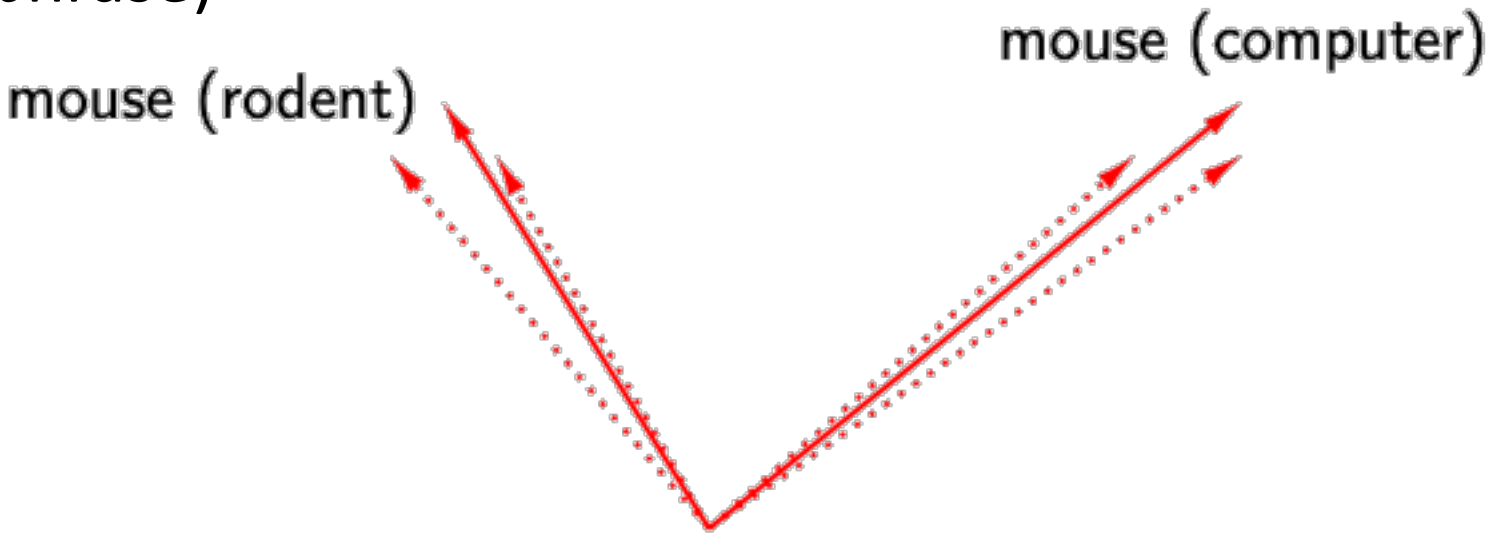
mouse[1] : .... a *mouse* controlling a computer system in 1968.
mouse[2] : .... a quiet animal like a *mouse*
bank[1] : ...a *bank* can hold the investments in a custodial account ...
bank[2] : ...as agriculture burgeons on the east *bank*, the river ...

Les embeddings contextualisés offrent un modèle continu du sens à haute dimension qui est plus précis que les sens discrets.

Embeddings contextualisés vs statiques

Les embeddings statiques représentent les types de mots (entrées de dictionnaire)

Les embeddings contextualisés représentent des instances de mots (une pour chaque fois que le mot apparaît dans n'importe quel contexte/phrase)



Embeddings contextualisés

- Intuition: Une représentation du sens d'un mot devrait être différente dans différents contextes !
- **Embedding contextualisé**: Chaque mot a un vecteur différent qui exprime des significations différentes en fonction des mots environnants
- Comment les calculer?
  - **En utilisant un modèle de langage neuronal**

Les types d’embeddings

- **Il existe trois principaux types d'embeddings utilisés en TALN et ils sont créés en utilisant trois méthodes différentes :**

    1. Embeddings épars et “Term Frequency-Inverse Document Frequency (TF-IDF)

    2. Embeddings denses et Word2Vec et l’algorithme skip-gram

    3. **Embeddings contextualisés et les LLMs**

[Pause - 15 minutes]

Utilisons les embeddings!

Mettez-vous en équipe!

Ouvrez exercices/week 4 dans votre repo du cours et commencez à écrire/exécuter du code !