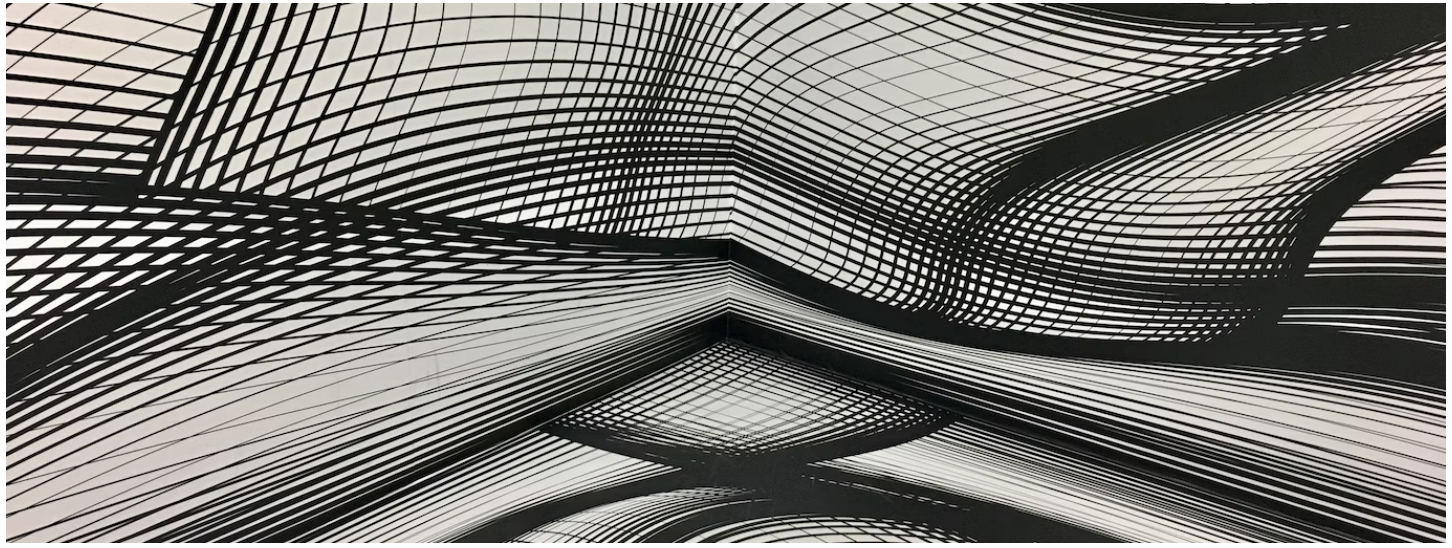
LANGUAGE & AI: REPRESENTATION



Chris Emmery
Department of Cognitive Science & AI
Tilburg University

@cmry • @_cmry • @cmry • cmry.github.io

♻️ RECAP PREVIOUS LECTURES

- We looked at some interesting language **features**.
- We discussed how to convert text into **vectors**, using a discrete n-gram representation.
- Using this vector space, we looked at distance and how it might be employed for **similarity**.
- We also looked at how we might infer **probabilistic** models from data, both to **model**, and **predict** from, the data.

MEANING AS COUNTS

💬 HUMAN LANGUAGE & AI

- Language / text is distinctive: unique driver behind our evolutionary development and dominance.
- Any instruction to an AI problem can be phrased as a language problem, but explaining what is happening is also a problem of language!
- Language is social, contextualized, and has deliberate meaning. Intentional and evolving communication.

🧠 MORE FEATURES

- Discrete, symbolic, and categorical (🎄 = tree, 🏃 = run).
- Symbols hold across media: speech, gestures, text.

> ***Cognitive Science**: does our brain contain symbolic representations, or is it just continuous neural activations over **sequences**?*

REPRESENTING LANGUAGE AS A SEQUENCE

We have mostly looked at condensed representations (BoW, one vector per input), but ignored sequential properties:

$$V = \text{(is it language or sequential)} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

🚀 DISTRIBUTIONAL SIMILARITY: REPRESENTING CONTEXT AS A VECTOR

"*You shall know a word by the company it keeps*" (Firth, 1957)

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	...

Works with $tf \cdot idf$ and Cosine Similarity!

BEYOND TF · IDF: POSITIVE POINTWISE MUTUAL INFORMATION

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \qquad \text{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

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

👀 PPMI SINGLE EXAMPLE

$$P(w = \text{information}, c = \text{data}) = \frac{3982}{11716} = .3399$$

$$P(w = \text{information}) = \frac{7703}{11716} = .6575$$

$$P(c = \text{data}) = \frac{5673}{11716} = .4842$$

$$\text{ppmi}(\text{information}, \text{data}) = \log_2 \frac{.3399}{.6575 \times .4842} = .0944$$

!! Rare words get high PPMI, offset:

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

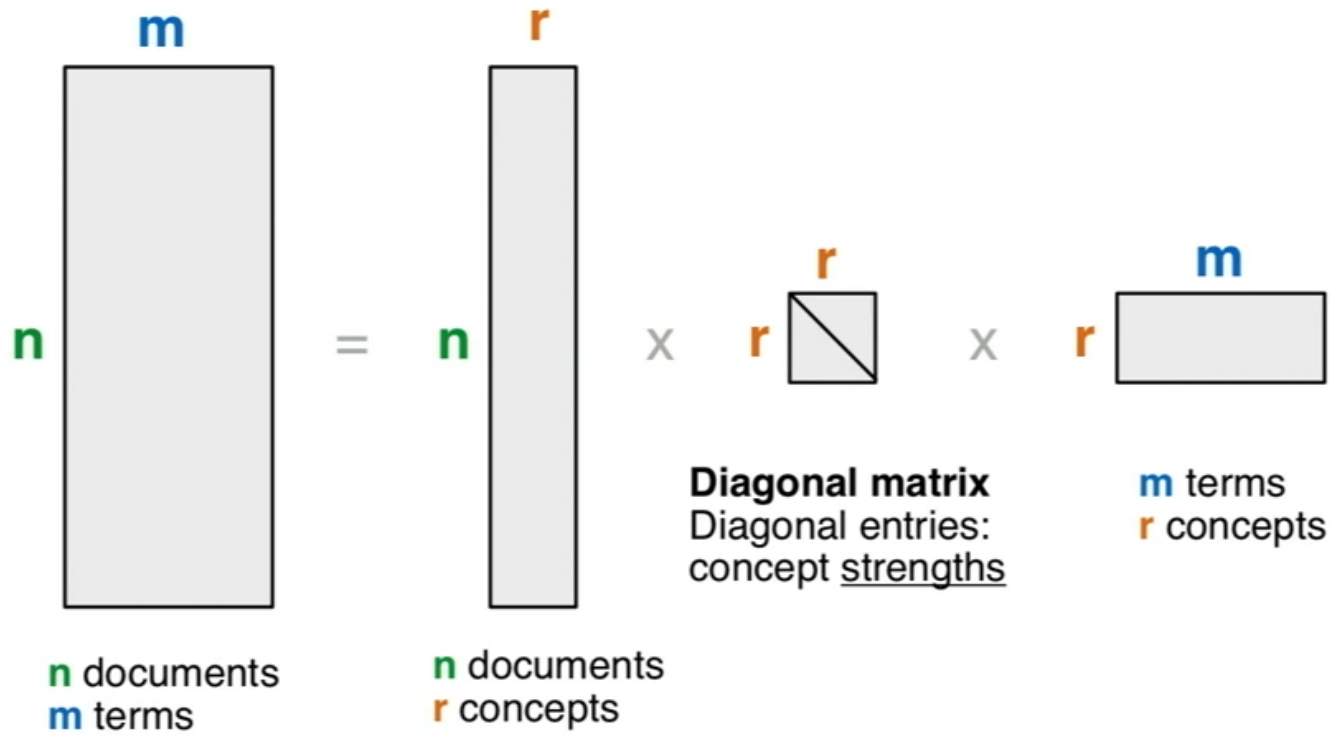
 INTRODUCING REPRESENTATIONAL DENSITY

- The vectors so far are very **sparse** and large ($|V|$).
- In turns out that for most language problems, **dense** vectors work much better.
- Why? Less weights, easier to optimize, dimensionality better represented.

SINGULAR VALUE DECOMPOSITION (SVD)

