Representation learning: Embeddings

CS 4804: Introduction to AI

Fall 2025

https://tuvllms.github.io/ai-fall-2025/

Tu Vu

VIRGINIA TECH

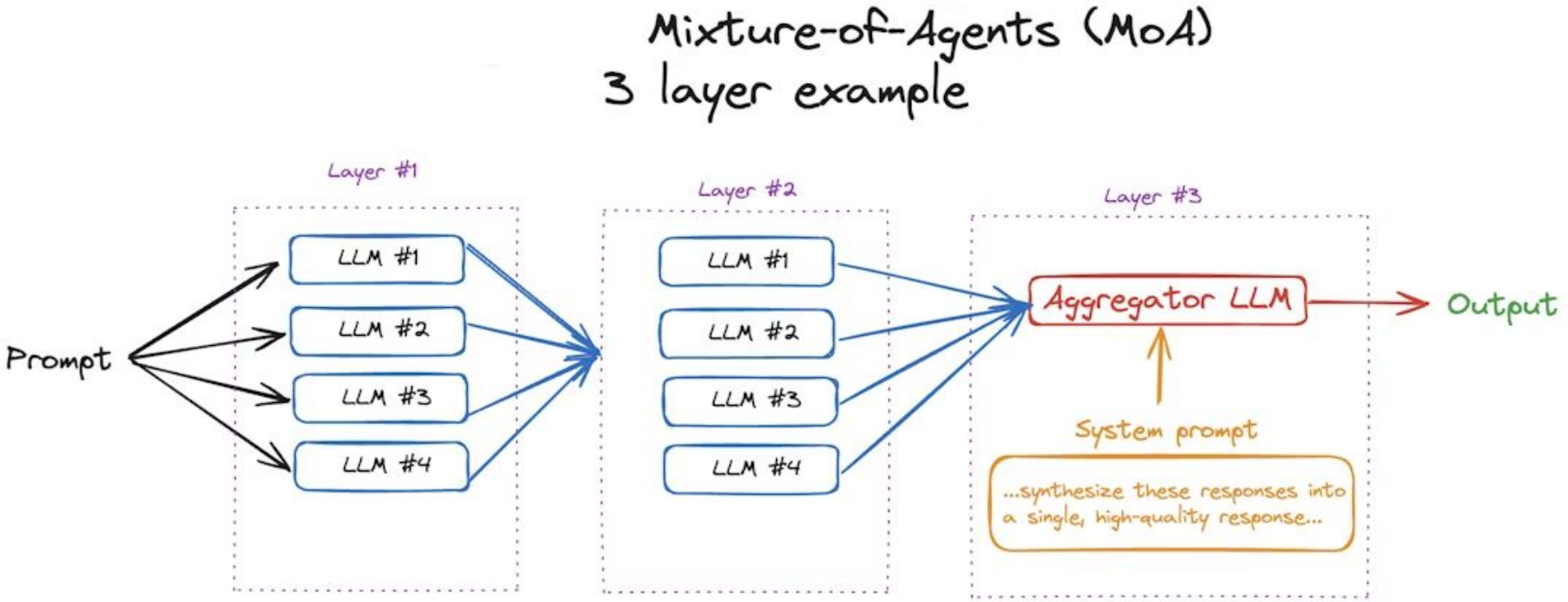
Logistics

- Homework 0 (due September 16th)
- Quiz 0 (due September 12th)
- Final project groups finalized this week

AI News: Nano Banana

- https://blog.google/products/gemini/updated-image-editing-model/
- Try here https://aistudio.google.com/models/gemini-2-5-flash-image
- Within just a few days of launching, the Gemini app has over 10 million new users thanks to Nano Banana

Deep Think / Parallel Agents / Mixture-of-Agents



https://docs.together.ai/docs/mixture-of-agents#advanced-moa-example

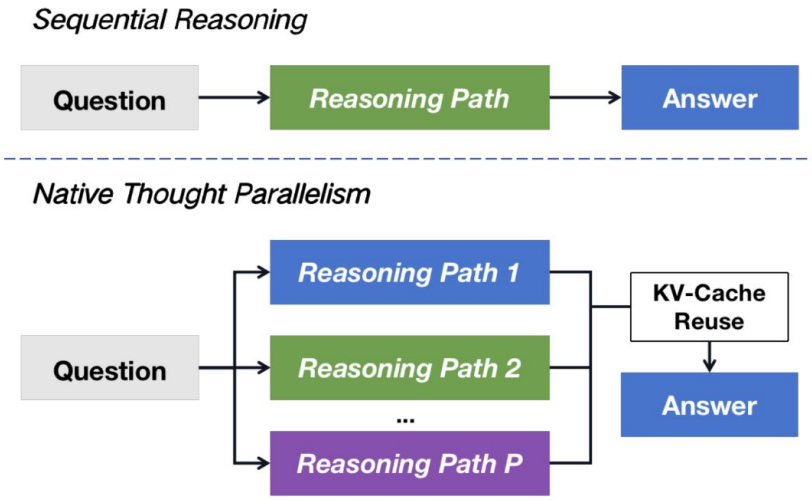
ParaThinker: Native Parallel Thinking as a New Paradigm to Scale LLM Test-time Compute

Hao Wen[1,*], **Yifan Su**[1,*,‡], **Feifei Zhang**[1], **Yunxin Liu**[1], **Yunhao Liu**[2], **Ya-Qin Zhang**[1], **Yuanchun Li**[1,†]

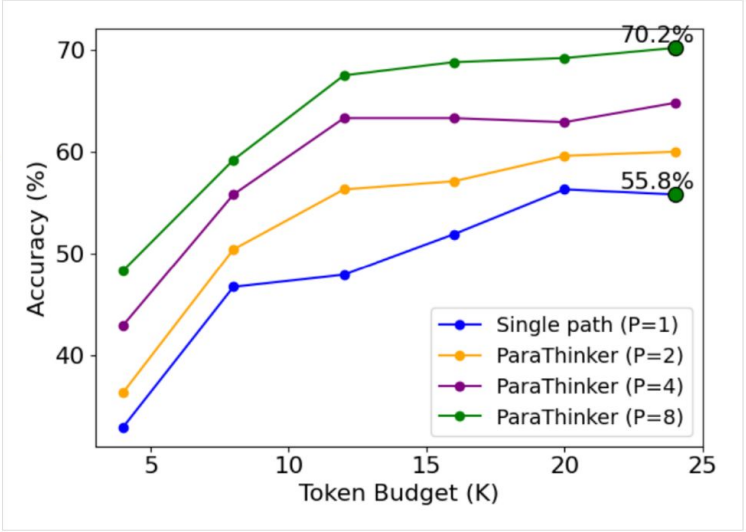
[1]Institute for AI Industry Research (AIR), Tsinghua University

