09/10/2025

Natural Language Processing (NLP)

Distributional Semantics

aivancity SCHOOL FOR TECHNOLOGY, BUSINESS & SOCIETY PARIS-CACHAN

Part 1

Bag of Words, really?

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Part 1

What do words mean?

- Why is it "brother" in English and "frère" in French?
- Because "brōþēr" in Proto-Germanic and "frātrem" in Latin!
  (arbitrariness of the sign, de Saussure, 1916)
  But why does it *mean* brother?
- The meaning of a word is its **use** in the language (Ludwig Wittgenstein, 1921):
  "I was playing with my **brother** and *sister*"
  "My *mom* is feeding my **brother**"
- "brother" co-occurs with "mom" and "sister"
  like "frère" co-occurs with "maman" and "sœur"
- Homonymy: "I sit on a *chair*" vs "He is the *chair* of this session"

Part 1

How words are used?

- words are defined by their environments (the words around them)
- If A and B have almost identical environments we say that they are **synonyms** (Harris, 1954).
- define the meaning of a word by its distribution in language use: its neighboring words

Part 1

What does "ongchoi" mean?

- Suppose you see these sentences:
  - *Ongchoi* is delicious **sautéed with garlic**.
  - *Ongchoi* is superb over **rice**
  - *Ongchoi* **leaves** with **salty** sauces
- And you've also seen these:
  - …*spinach* **sautéed with garlic** over **rice**
  - *Chard* stems and **leaves** are **delicious**
  - *Collard greens* and other **salty** leafy greens
- *Ongchoi* is a leafy green like *spinach*, *chard*, or *collard greens*

Defining context (word-word matrix)

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

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

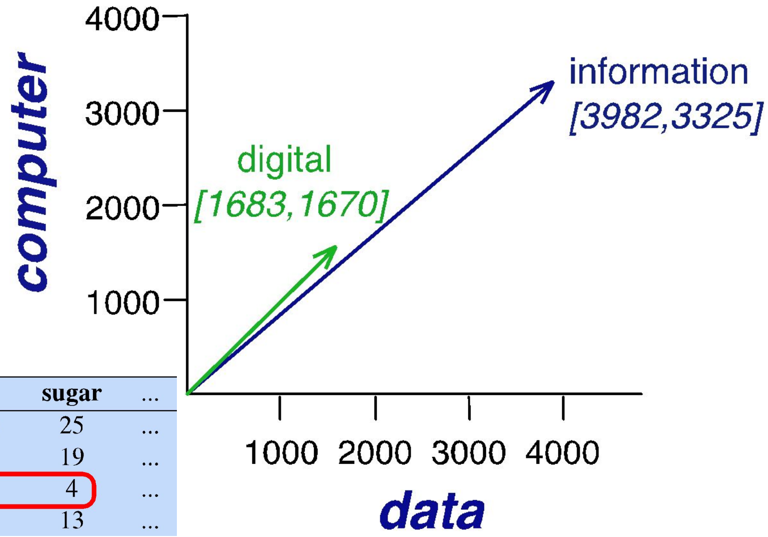
	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Part 1

Defining context (word-word matrix)

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

	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	...



Part 1

Computing word similarity: Dot product

The dot product between two vectors is a scalar:

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

The dot product tends to be high when the two vectors have large values in the same dimensions

Dot product can thus be a useful similarity metric between vectors

Part 1

Problem with raw dot-product

Dot product favors long vectors

Dot product is higher if a vector is longer (has higher values in many dimension)

Vector length (euclidean norm):

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

Frequent words (of, the, you) have long vectors (since they occur many times with other words).

So dot product overly favors frequent words

Part 1

Alternative: cosine for word similarity

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

Based on the definition of the dot product between two vectors ***a*** and ***b***

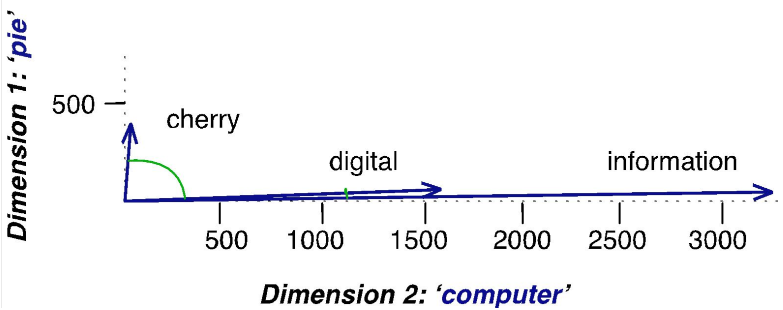
$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$

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

Part 1

Cosine examples

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



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

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

Part 1

Can we compute word similarity like this?

V vocabulary size

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...

- **Sparse** vectors (most words vever co-occur together)
- Very **high dimension**! ***V***: vocabulary size (usually 20,000 - 200,000)

Part 1

How do we reduce dimensionality?

from **V** (vocabulary size) to **d** << **V**

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...

- Generic solutions:
  - Principal Component Analysis (PCA)
  - Singular Value Decomposition (SVD) → Latent Semantic Indexing/Analysis (Deerwester et al., 1990)
- Deep learning solution: Skipgram (word2vec, Mikolov 2013)
- Output = **embedding** = **dense** vector of dimension ***d***

  → reflects semantic similarity

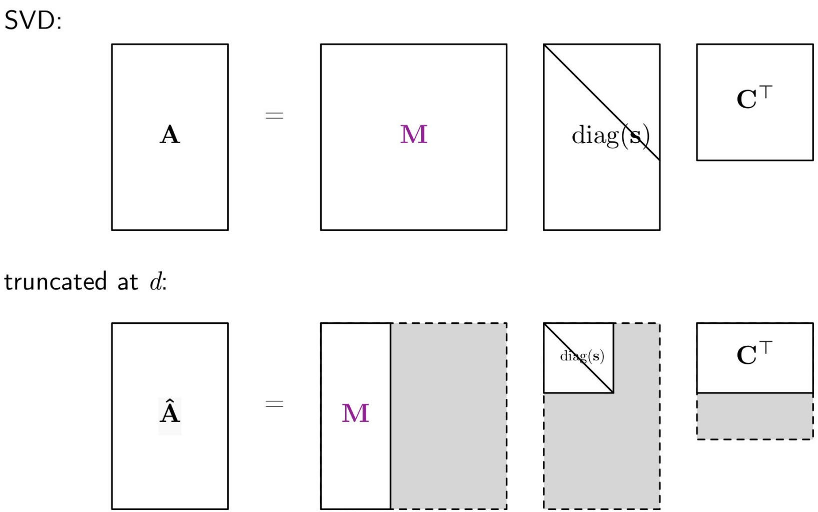
  → can be used as features by any machine learning model