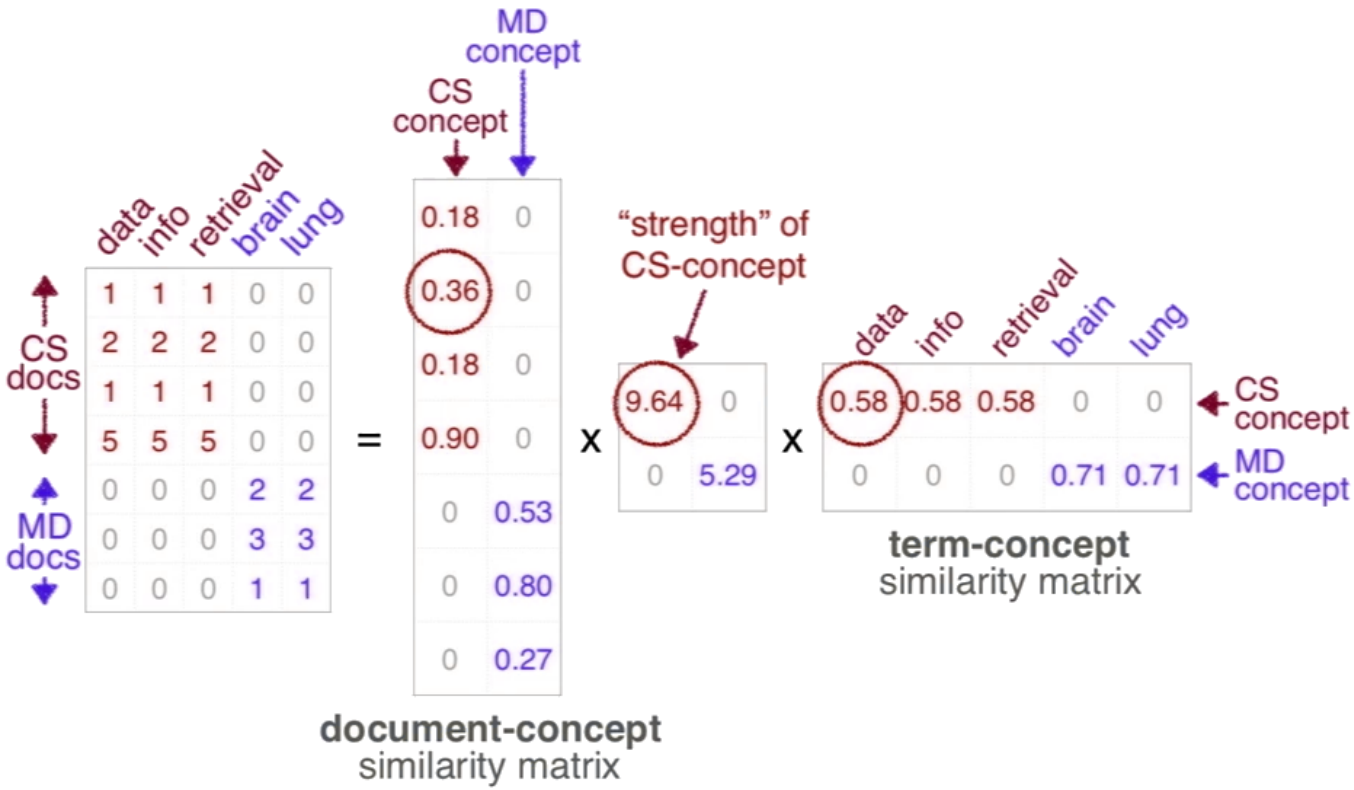
→

LATENT SEMANTIC ANALYSIS (LSA)

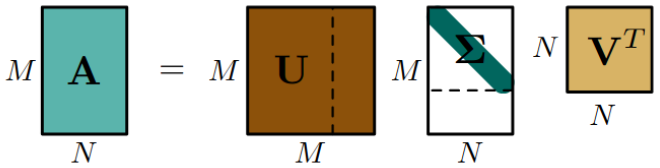


Images from the Polo Club of Data Science.

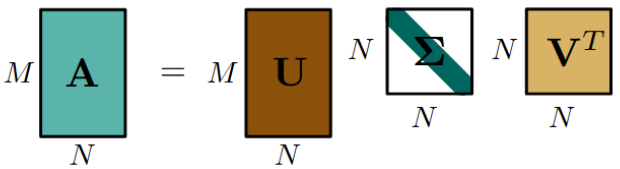
LSA INTUITIONS



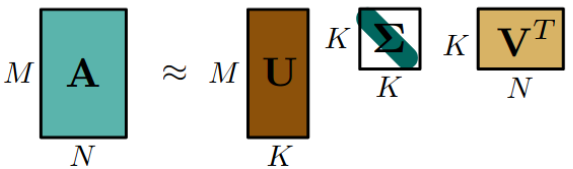
✂️ LOWER-RANK APPROXIMATIONS



(a) full SVD



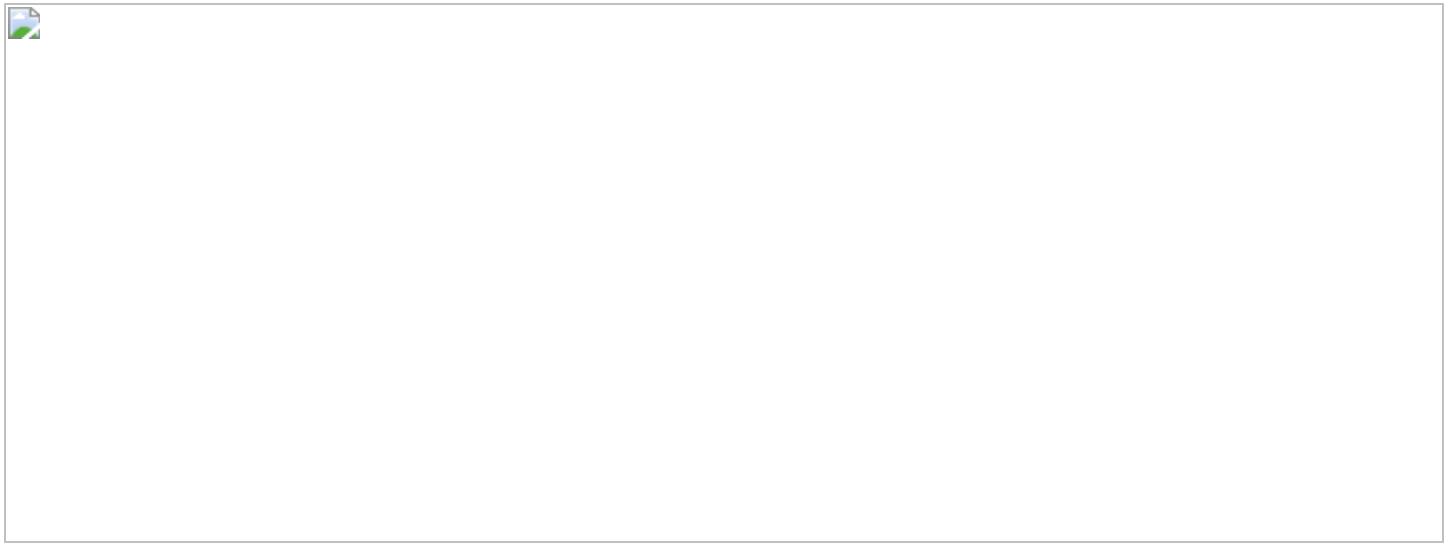
(b) reduced SVD



(c) truncated SVD

Image from Susanne Suter (2013)