[2]Global Innovation Exchange & Department of Automation, Tsinghua University

https://www.arxiv.org/abs/2509.04475



(1) ParaThinker Overview



(2) Scaling with Token Budget and Paths

The Majority is not always right: RL training for solution aggregation

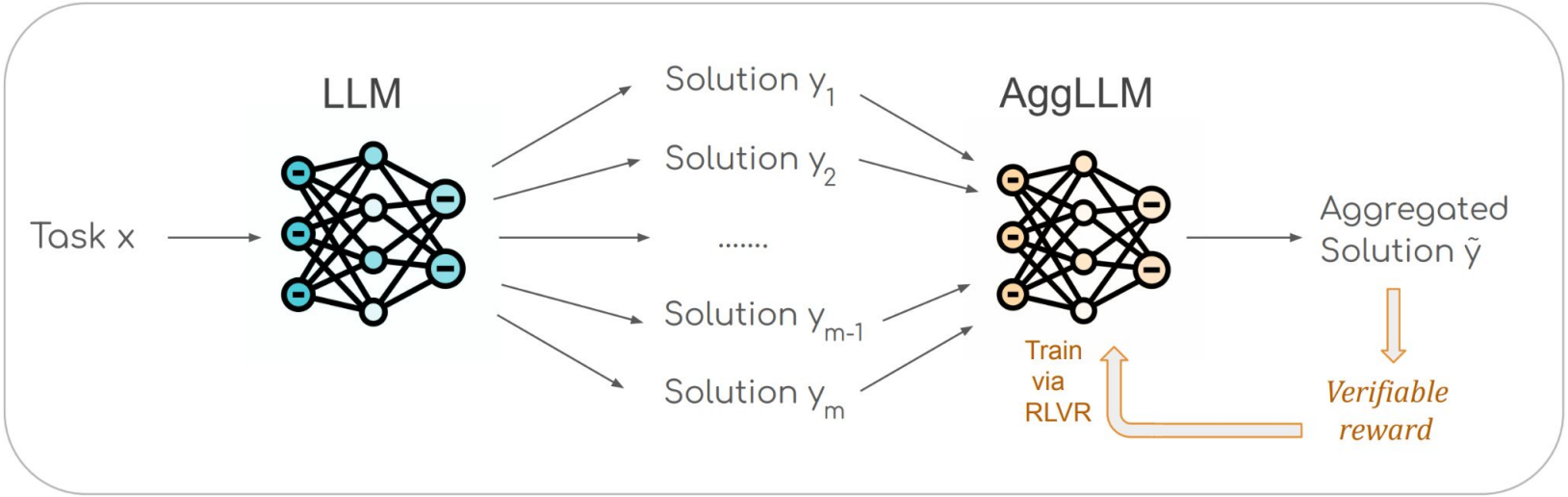
Wenting Zhao[1] **Pranjal Aggarwal**[1,2] **Swarnadeep Saha**[1] **Asli Celikyilmaz**[1] **Jason Weston**[1] **Ilia Kulikov**[1]

[1]FAIR at Meta, [2]CMU

Scaling up test-time compute, by generating multiple independent solutions and selecting or aggregating among them, has become a central paradigm for improving large language models (LLMs) on challenging reasoning tasks. While most prior work relies on simple majority voting or reward model ranking to aggregate solutions, these approaches may only yield limited benefits. In this work, we propose to learn aggregation as an explicit reasoning skill: given a set of candidate solutions, we train an aggregator model to review, reconcile, and synthesize a final, correct answer using reinforcement learning from verifiable rewards. A key ingredient is careful balancing of easy and hard training examples, allowing the model to learn both to recover minority-but-correct answers as well as easy majority-correct answers. Empirically, we find our method, AggLM, outperforms both strong rule-based and reward-model baselines, across multiple benchmarks. Furthermore, it generalizes effectively to solutions from differing models, including stronger ones than contained in the training data, all while requiring substantially fewer tokens than majority voting with larger numbers of solutions.

Date: September 9, 2025

Meta



https://arxiv.org/abs/2509.06870

Cosine similarity

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

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

normalized dot product

Cosine similarity (cont'd)

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

The maximum value is

$$\max \cos(\mathbf{v}, \mathbf{w}) = 1,$$

and equality holds exactly when

$$\frac{v_1}{w_1} = \frac{v_2}{w_2} = \cdots = \frac{v_N}{w_N},$$

for all indices with $w_i \neq 0$.

Cross-entropy loss review

For probability distributions p, q:

$$H(p, q) = -\sum_i p_i \log q_i.$$

Gibbs' inequality states:

$$H(p, q) \geq H(p) = -\sum_i p_i \log p_i,$$

with equality if and only if $p = q$.

Cross-entropy loss (cont'd)

The predicted probabilities

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_V \end{bmatrix}$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_V \end{bmatrix}$$

The ground truth label

$$y_i = \begin{cases} 1, & \text{if } i = c \text{ (correct class index)} \\ 0, & \text{otherwise} \end{cases}$$

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}, \quad \text{for } i = 1, 2, \ldots, V$$

Cross-entropy loss (cont'd)

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{V} y_i \log \hat{y}_i$$

$$L_{CE}(\hat{y}, y) = -\left(y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \cdots + y_V \log \hat{y}_V\right)$$

Since the true label y is one-hot encoded, only one term in the sum is nonzero, corresponding to the correct class c, where $y_c = 1$ and $y_i = 0$ for all $i \neq c$. This simplifies the sum to:

$$L_{CE}(\hat{y}, y) = -y_c \log \hat{y}_c$$

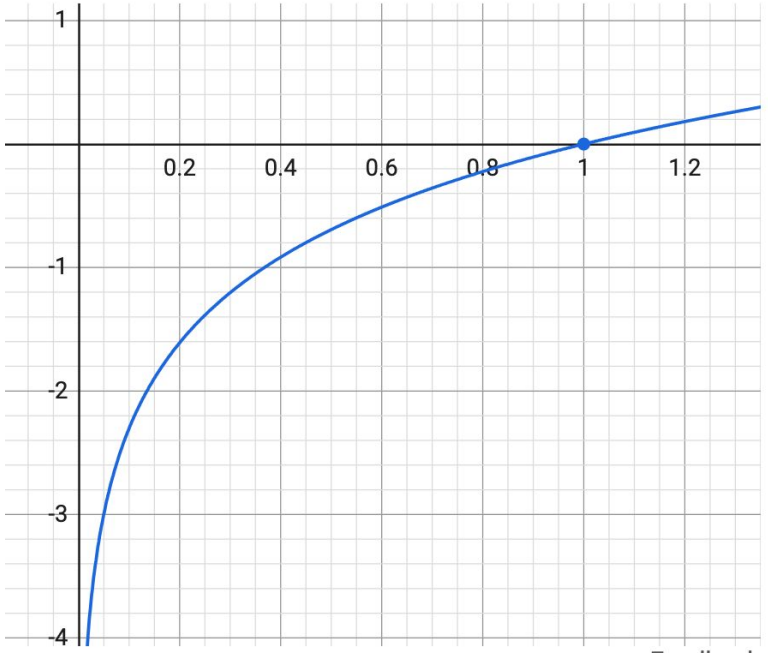
Since $y_c = 1$, this further reduces to:

$$L_{CE}(\hat{y}, y) = -\log \hat{y}_c$$

The loss models the distance between the system output and the gold output —lower is better

Cross-entropy loss (cont'd)

$$L_{CE}(\hat{y}, y) = -\log \hat{y}_c$$



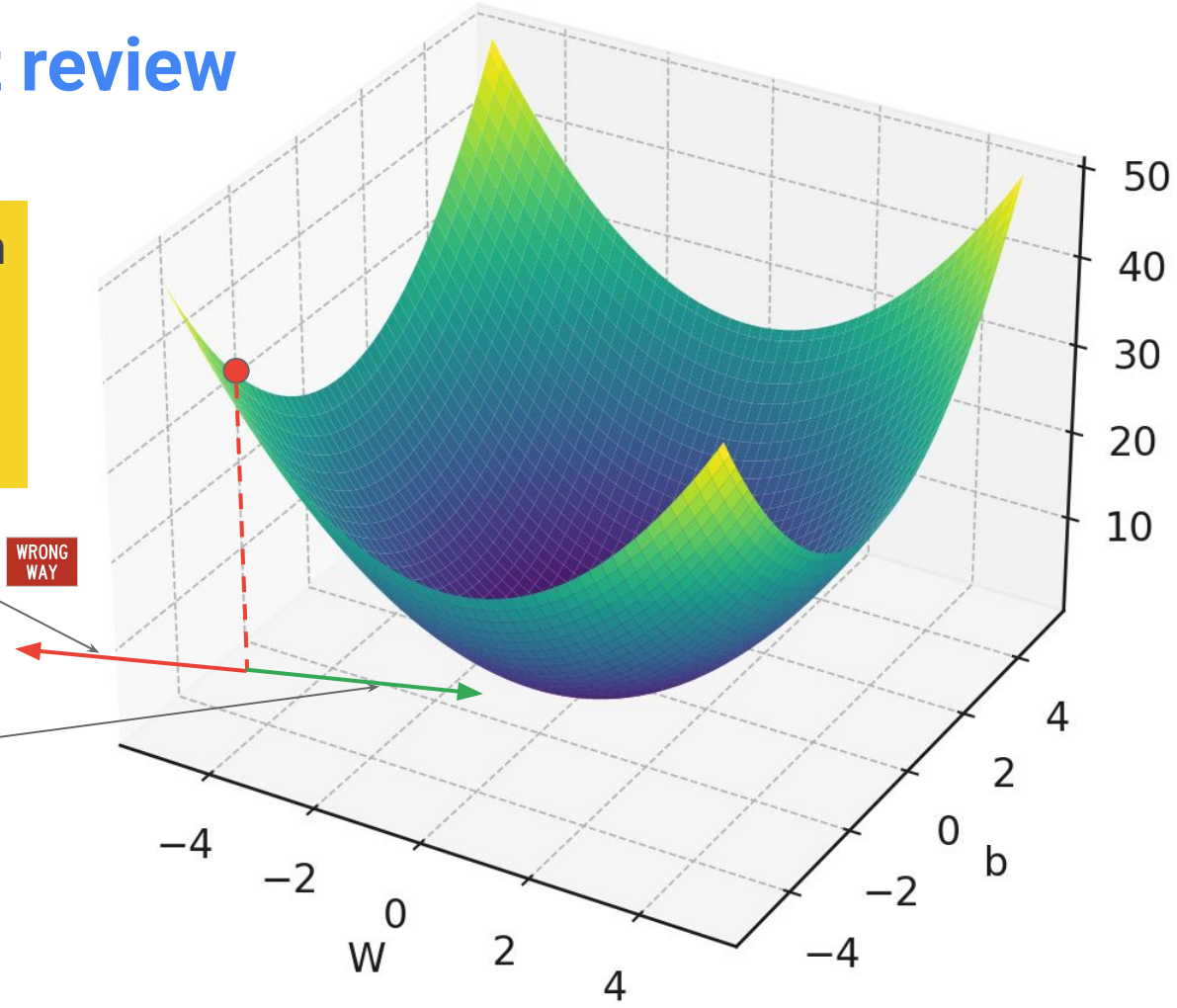
$$\hat{y}_c \to 0, \log \hat{y}_c \to -\infty$$

- If $\hat{y}_c = 0.9$, then $\log(0.9) \approx -0.105$, and the loss will be small.
- If $\hat{y}_c = 0.1$, then $\log(0.1) \approx -2.302$, and the loss will be much larger.