Latent Semantic Indexing/Analysis

$$\underset{V \times C}{\mathbf{A}} \approx \hat{\mathbf{A}} = \underset{V \times d}{\mathbf{M}} \times \underset{d \times d}{\mathrm{diag}(\mathbf{s})} \times \underset{d \times C}{\mathbf{C}}^{\top}$$

Singular Value Decomposition (SVD)

- Usually done with word-document occurrences instead of word-word
- Actually Pointwise Mutual Information instead of raw counting
- Closely related to Skipgram (Levy and Goldberg, 2014)



Part 2

Break for questions and "appel"

Part 2

Skipgram (word2vec, Mikolov)

- Instead of **counting** how often each word ***w*** occurs near "*apricot*"
  Train a classifier on a binary prediction task: Is ***w*** likely to show up near "*apricot*"?
- We don’t actually care about this task
  But we'll take the learned classifier weights as the word embeddings
- Big idea: **self-supervision**:
  - A word ***c*** that occurs near *apricot* in the corpus cats as the gold "correct answer" for supervised learning
  - No need for human labels

Part 2

Skipgram (word2vec, Mikolov)

- Treat the target word ***w*** and a neighboring context word ***c*** as positive examples.
- Randomly sample other words in the lexicon to get negative examples
- Use logistic regression to train a classifier to distinguish those two cases
- Use the learned weights as the embeddings

Part 2

Skipgram (word2vec, Mikolov)

Assume a +/- 2 word window, given training sentence:

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

Goal: train a classifier that is given a candidate (word, context) pair

(apricot, jam)

(apricot, aardvark)

And assigns each pair a probability:

$P(+|w, c)$

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

Part 2

Turning dot products into probabilities

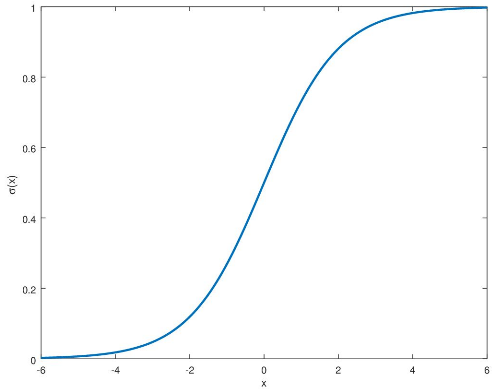
Sim(w,c) ≈ w · c

To turn this into a probability

We'll use the sigmoid from logistic regression:

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

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



Part 2

From 1 context word to full context

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

Assume all context words are **independent** → joint probability = product

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

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

log Prob: **systematic** trick for **numerical stability**

Skip-Gram Training data

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

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

- Maximize the similarity of the target word, context word pairs ***(w, c+)*** drawn from the positive data
- Minimize the similarity of the ***(w, c-)*** pairs drawn from the negative data.

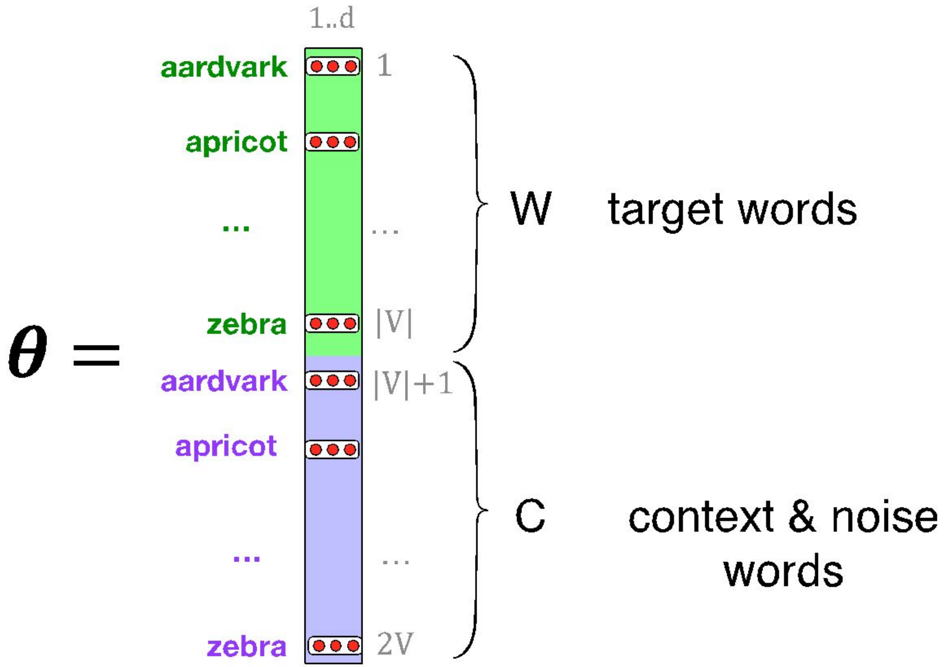
Part 2

Loss function for one w

- Maximize the similarity of the target word, context word pairs ***(w, c+)*** drawn from the positive data
- Minimize the similarity of the ***(w, c-)*** pairs drawn from the negative data.