🕶 PREDICTING WORD MEANING

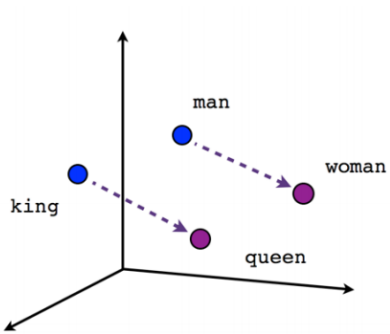


LEXICAL SEMANTICS

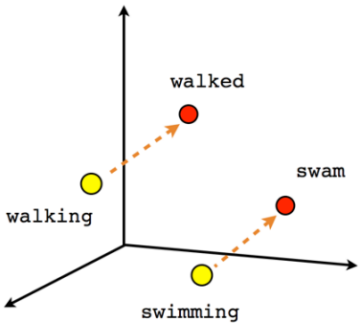
- BoW models ascribe atomic meaning: `LIFE`
- We want to have a broader sense of meaning:
  - `cat` similar to `dog` (categories)
  - `cold` is opposite of `hot` (antonyms)
  - `fake`/`copy` different connotations
  - capture related sequences: `buy`, `sell`, `pay`
  - `mouse` = rodent and controller (polysemy)
  - `couch` = sofa (synonymy)
- Useful for meaning-related tasks (Q&A).
- Have a sense of similarity, relatedness, and semantic role.

🏹 VECTOR SEMANTICS

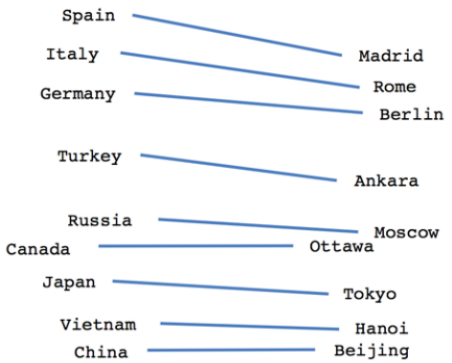
‘Embed’ words; map from one structure to the other.



Male-Female



Verb tense



Country-Capital

Count! *vs.* *Predict!*

 WORD2VEC

- Algorithm suite: Continuous Bag-of-Words (CBOW, predict word given context), and (SGNS, Skip-Gram with Negative Sampling) — we focus on the latter.
- Fast, efficient to train, share embeddings (gensim).
- Static (different from transformers).

Task: "*Is word w likely to show up near apricot?*"

Target: Self-supervised, actual word is the gold standard.

→ only use embedding encoding

Mikolov et al. (2013a), Mikolov et al. 2013b.

✨ DIFFERENCE WITH 'FANCY' NLMS

- Binary classification, simple Logistic Regression.
- How? Skip-grams:
  - Treat the target word and a neighboring context word as positive examples.
  - Randomly sample other words in the lexicon to get negative samples.
  - Use logistic regression to train a classifier to distinguish those two cases.
  - Use the learned weights as the embeddings.

📈 CLASSIFIER

...	lemon,	a	tablespoon	of	apricot	jam,	a	pinch	...
		[	c1		c2	w	c3	c4	]

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

w	c_{neg}*
apricot	aardvark
apricot	my
apricot	where
apricot	coaxial

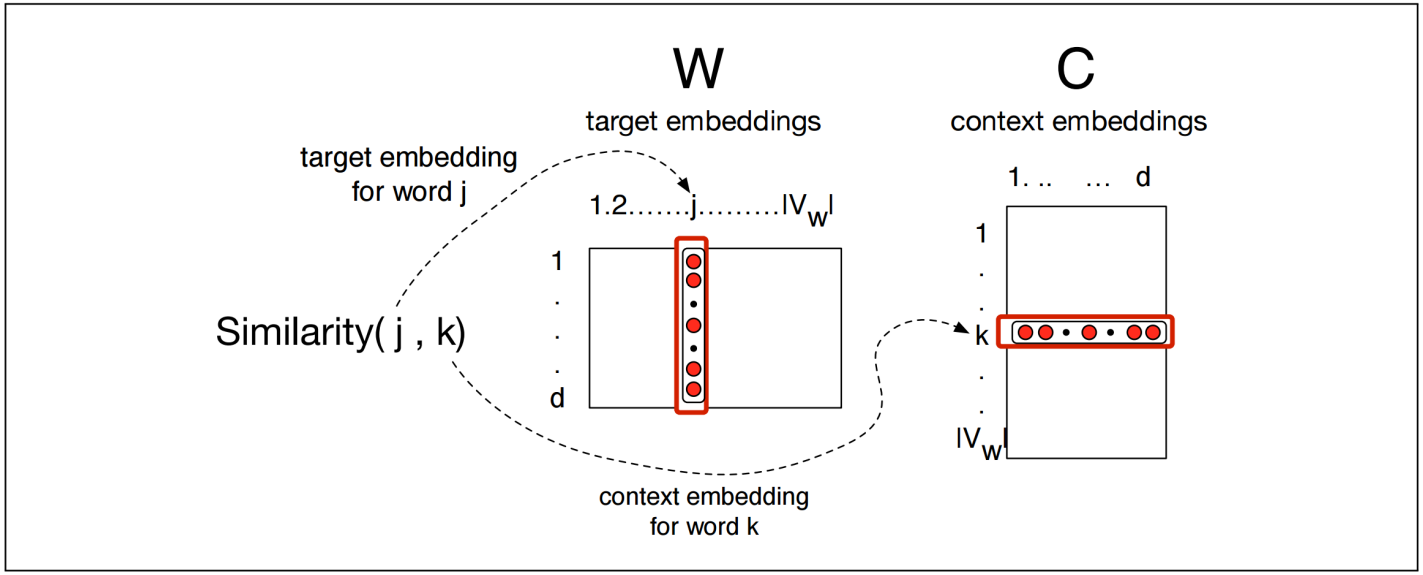
$P(+|w,c)$ $\quad$ $P(-|w,c) = 1 - P(+|w,c)$ $\quad$ $\text{Similarity}(w,c) \approx \mathbf{c} \cdot \mathbf{w}$