Gradient descent review

the gradient points in the direction of the steepest increase in the loss

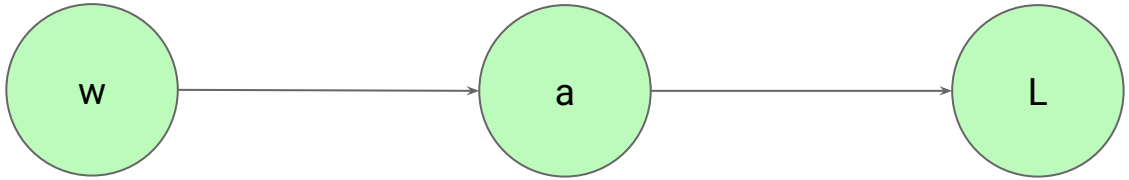
negative gradient



Updating parameters

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w_t}$$

Backpropagation

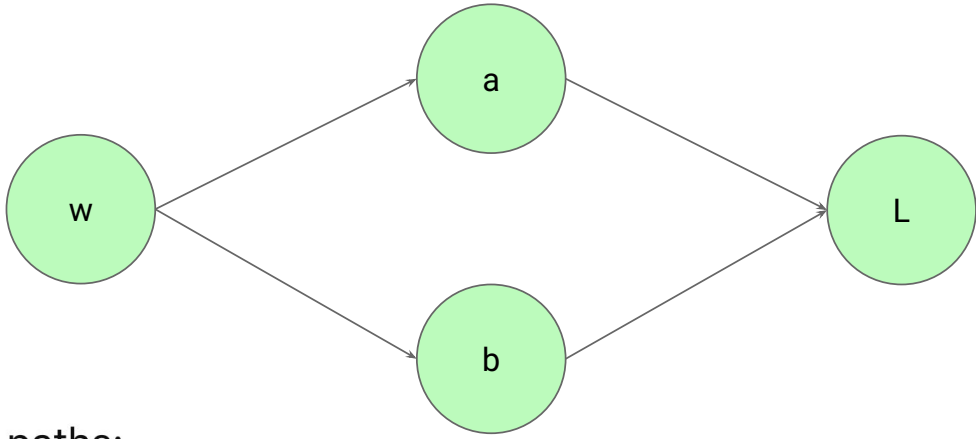


$$\frac{dL}{dw} = \frac{dL}{da} \cdot \frac{da}{dw}.$$

Step-by-step

1. Identify the local derivative from a to L: $\frac{dL}{da}$.
2. Identify the local derivative from w to a: $\frac{da}{dw}$.
3. Multiply them: $\frac{dL}{dw} = \frac{dL}{da} \cdot \frac{da}{dw}$.

Backpropagation (cont'd)



Gradient Calculation

By the multivariable chain rule, we sum over all paths:

$$\frac{dL}{dw} = \frac{dL}{da}\frac{da}{dw} + \frac{dL}{db}\frac{db}{dw}$$

Step-by-step

1. Compute $\frac{dL}{da}$ and $\frac{da}{dw}$ for the $w \to a \to L$ path.
2. Compute $\frac{dL}{db}$ and $\frac{db}{dw}$ for the $w \to b \to L$ path.
3. Add them together.

Word2vec

- Continuous bag-of-words (CBoW)
- Skip-gram with negative sampling (SGNS)

Simple count-based embeddings

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

	a	**computer**	**pie**
cherry	1	0	1
strawberry	0	0	2
digital	0	1	0
information	1	1	0

Simple count-based embeddings (cont'd)

	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	...

Skip-gram with negative sampling (SGNS)

- Simplifies the task: *binary classification* instead of *word prediction*

- Simplifies the architecture: *logistic regression* instead of *multi-layer neural network*

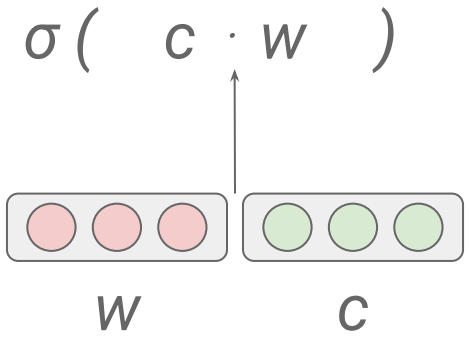
Skip-gram with negative sampling (SGNS) (cont'd)

- Intuition
  - Treat the target word and a neighboring context word as *positive examples*
  - Randomly sample other words in the lexicon to get *negative samples*
  - Train a classifier (**self-supervision**) to distinguish those two cases
  - Use the learned weights as the embeddings

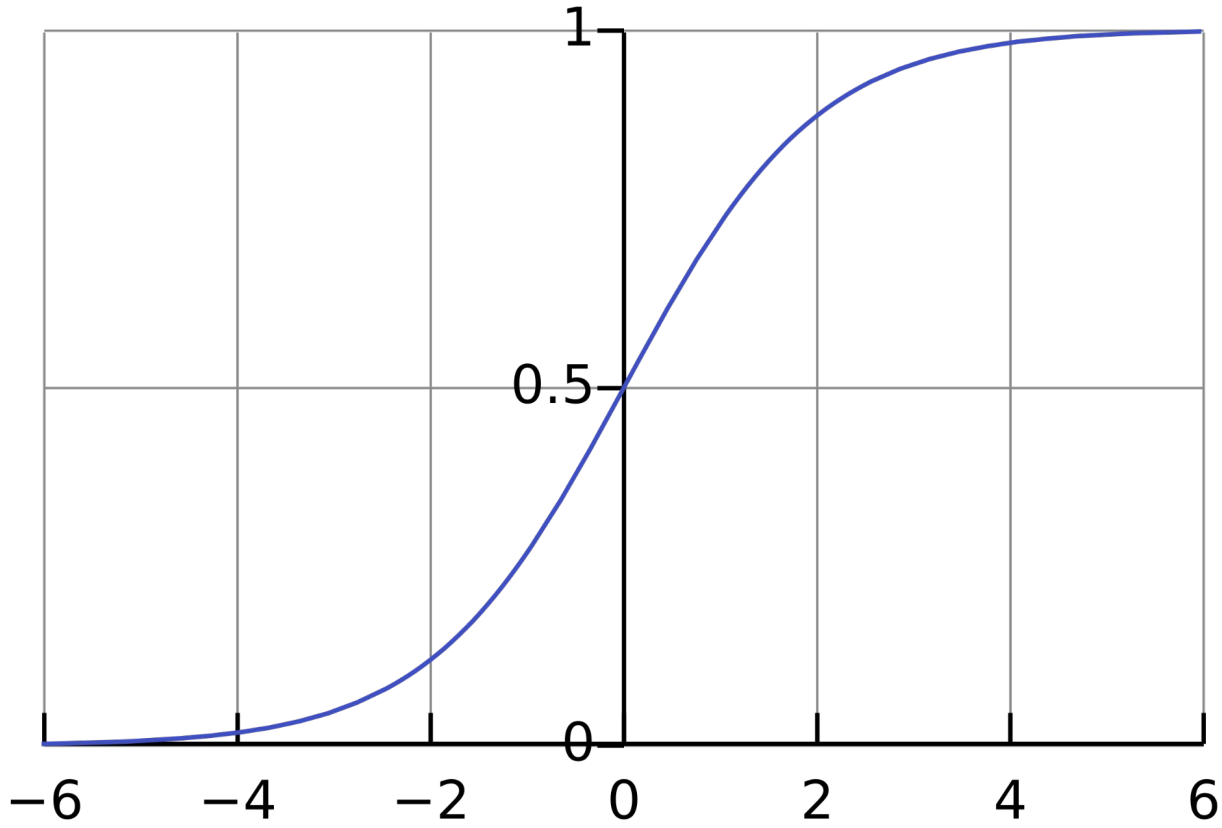
Logistic regression: Sigmoid function

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

the probability that word *c* is a real context word for target word *w*



Sigmoid function



Sigmoid function (cont'd)