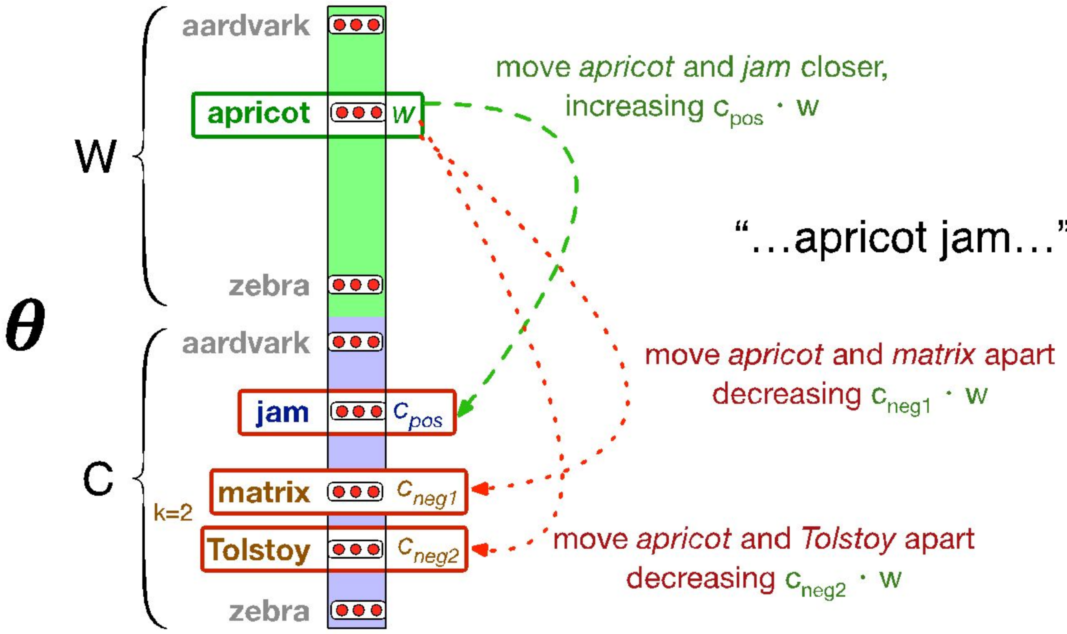
$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log\sigma(c_{pos}\cdot w)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

Learning with Stochastic gradient descent



Part 2

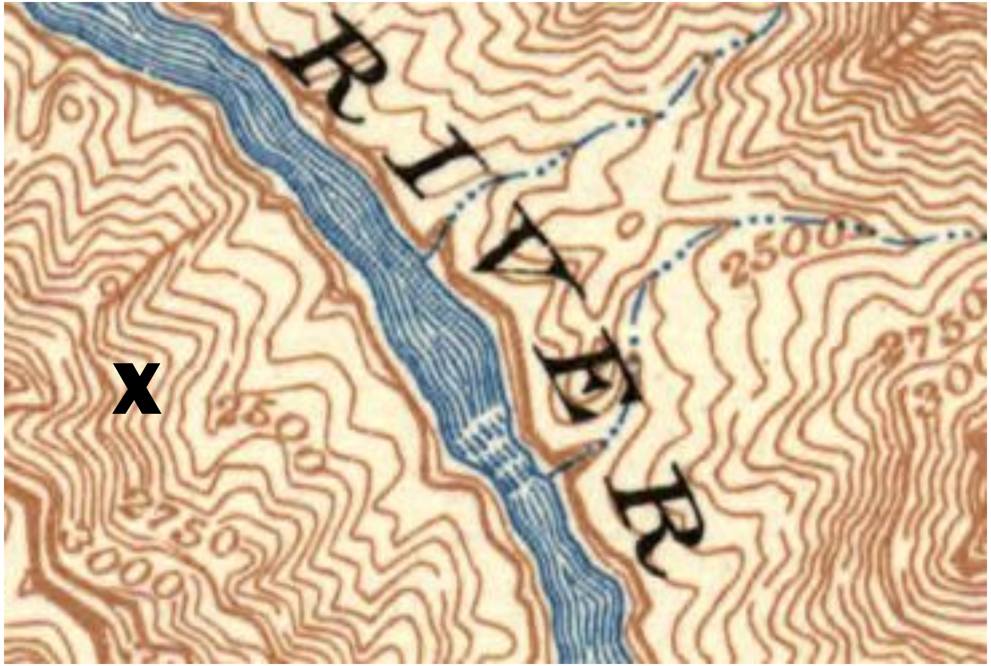
Learning with Stochastic gradient descent



Part 2

Intuition of gradient descent

How do I get to the bottom of this river canyon?



Paul

Look around me 360°

Find the direction of steepest slope down

Go that way

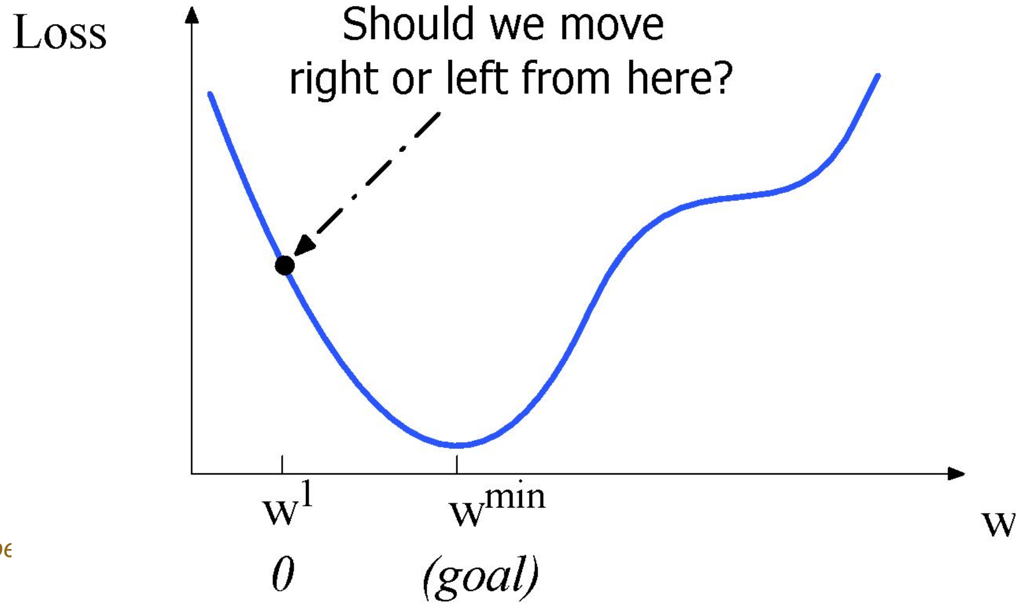
aivancity PARIS-CACHAN

Part 2

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move *w* in the reverse direction from the slope of the function