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

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

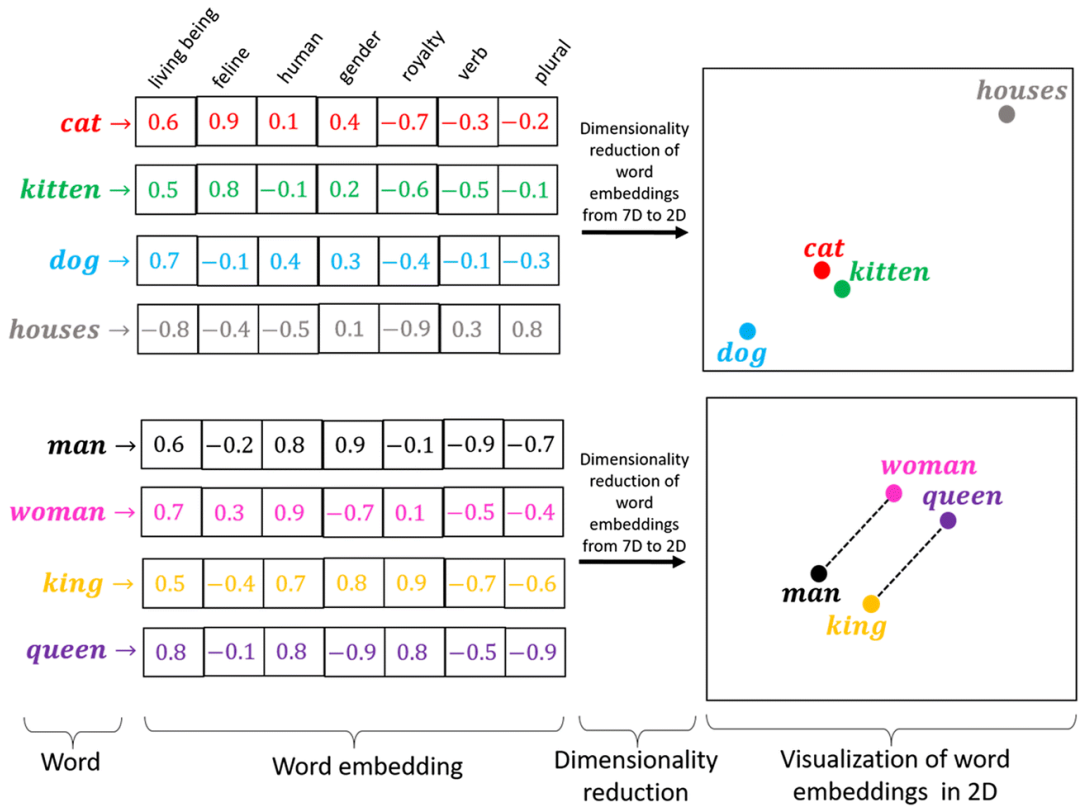
Maximize dot product between c_{pos}, minimize c_{neg}.
**Negative examples are sampled using weighted $\alpha = 0.75$ unigram count, similar to PPMI.*

EMBEDDING REPRESENTATION



Either we: represent word i (above j) with the vector $\boldsymbol{w}_i + \boldsymbol{c}i$. Or: yeet $\boldsymbol{C}$, and represent word i by the vector $\boldsymbol{w}_i$. Windows size L important to tune.

👑 EMBEDDING PROPERTIES



Source image unknown.