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

For $1 - \sigma(x)$:

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

Dividing numerator and denominator by e^{-x}:

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

Logistic regression (cont'd)

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

the probability that word *c* is a real context word for target word *w*

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

the probability that word *c* is *not* a real context word for target word *w*

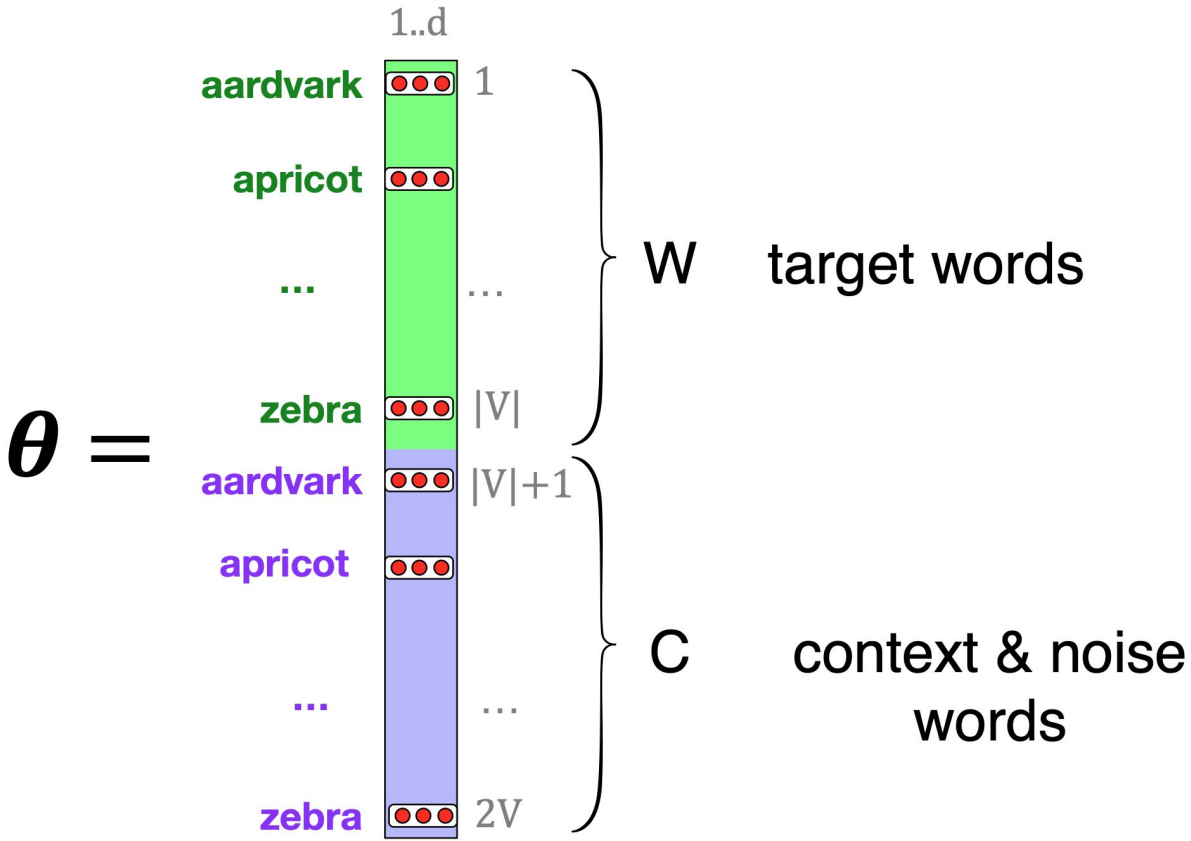
The probability for many context words

Given a target word w and its context window of *L* words $c_{1:L}$, assigns a probability based on how similar this context window is to the target word

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

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

Two embedding tables



Learning skip-gram embeddings

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

This example has a target word w (apricot), and 4 context words in the $L = \pm 2$ window, resulting in 4 positive training instances (on the left below):

positive examples +

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

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Loss function

Maximize the similarity of the target word, context word pairs (w, c_{pos}) drawn from the positive examples

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

Loss function (cont'd)