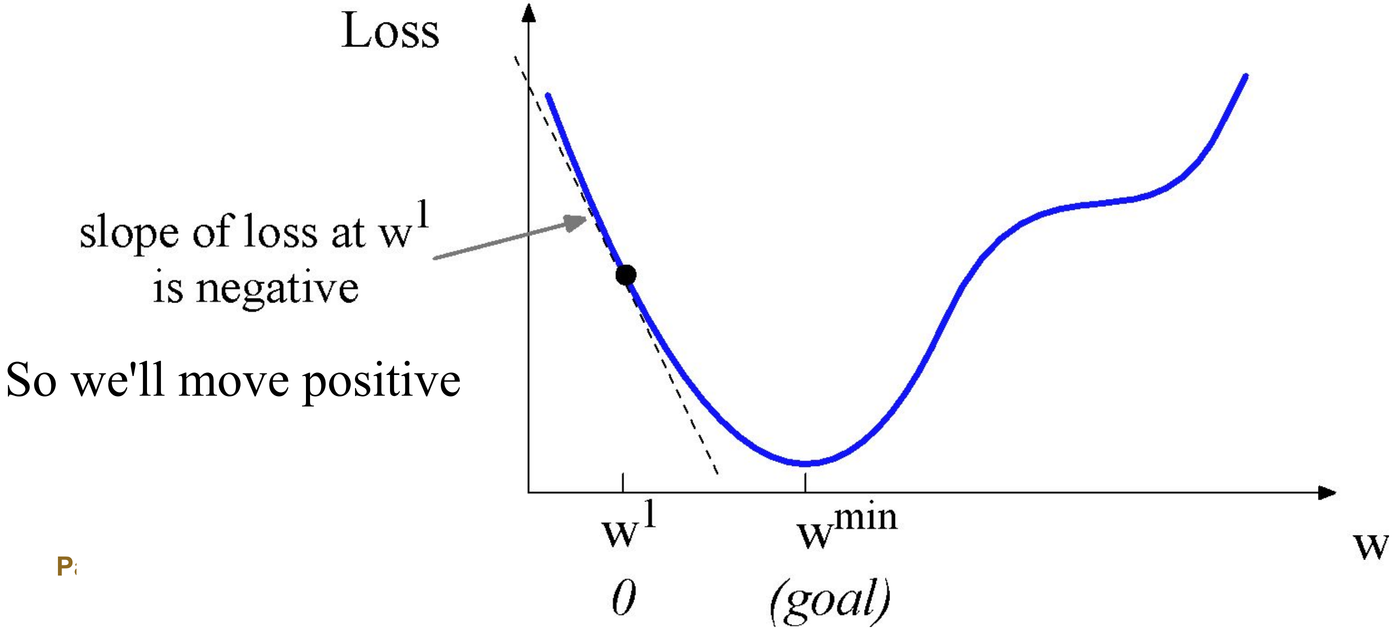


Part 2

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move *w* in the reverse direction from the slope of the function

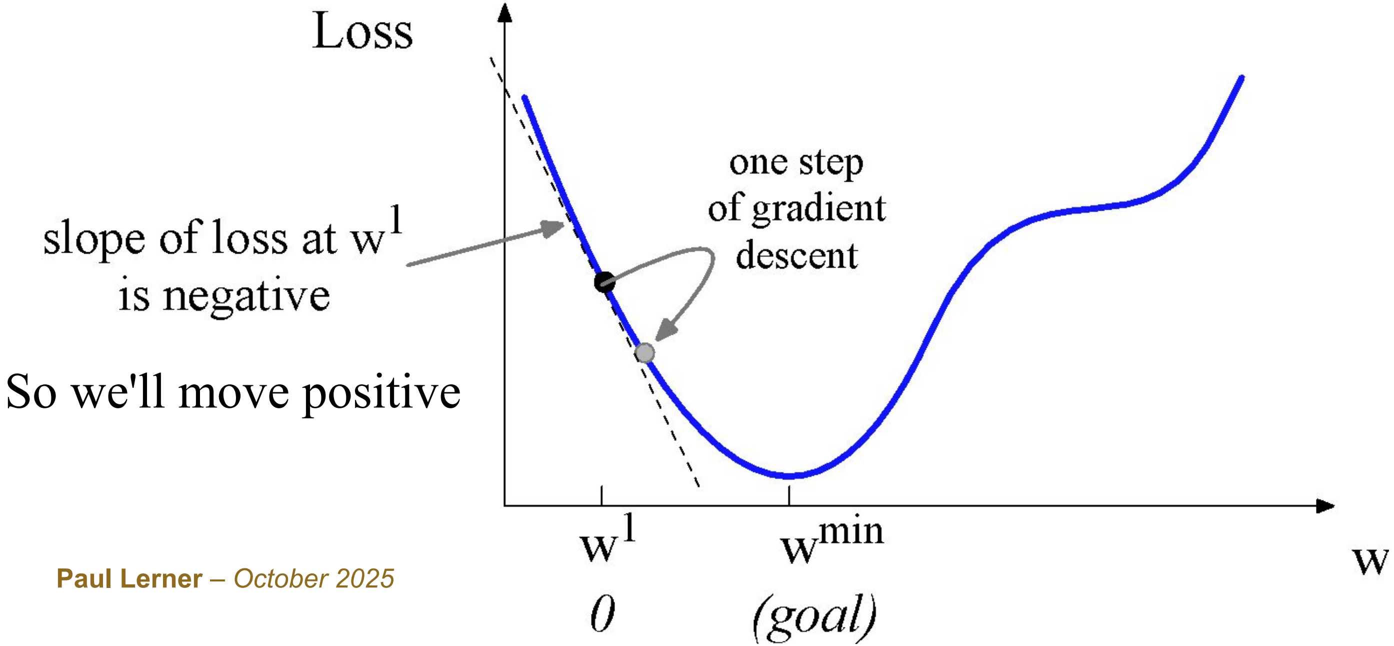


aivancity PARIS-CACHAN

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move *w* in the reverse direction from the slope of the function



Stochastic gradient descent (SGD) reminder

- Learning rate $\alpha \in \mathbb{R}, \alpha > 0$
- Randomly initialize $\theta^{(0)}$
- Iteratively get better estimate with:

Next estimate

Learning rate (step size)

$$\theta^{(i+1)} = \theta^{(i)} - \alpha * \boxed{\frac{\partial L}{\partial \theta}(\theta^{(i)})}$$