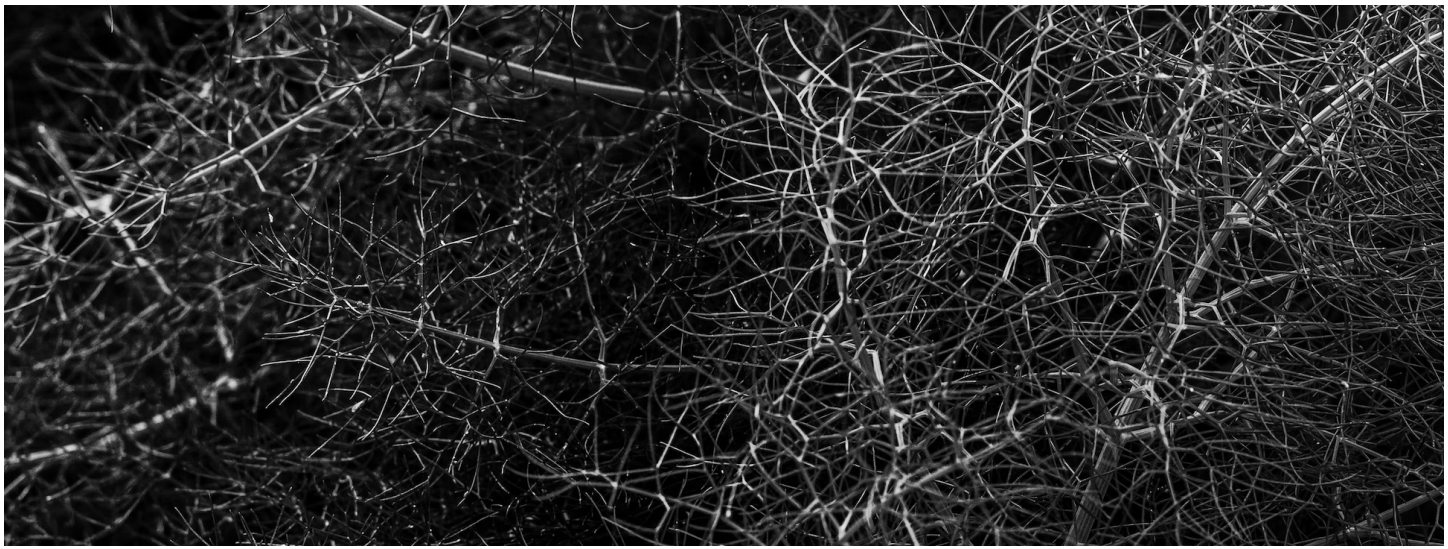
🧪 EVALUATING EMBEDDINGS

Downstream and intrinsic evaluations. Latter:

$$\hat{\mathbf{b}}^* = \arg\min_{\mathbf{x}} \text{distance}\left(\mathbf{x}, \mathbf{a}^* - \mathbf{a} + \mathbf{b}\right)$$

Allocational and representational harm (also see this talk by Kate Crawford). Try it yourself.

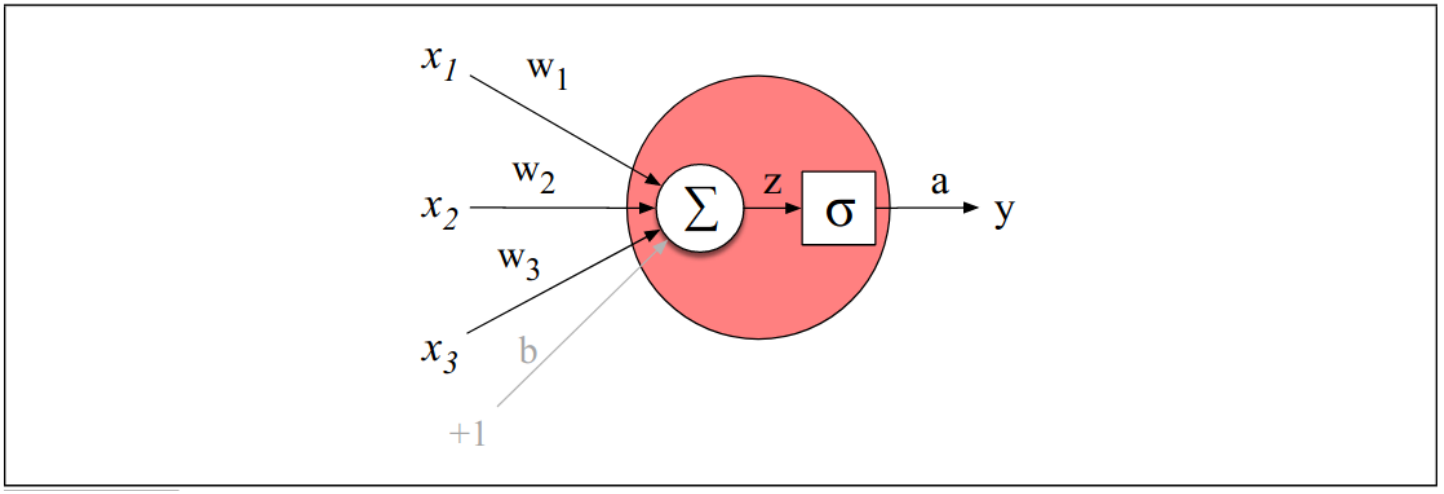
🤖 NEURAL MODELS OF LANGUAGE



🦄 THE NEURAL IN DEEP LEARNING

- Early work biologically inspired, not anymore.
- This lecture: **feedforward network**.
- Stack of logistic regression models, but:
  - Non-linearities; one layer can learn any function.
  - Representation learning; no hand-crafting features.