Minimize the similarity of the (w, c_{neg}) pairs from the negative examples

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

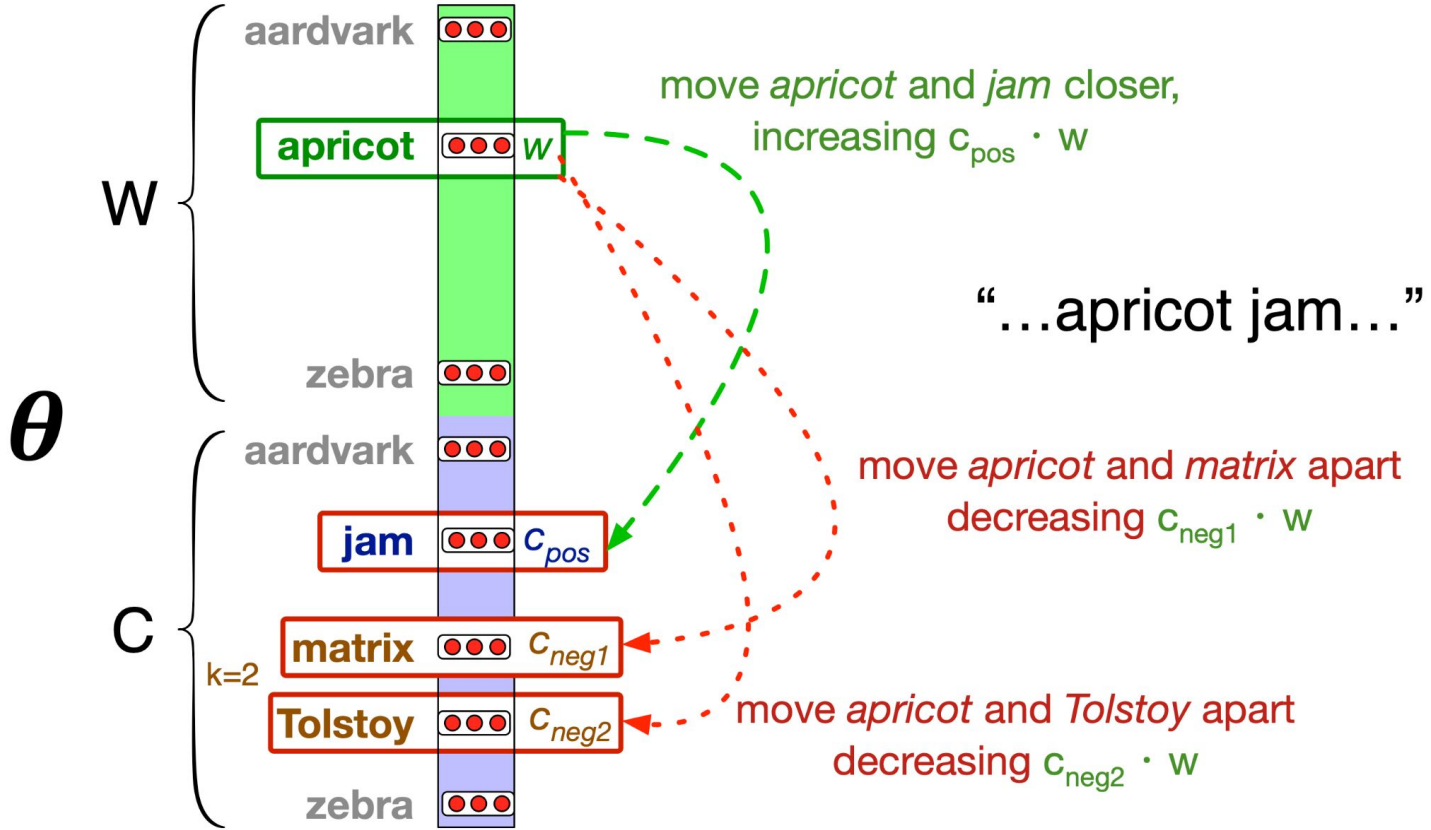
Training

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

The update equations going from time step t to $t+1$ in stochastic gradient descent are thus:

$$\mathbf{c}_{pos}^{t+1} = \mathbf{c}_{pos}^{t} - \eta[\sigma(\mathbf{c}_{pos}^{t} \cdot \mathbf{w}^{t}) - 1]\mathbf{w}^{t}$$
$$\mathbf{c}_{neg}^{t+1} = \mathbf{c}_{neg}^{t} - \eta[\sigma(\mathbf{c}_{neg}^{t} \cdot \mathbf{w}^{t})]\mathbf{w}^{t}$$
$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta\left[[\sigma(\mathbf{c}_{pos}^{t} \cdot \mathbf{w}^{t}) - 1]\mathbf{c}_{pos}^{t} + \sum_{i=1}^{k}[\sigma(\mathbf{c}_{\mathrm{neg}_i}^{t} \cdot \mathbf{w}^{t})]\mathbf{c}_{neg_i}^{t}\right]$$

Training (cont'd)

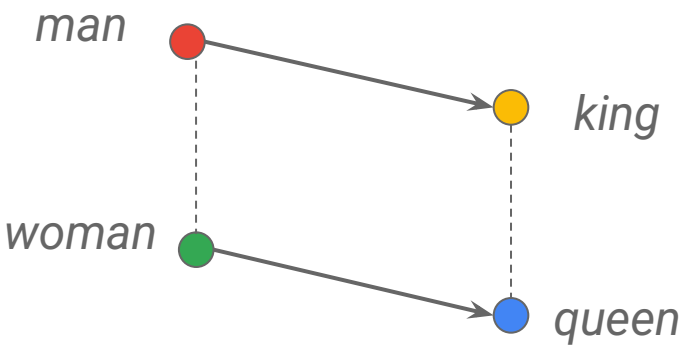


A computational approach

Visualizing word embeddings

A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space.

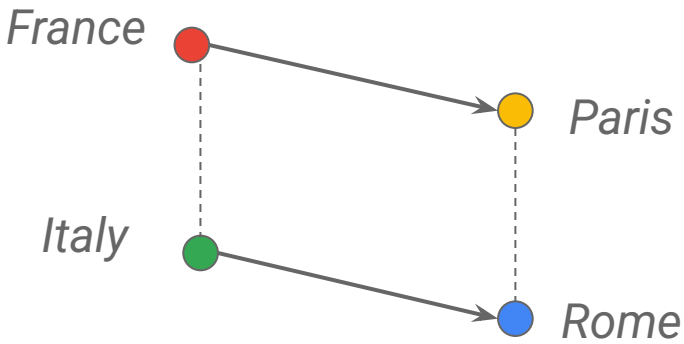
Semantic properties of word embeddings

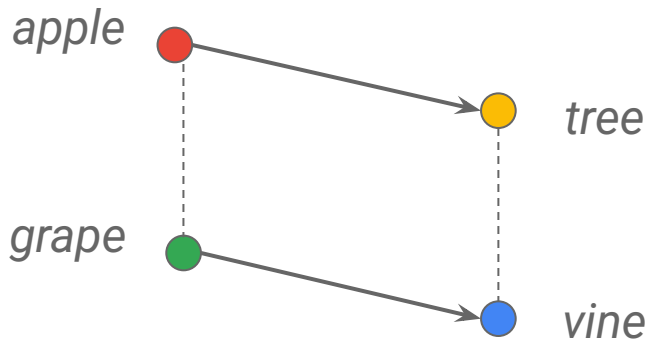


$$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} \approx \overrightarrow{\text{queen}} - \overrightarrow{\text{woman}}$$

$$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{queen}}$$

Semantic properties of word embeddings (cont'd)