Previous Estimate

Gradient is:

- the vector of partial derivatives of the parameters with respect to the loss function
- A linear approximation of the loss function at $\theta^{(i)}$

$$\frac{\partial L}{\partial \theta}(\theta^{(i)}) = \begin{bmatrix} \frac{\partial L}{\partial \theta_1^{(i)}} \\ \frac{\partial L}{\partial \theta_2^{(i)}} \\ \vdots \\ \frac{\partial L}{\partial \theta_n^{(i)}} \end{bmatrix}$$

Part 2

The derivatives of the loss function

$$L_{CE} = -\left[\log\sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]$$

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos}\cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg}\cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos}\cdot w) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{\mathrm{neg}_i}\cdot w)]c_{neg_i}$$

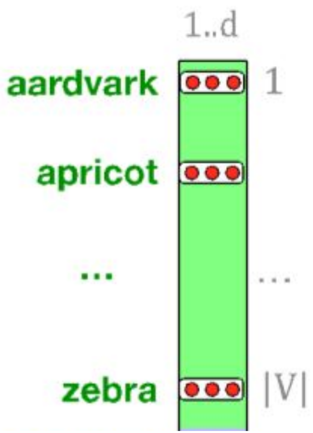
Part 2

Stochastic gradient descent update

$$
\begin{aligned}
c_{pos}^{t+1} &= c_{pos}^{t} - \eta[\sigma(c_{pos}^{t} \cdot w^{t}) - 1]w^{t} \\
c_{neg}^{t+1} &= c_{neg}^{t} - \eta[\sigma(c_{neg}^{t} \cdot w^{t})]w^{t} \\
w^{t+1} &= w^{t} - \eta\left[[\sigma(c_{pos} \cdot w^{t}) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{\mathrm{neg}_i} \cdot w^{t})]c_{neg_i}\right]
\end{aligned}
$$

Part 2

Embedding = lookup table or linear layer?



lookup table

One-hot encoding

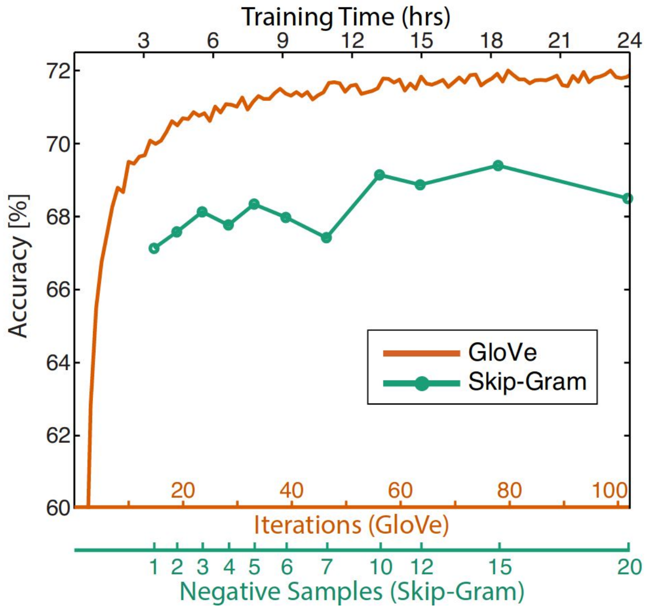
Standard basis of $\mathbb{R}^n$: $e_1 = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}, e_2 = \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix}, \ldots, e_n = \begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix}$

$|\mathcal{V}| = n$:

$$\text{features}(v_i) = We_i \in \mathbb{R}^k = \text{ith column of } W$$

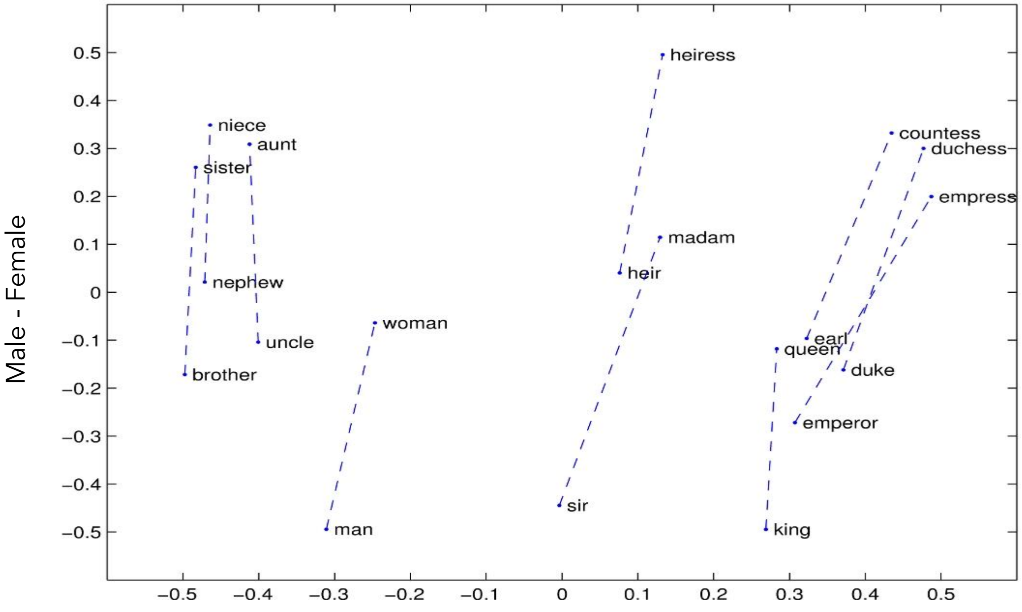
As always, hyperparameters

- Vocabulary size ***V***
- Context window ***C***
- Number of negative examples ***k***
- Embedding dimension ***d***
- The usual:
  - learning rate etc.
- → **Empirical** evaluation!



Part 2

What now?



