Word Representation

CS 288 Spring 2026
UC Berkeley
cal-cs288.github.io/sp26

Berkeley EECS BAIR

Today's Question:
How to Represent a Word?

Computers don't understand words — they see them as symbols.

But can we make a computer understand *that* ***'king'*** *and* ***'queen'*** *are related*, or that ***'dog'*** *is similar to* ***'cat'***?

A naive option: One-hot encoding

Each word in the vocabulary is assigned a unique index and is represented as a binary vector with a single 1 in the position corresponding to the word's index and 0s elsewhere.

$$v_{\text{cat}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{dog}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{the}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{language}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \cdots \end{bmatrix}$$

Lookup table:

Index	0	1	2	3	4
Word	cat	dog	the	Language	…

High-dimensional & Sparse

Does not capture semantic meaning of the words (all words are equally distant)

Need for word *meaning*

- With words, a feature is a word identity (= string)
- Requires **exact same word** to be in the training and testing set

“terrible” ≠ “horrible”

- If we can represent **word meaning** in vectors:
  - The previous word was vector [35, 22, 17, …]
  - Now in the test set we might see a similar vector [34, 21, 14, …]
  - We can generalize to **similar but unseen** words!!!

What do words *mean*?

- **Synonyms**: couch/sofa, car/automobile, filbert/hazelnut
- **Antonyms**: dark/light, rise/fall, up/down
- Some words are not synonyms but they share some element of meaning
  - cat/dog, car/bicycle, cow/horse
- Some words are not similar but they are **related**
  - coffee/cup, house/door, chef/menu
- **Affective meanings** or **connotations**:

vanish	disappear	9.8
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999

valence: the pleasantness of the stimulus

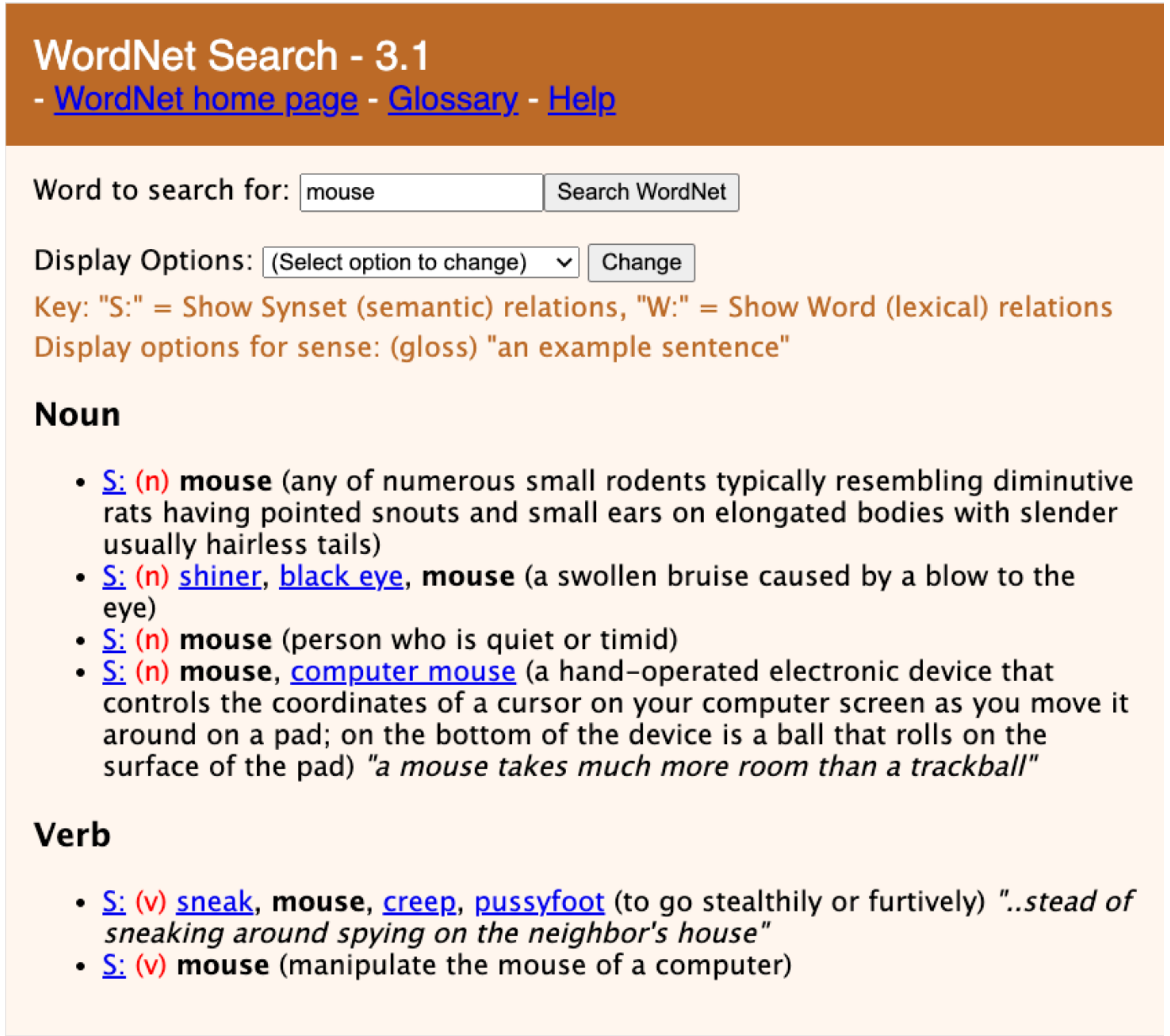
arousal: the intensity of emotion provoked by the stimulus