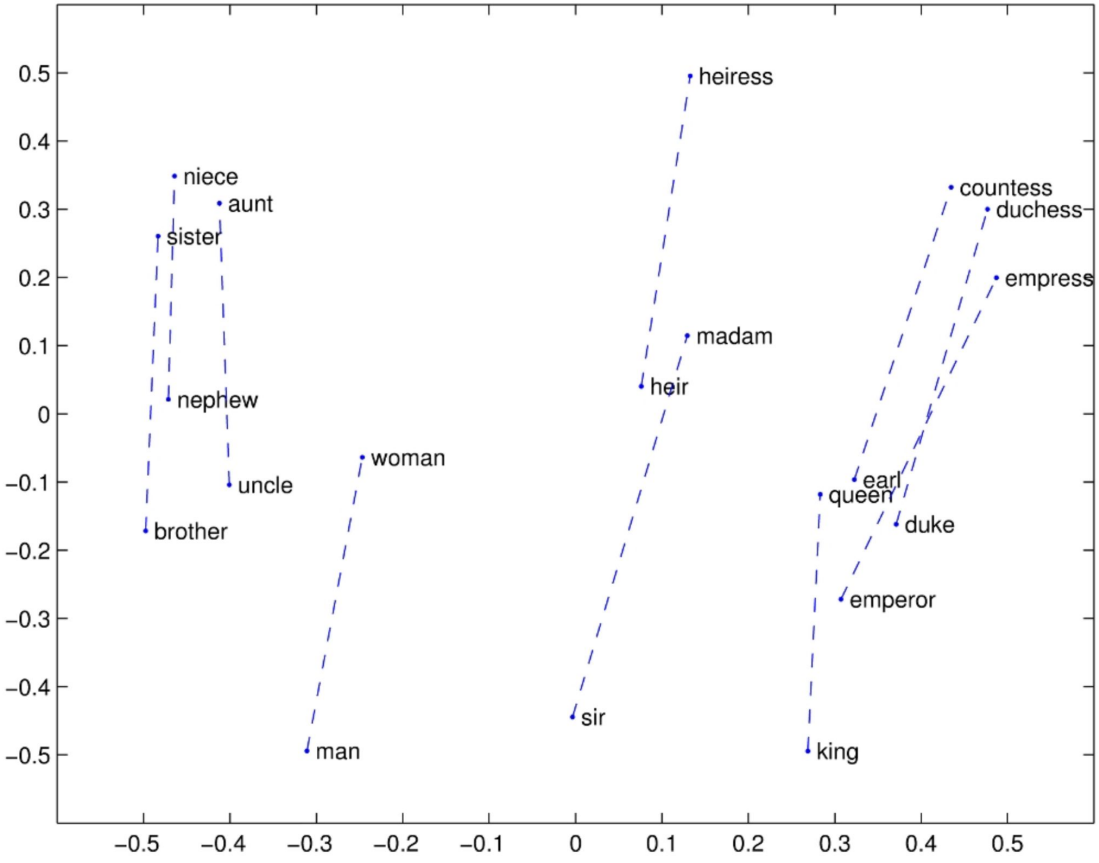
$$\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} \approx \overrightarrow{\text{Rome}} - \overrightarrow{\text{Italy}}$$

$$\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}} \approx \overrightarrow{\text{Rome}}$$

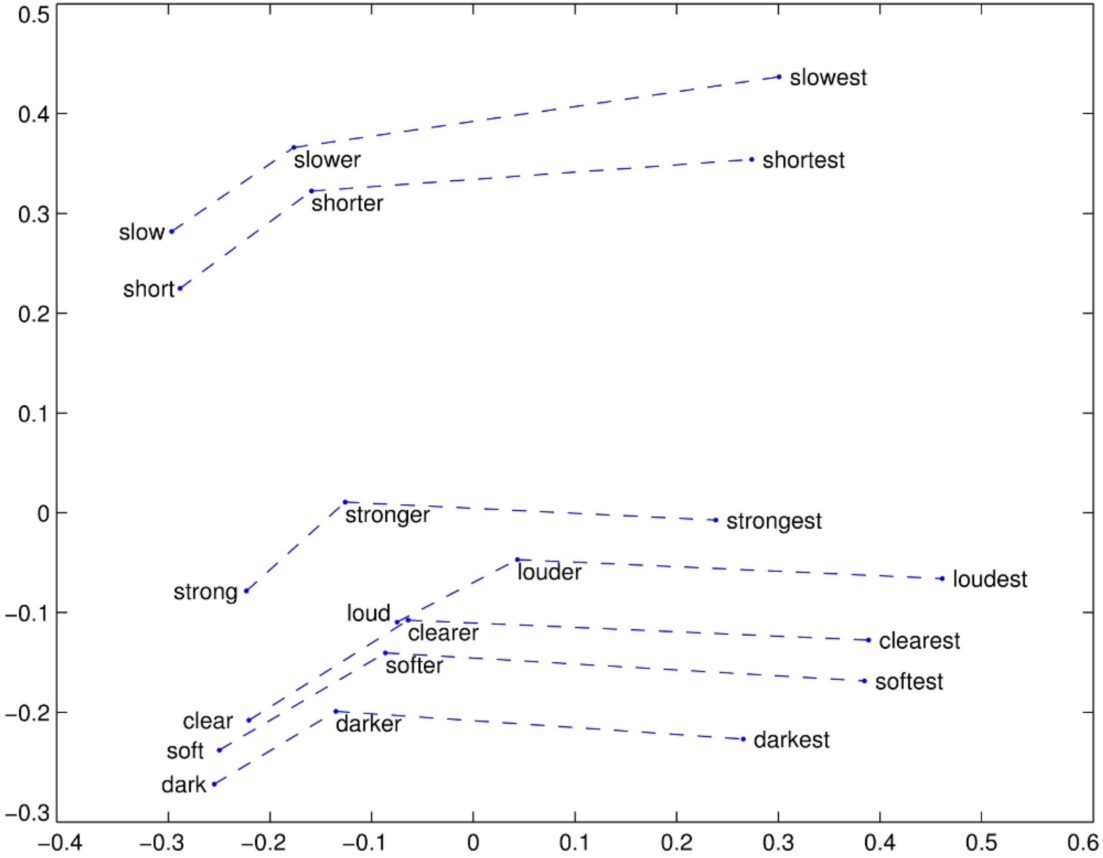
$$\overrightarrow{\text{tree}} - \overrightarrow{\text{apple}} \approx \overrightarrow{\text{vine}} - \overrightarrow{\text{grape}}$$

$$\overrightarrow{\text{tree}} - \overrightarrow{\text{apple}} + \overrightarrow{\text{grape}} \approx \overrightarrow{\text{vine}}$$

Semantic properties of word embeddings (cont'd)



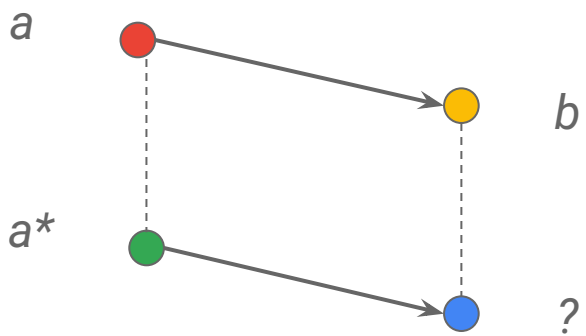
Semantic properties of word embeddings (cont'd)