🎱 SINGLE NEURON

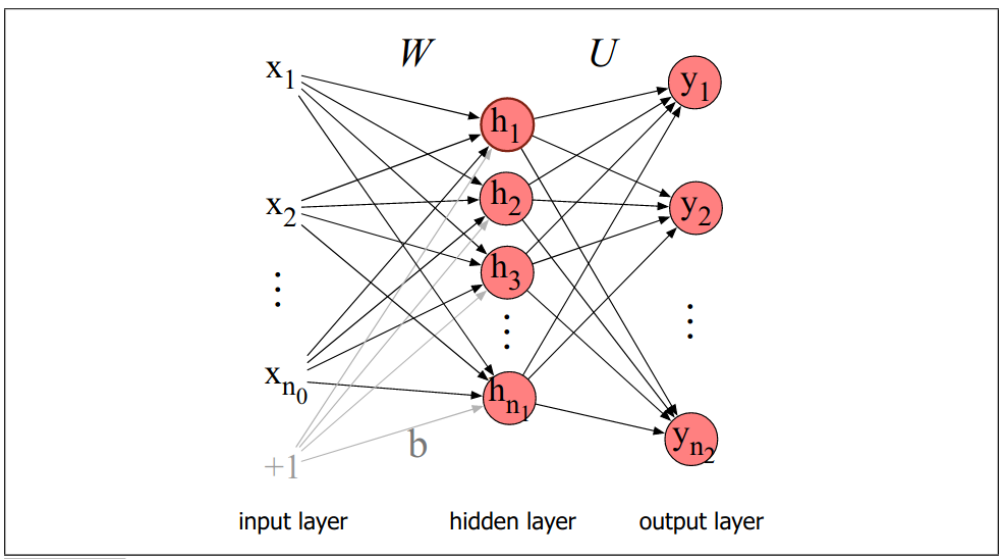


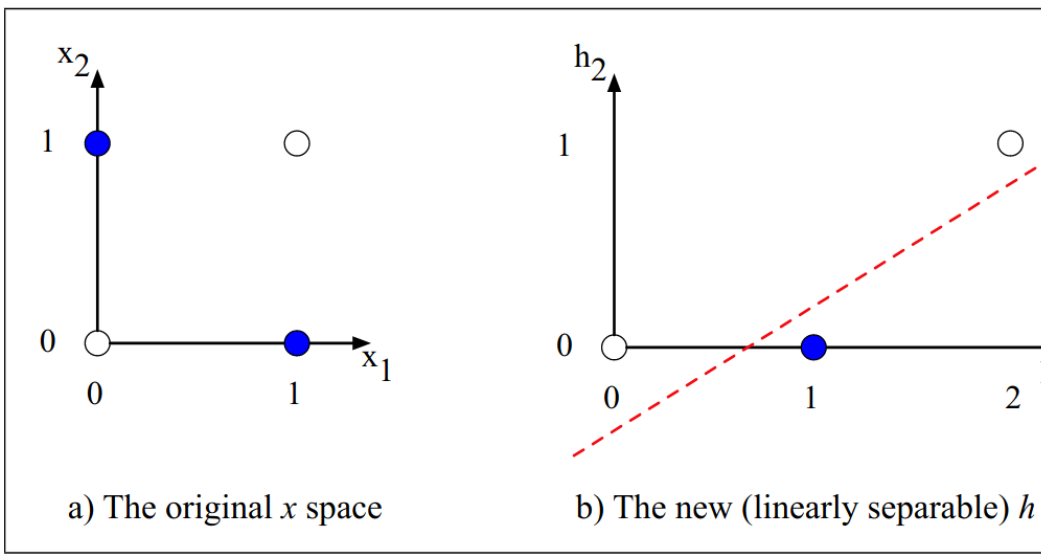
$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

Often uses tanh ($y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$) or ReLU

$y = max\ (z, 0)$ as activations rather than sigmoid.

NEURAL NETWORK





❌ FORMALLY

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = \mathbf{U}\mathbf{h}$$

$$\mathbf{y} = \text{softmax}(\mathbf{z})$$

$$\text{softmax}\left(\mathbf{z}_i\right) = \frac{\exp\left(\mathbf{z}_i\right)}{\sum_{j=1}^{d}\exp\left(\mathbf{z}_j\right)} \quad 1 \leq i \leq d$$

for i in $1 \ldots n$

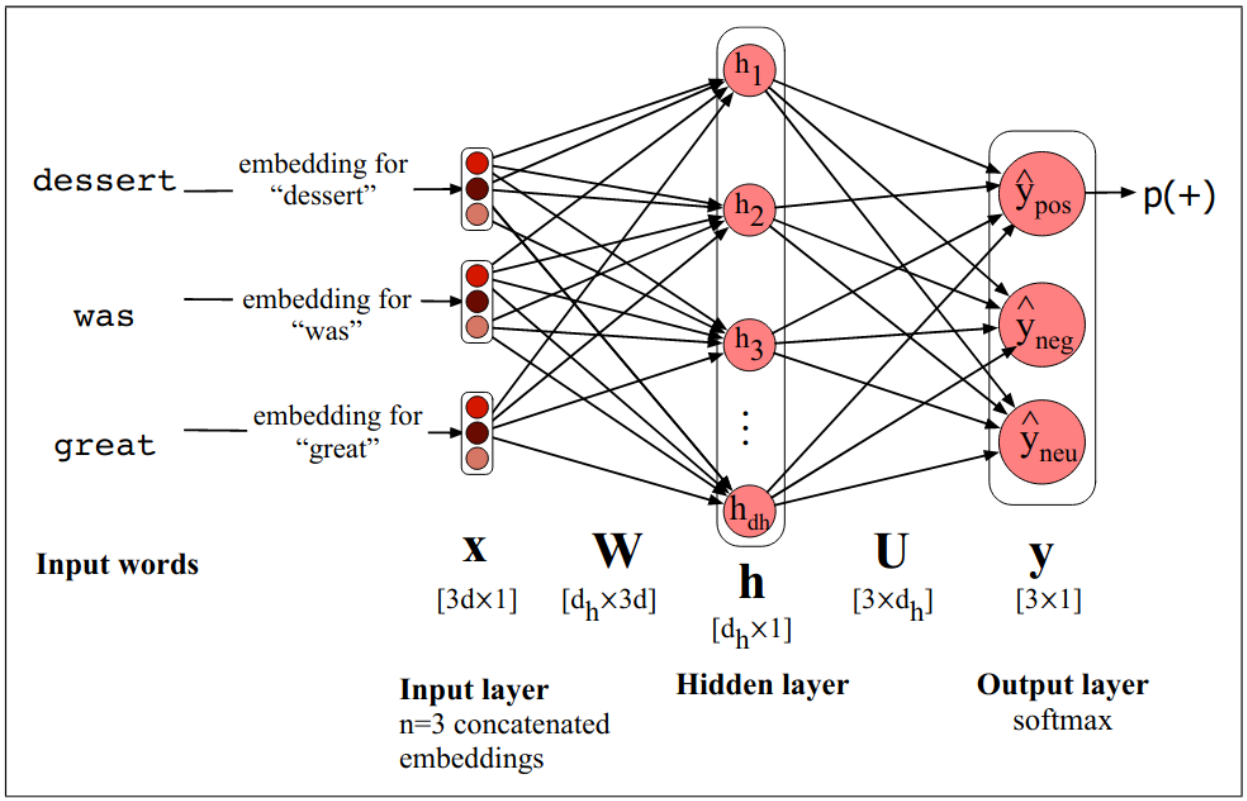
$$\mathbf{z}^{[i]} = \mathbf{W}^{[i]}\mathbf{a}^{[i-1]} + \mathbf{b}^{[i]}$$