dominance: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

(Osgood et al., 1957)

Lexical resources



WordNet Search - 3.1
- [WordNet home page] - [Glossary] - [Help]

Word to search for: mouse [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **mouse** (any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails)
- S: (n) shiner, black eye, **mouse** (a swollen bruise caused by a blow to the eye)
- S: (n) **mouse** (person who is quiet or timid)
- S: (n) **mouse**, computer mouse (a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it around on a pad; on the bottom of the device is a ball that rolls on the surface of the pad) *"a mouse takes much more room than a trackball"*

Verb

- S: (v) sneak, **mouse**, creep, pussyfoot (to go stealthily or furtively) *"..stead of sneaking around spying on the neighbor's house"*
- S: (v) **mouse** (manipulate the mouse of a computer)

https://wordnet.princeton.edu/

(-) Huge amounts of human labor to create and maintain

Distributional hypothesis

A simple linguistic intuition

“Words that occur in similar contexts tend to have similar meanings.”

Example:

- “The **cat** sat on the _____.”
- “The **dog** lay on the _____.”

Distributional semantics

Words that occur in similar **contexts** tend to have similar meanings

J.R.Firth 1957

- "You shall know a word by the company it keeps"
- One of the most successful ideas of modern statistical NLP!

When a word *w* appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

This idea turns semantics into a statistical problem over text.

Distributional semantics

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

Quick quiz: Word guessing!

(Example from Eisenstein’s book)

Everybody likes tezg¨uino.

We make tezg¨uino out of corn.

A bottle of tezg¨uino is on the table.

Don’t have tezg¨uino before you drive.

Distributional semantics

How we can do the same thing computationally?

- Count the words in the context of a target word
- See what other words occur in those contexts

We can represent a word’s context using vectors!

Word-word co-occurrence matrix

Solution: Let's use **word-word co-occurrence counts** to represent the meaning of words!