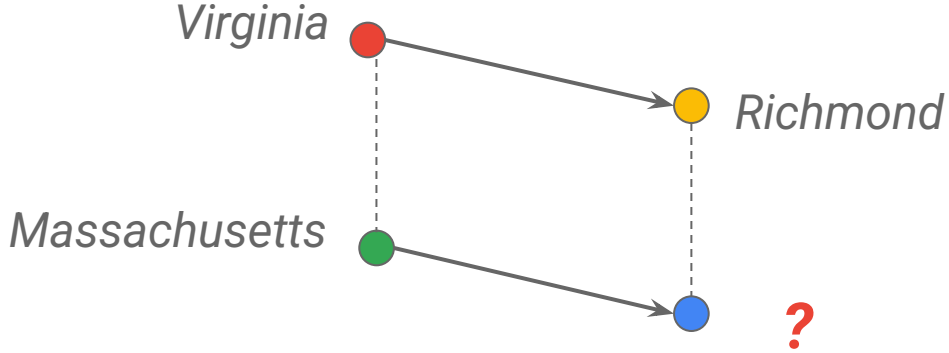


comparative & superlative morphology

Analogy problem $\mathbf{a} : \mathbf{b} :: \mathbf{a}^* : \mathbf{b}^*$

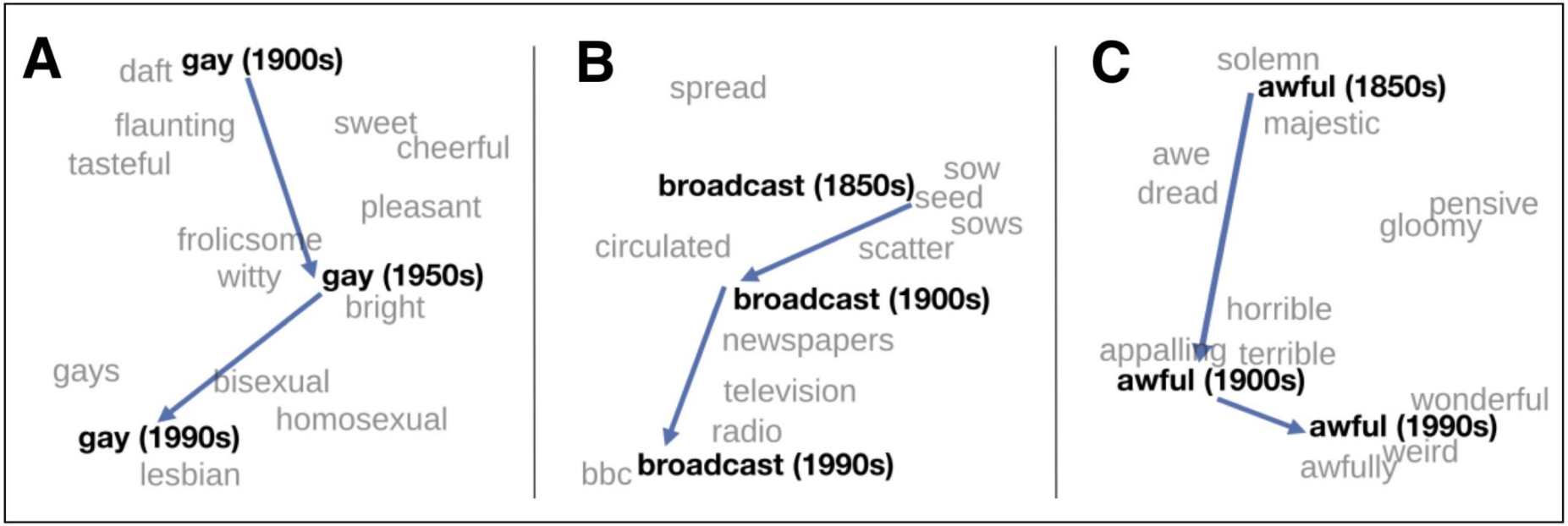
a* is to *b* as *a* **is to what?**





$$\hat{\mathbf{b}}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ \operatorname{distance}(\mathbf{x}, \mathbf{b} - \mathbf{a} + \mathbf{a}^*)$$

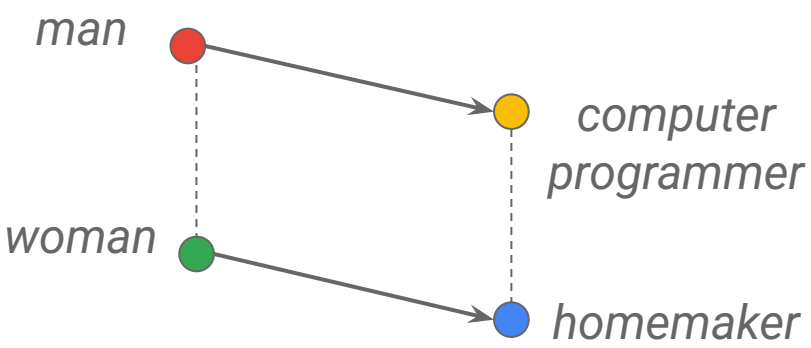
Historical Semantics

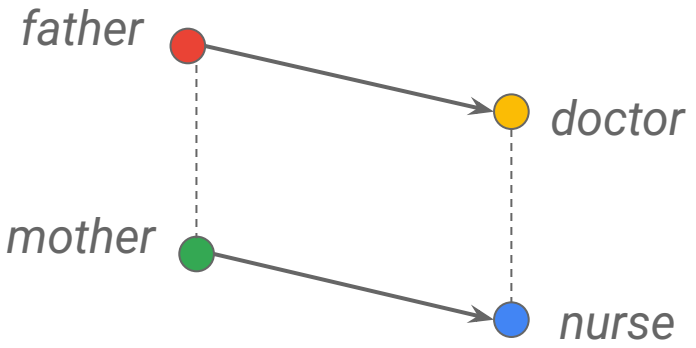


A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors.

Biases in word embeddings

- Gender stereotypes





Biases in word embeddings (cont'd)

- People in the US associate:
  - African-American names with unpleasant words (more than European-American names)
  - male names more with mathematics and female names with the arts, and old people's names with unpleasant words
- …

Word embedding methods

- Word2vec
- Glove: https://nlp.stanford.edu/projects/glove/
- Fasttext: https://fasttext.cc/

Thank you!