(2-d projections of embedding space) https://nlp.stanford.edu/projects/glove/

Part 2

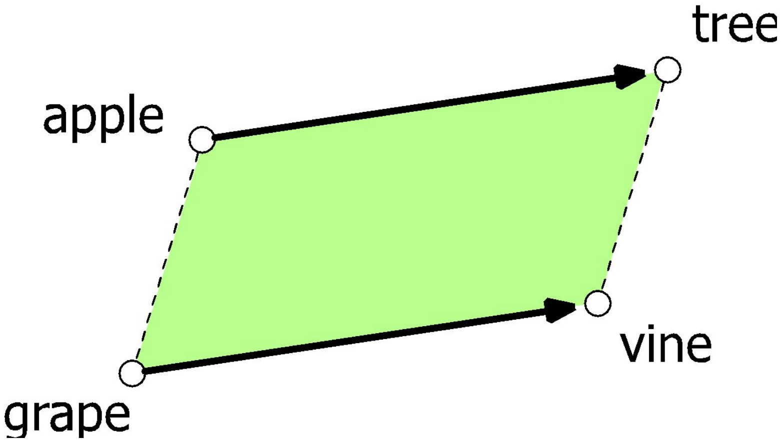
Intrinsic evaluation

- Do (cosine) similarities of pairs of words' vectors correlate with judgments of similarity by humans?
- TOEFL-like synonym tests, e.g., rug → {sofa ✗ ottoman ✗ carpet ✓ hallway ✗}
- analogies:
  - syntactic
  - semantic

Part 2

Analogical relations

- The classic parallelogram model of analogical reasoning (Rumelhart and Abrahamson 1973)
- To solve: "apple is to tree as grape is to ______"
- Add tree – apple to grape to get vine
- Syntactic analogies, e.g., “walking is to walked as eating is to what?” Solved via:



$$\max_{v \in \mathcal{V}} \cos\left(\mathbf{v}_v, -\mathbf{v}_{walking} + \mathbf{v}_{walked} + \mathbf{v}_{eating}\right)$$

Part 2

Quantitatively

WS353 (WORDSIM) [13]			MEN (WORDSIM) [4]		
Representation		Corr.	Representation		Corr.
SVD	(k=5)	0.691	SVD	(k=1)	0.735
SPPMI	(k=15)	0.687	SVD	(k=5)	0.734
SPPMI	(k=5)	0.670	SPPMI	(k=5)	0.721
SGNS	(k=15)	0.666	SPPMI	(k=15)	0.719
SVD	(k=15)	0.661	SGNS	(k=15)	0.716
SVD	(k=1)	0.652	SGNS	(k=5)	0.708
SGNS	(k=5)	0.644	SVD	(k=15)	0.694
SGNS	(k=1)	0.633	SGNS	(k=1)	0.690
SPPMI	(k=1)	0.605	SPPMI	(k=1)	0.688

Spearman's ρ k is the number of "negative" samples

MEN : 3000 items

a	b	label
sun	sunlight	50.0
automobile	car	50.0
river	water	49.0
stairs	staircase	49.0
morning	sunrise	49.0
...	...	...
feathers	truck	1.0
festival	whiskers	1.0
muscle	tulip	1.0
bikini	pizza	1.0
bakery	zebra	0.0

Part 2

Quantitatively

MIXED ANALOGIES [20]			SYNT. ANALOGIES [22]		
Representation		Acc.	Representation		Acc.
SPPMI	(k=1)	0.655	SGNS	(k=15)	0.627
SPPMI	(k=5)	0.644	SGNS	(k=5)	0.619
SGNS	(k=15)	0.619	SGNS	(k=1)	0.59
SGNS	(k=5)	0.616	SPPMI	(k=5)	0.466
SPPMI	(k=15)	0.571	SVD	(k=1)	0.448
SVD	(k=1)	0.567	SPPMI	(k=1)	0.445
SGNS	(k=1)	0.540	SPPMI	(k=15)	0.353
SVD	(k=5)	0.472	SVD	(k=5)	0.337
SVD	(k=15)	0.341	SVD	(k=15)	0.208

Word Pair 1		Word Pair 2	
Athens	Greece	Oslo	Norway
Astana	Kazakhstan	Harare	Zimbabwe
Angola	kwanza	Iran	rial
Chicago	Illinois	Stockton	California
brother	sister	grandson	granddaughter

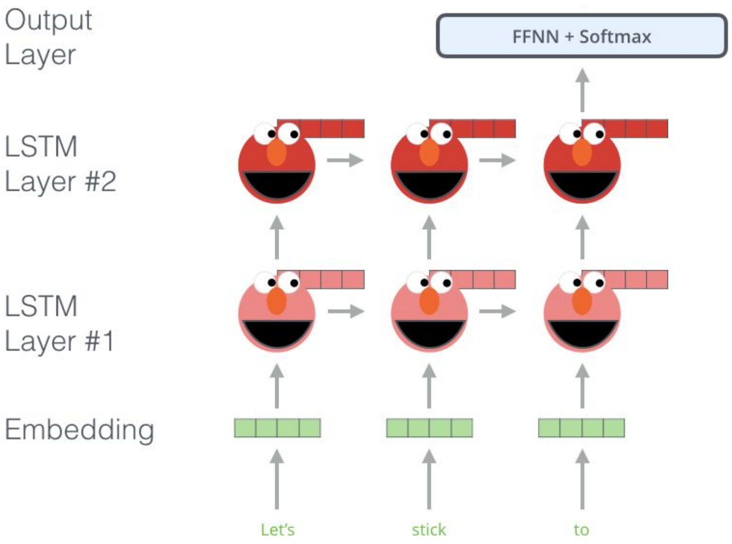
Example
good:better rough:___
good:best rough:___
better:best rougher:___
year:years law:___
city:city's bank:___
see:saw return:___
see:sees return:___
saw:sees returned:___

k is the number of “negative” samples

Part 2

Extrinsic evaluation

- Embeddings are the first brick of any more complex models
- Embeddings can be initialized with Skip-gram: **pretraining**/transfer learning
- either keep them **frozen** or **fine-tune** them



Named Entity Recognition with pretrained embeddings

Washington is the capital of the USA. It hosts the White House.