$$\mathbf{a}^{[i]} = g^{[i]}\left(\mathbf{z}^{[i]}\right)$$

$$\hat{\mathbf{y}} = \mathbf{a}^{[n]}$$

👍 NNS FOR CLASSIFICATION

- Learn representation for input documents.
- Output predictions using softmax.
- **Pre-train**: use existing word embeddings as input!
- Encode longer histories, generalize over different meanings.



NNS FOR LANGUAGE MODELLING

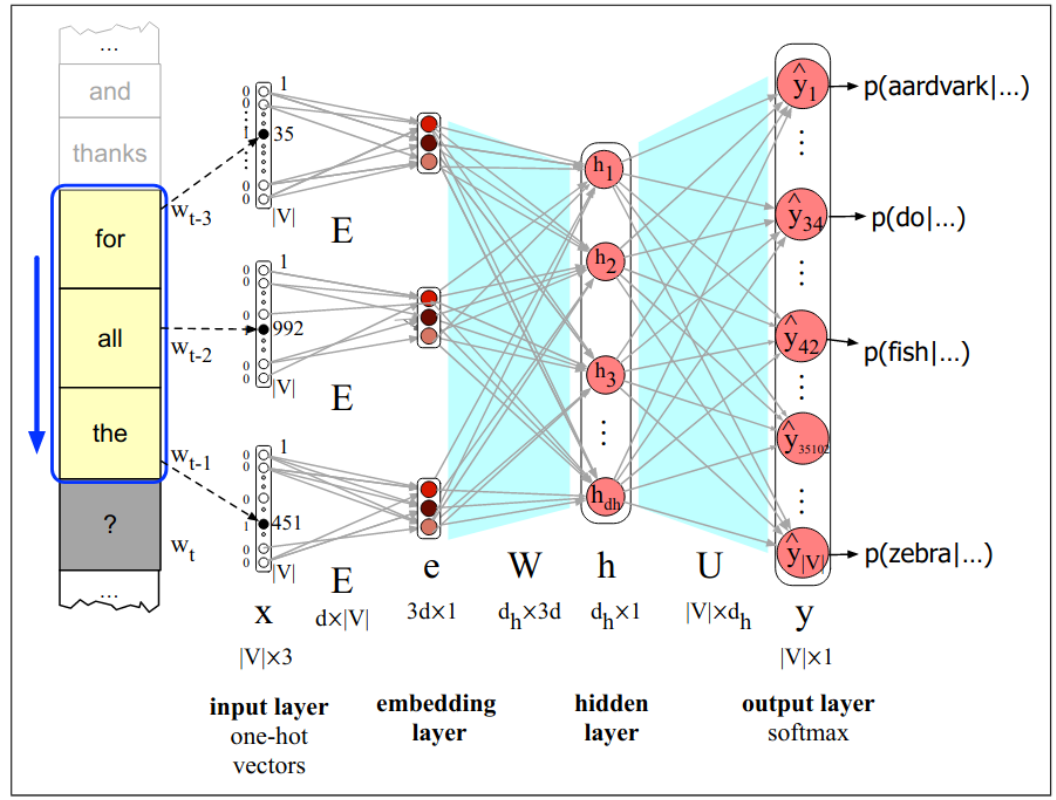
- Approximate probability of a word based on N previous **word embeddings**: $P(w_t \mid w_{t-N+1}, \ldots, w_{t-1})$.
- We get these by multiplying a one-hot vector (shown earlier) by embedding matrix $\mathbf{E}$.

$$\mathbf{e} = \left[\mathbf{E}\mathbf{x}_{t-3}; \mathbf{E}\mathbf{x}_{t-2}; \mathbf{E}\mathbf{x}_{t-1}\right]$$

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{e} + \mathbf{b})$$

$$\mathbf{z} = \mathbf{U}\mathbf{h}$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$$



📝 A NOTE ON BACKPROP

- Modern NN/DL Libraries (PyTorch, TensorFlow, Keras) have auto-differentation.
- If you want to read the chapters on derivatives, the chain rule, gradient descent, backpropagation, etc. in detail: knock yourself out.
- I will not ask you to calculate gradient updates by hand.

QUESTIONS

Post on the Discussion board and join class on Thursdays!