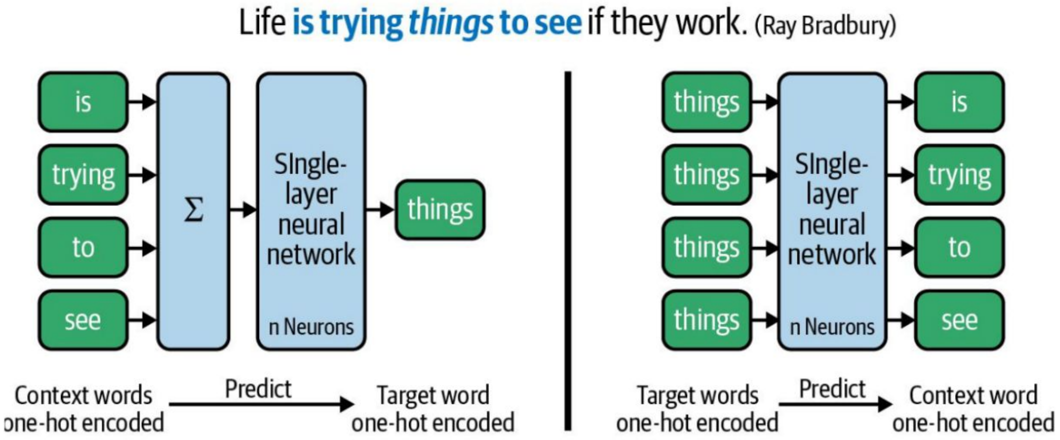
Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

F1 score

Part 2

Alternatives to Skipgram: continuous bag of words (CBOW)

instead of predicting context from word, predict word from context (much like a language model)



Part 2

Alternatives to Skipgram: continuous bag of words (CBOW)

"bag of words" because does not model word order, puts all words in the same "bag"

$$\overline{\boldsymbol{v}}_m = \frac{1}{2h}\sum_{n=1}^{h} \boldsymbol{v}_{w_{m+n}} + \boldsymbol{v}_{w_{m-n}}$$

average of embeddings for words in the immediate neighborhood (**m−h**, ... , **m+h**)

Part 2

Alternatives to Skipgram: continuous bag of words (CBOW)

$$
\begin{aligned}
\log \mathrm{p}(\boldsymbol{w}) &\approx \sum_{m=1}^{M} \log \mathrm{p}(w_m \mid w_{m-h}, w_{m-h+1}, \ldots, w_{m+h-1}, w_{m+h}) \\
&= \sum_{m=1}^{M} \log \frac{\exp\left(\boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m\right)}{\sum_{j=1}^{V} \exp\left(\boldsymbol{u}_j \cdot \overline{\boldsymbol{v}}_m\right)} \\
&= \sum_{m=1}^{M} \boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m - \log \sum_{j=1}^{V} \exp\left(\boldsymbol{u}_j \cdot \overline{\boldsymbol{v}}_m\right).
\end{aligned}
$$

Empirical comparison

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Spearman’s ρ

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7

Word analogy

Part 2

Alternatives to Skipgram: GloVe

studies ratio of co-occurrence instead of co-occurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

$$\min_{\boldsymbol{u},\boldsymbol{v},b,\tilde{b}} \quad \sum_{j=1}^{V} \sum_{j \in C} f(M_{ij}) \left(\widehat{\log M_{ij}} - \underbrace{\log M_{ij}}_{\boxed{\log \text{count}(i,j)}} \right)^2$$

$$\text{s.t.} \quad \widehat{\log M_{ij}} = \boldsymbol{u}_i \cdot \boldsymbol{v}_j + b_i + \tilde{b}_j,$$

Part 2

Empirical comparison

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Spearman's ρ

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7

Word analogy

Part 2

Skipgram with character n-grams (fastText)

- brother: `bro, rot, oth, the, her` (trigrams)
- brothers: `bro, rot, oth, the, her, ers` : almost the same!
- also enables to model Out-of-Vocabulary words (OOV), e.g. brotha
- rough way of modelling **morphology**: relation between words
- same objective as skipgram: $\log\left(1+e^{-s(w_t,\, w_c)}\right) + \sum_{n \in \mathcal{N}_{t,c}} \log\left(1+e^{s(w_t,\, n)}\right)$
- simply redefine similarity: sum over all n-grams of the word

$$s(w,c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$$

Part 2

Empirical comparison

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

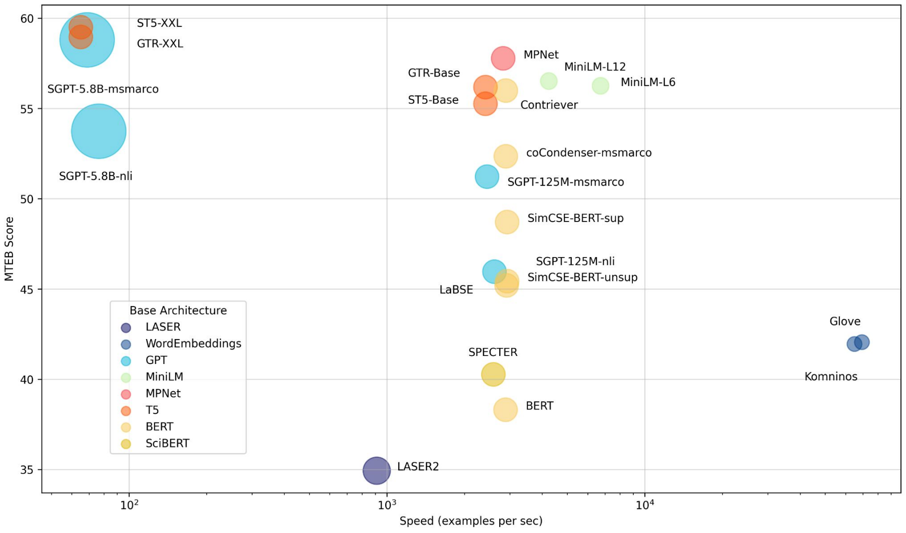
Spearman's ρ

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

Word analogy

Welcome LLMs, exit Embeddings?

- Large Language Models are effective but not so efficient
- Embeddings are very lightweight, relevant for many industrial applications
- fastText: efficient implementation
- LLMs build on similar hypothesis and methods as Embeddings

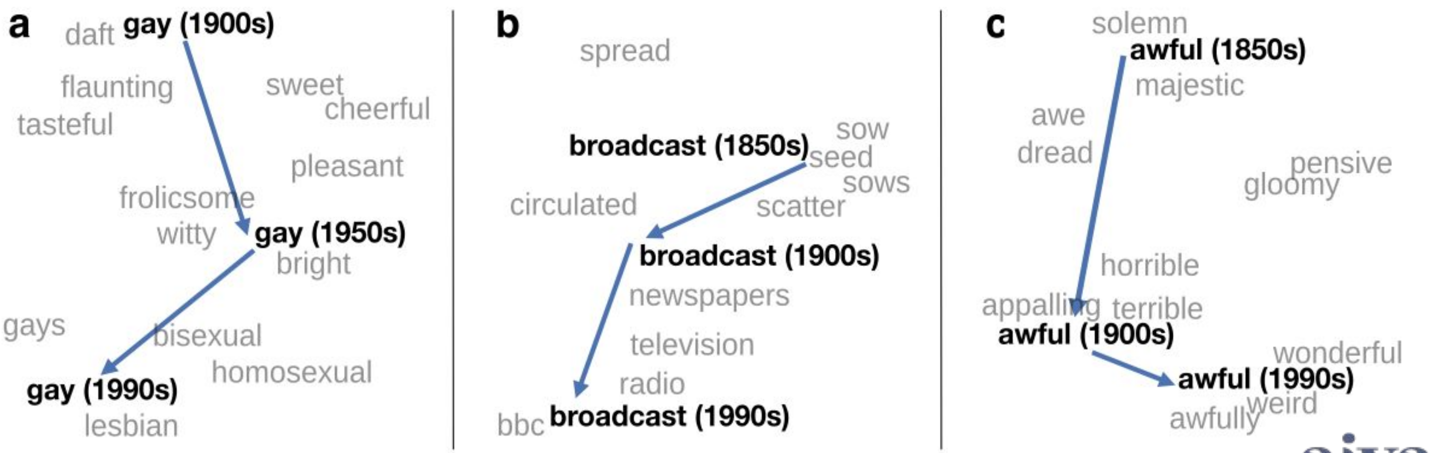


In Summary

- Meaning of a word is its use in the language: distributional semantics
- Skip-gram (word2vec): compute embeddings of words by predicting their context (**self-supervised learning**)
- Use as building block (**pre-training**) or solve analogies or measure word semantic similarity

Limitations

- Cannot model homonymy: chair [furniture] vs chair [person] has only one embedding "chair"
- Meaning changes through time/domain…



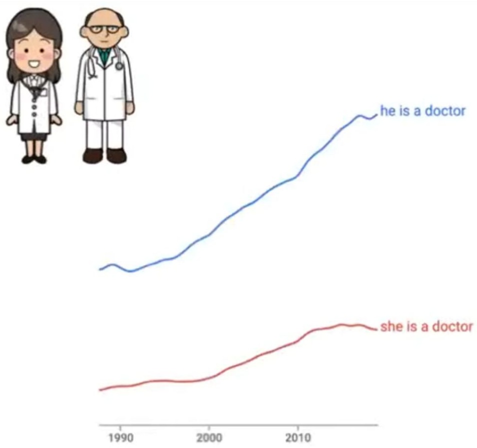
Part 2

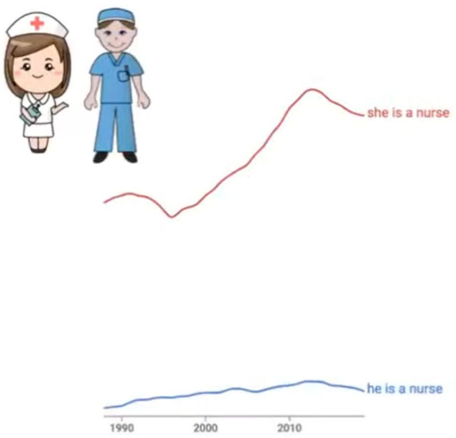
Embeddings reflect cultural bias!

- Statistical patterns in text reflect both **intrinsic meaning** and **extrinsic use**
- Ask “father : doctor :: mother : x”

  x = nurse
- Ask “man : computer programmer :: woman : x”

  x = homemaker





Part 2

Next class: Language Modeling

$p(x \mid \text{START})$		$p(x \mid \text{START I})$		$p(x \mid \cdots \text{went})$		$p(x \mid \cdots \text{to})$		$p(x \mid \cdots \text{the})$		$p(x \mid \cdots \text{park})$		$p(x \mid \text{START I went to the park.})$	
The	3 %	think	11 %	to	35 %	the	29 %	bathroom	3 %	and	14 %	I	21 %
When	2,5 %	was	5 %	back	8 %	a	9 %	doctor	2%	with	9	It	6
They	2 %	went	2 %	into	5 %	see	5 %	hospital	2 %	,	8 %	The	3 %
…	…	am	1 %	through	4 %	my	3 %	store	1,5 %	to	7 %	There	3 %
I	1 %	will	1 %	out	3 %	bed	2 %	…	…	…	…	…	…
…	…	like	0,5 %	on	2 %	school	1 %	park	0,5 %	.	6 %	STOP	1 %
Banana	0,1 %	…	…	…	…%	…	…	…	…	…	…	…	…

Language Model

START	I	went	to	the	park	.	STOP

Part 0

Acknowledgements

This class directly builds upon:

- **Jurafsky, D., & Martin, J. H.** (2024). Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rdéd.).
- **Eisenstein, J.** (2019). Natural Language Processing. 587.
- **Yejin Choi**. (Winter 2024). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Noah Smith**. (Winter 2023). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Benoît Sagot**. (2023-2024). *Apprendre les langues aux machines* (Collège de France)
- **Chris Manning**. (Spring 2024). Stanford CS224N: Natural Language Processing with Deep Learning
- Classes where I was/am Teacher Assistant:
    - **Christopher Kermorvant**. Machine Learning for Natural Language Processing (ENSAE)
    - **François Landes** and **Kim Gerdes**. Introduction to Machine Learning and NLP (Paris-Saclay)

Also inspired by:

- My PhD thesis: *Répondre aux questions visuelles à propos d'entités nommées* (2023)
- **Noah Smith** (2023): Introduction to Sequence Models (LxMLS)
- **Kyunghyun Cho**: Transformers and Large Pretrained Models (LxMLS 2023), Neural Machine Translation (ALPS 2021)
- My former PhD advisors **Olivier Ferret** and **Camille Guinaudeau** and postdoc advisor **François Yvon**
- My former colleagues at LISN

aivancity

PARIS-CACHAN

advancing education
in artificial intelligence