Each word is represented by the corresponding **row vector**

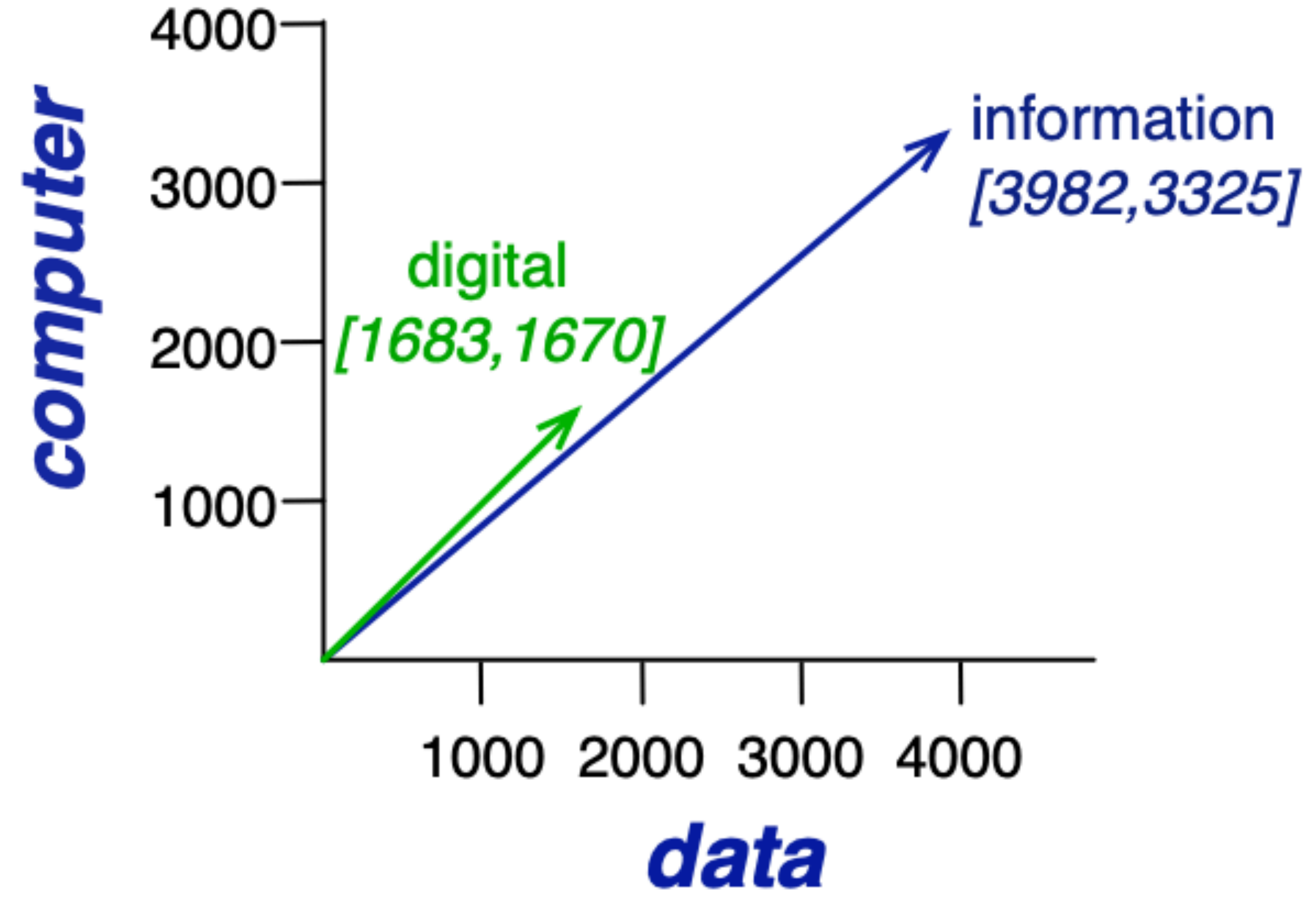
context words:
4 words to the left +
4 words to the right

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	...

Very high-dimensional; Most entries are 0s $\Longrightarrow$ sparse vectors

Measuring similarity



A common similarity metric: **cosine** of the angle between the two vectors (the larger, the more similar the two vectors are)

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2}\sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

Q: Why cosine similarity instead of dot product $\mathbf{u} \cdot \mathbf{v}$?

Measuring similarity

What is the range of $\cos(u, v)$ if u, v are **count vectors**?

(A) [-1, 1]
(B) [0, 1]
(C) (0, 1)
(D) (-1, 1)
(E) $(-\infty, +\infty)$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2}\sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

The answer is (b). Cosine similarity ranges between -1 and 1 in general. In this model, all the values of u_i, v_i are non-negative.

Sparse vs. dense vectors

Sparse vs. dense vectors

- The vectors in the word-word occurrence matrix are
  - Long: vocabulary size
  - Sparse: most are 0's
- Alternative: we want to represent words as short (50-300 dimensional) & dense (real-valued) vectors
  - This is the basis for modern NLP systems!

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix} \quad v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Why dense vectors?

- Short vectors are easier to use as features in ML systems
- Dense vectors may generalize better than explicit counts
- They can't capture high-order co-occurrence
  - w_1 co-occurs with "car", w_2 co-occurs with "automobile"
  - They should be similar but they aren't because "car" and "automobile" are distinct dimensions
- In practice, they work better!

How to get short dense vectors?

- **Count-based methods:** Singular value decomposition (SVD) of count matrix

$$
\underset{|V|\times|V|}{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ \\ C \\ \\ \\ \end{bmatrix}}
$$

$$
\underset{|V|\times|V|}{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}} = \underset{|V|\times k}{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \begin{bmatrix} C \\ k\times|V| \end{bmatrix}
$$

embedding for word i

$$
\underset{|V|\times k}{\begin{bmatrix} \\ \boxed{\quad\quad} \\ W \\ \\ \end{bmatrix}}
$$

We can approximate the full matrix by only keeping the top k (e.g., 100) singular values!

How to get short dense vectors?

- **Count-based methods**: Singular value decomposition (SVD) of count matrix

- **Prediction-based methods**:
  - Vectors are created by training a classifier to predict whether a word *c* ("pie") is likely to appear in the context of a word w ("cherry")
  - Examples: **word2vec** (Mikolov et al., 2013), **Glove** (Pennington et al., 2014), **FastText** (Bojanowski et al., 2017)

Also called word **embeddings!**

***Don't count, predict!* A systematic comparison of context-counting vs. context-predicting semantic vectors**

Marco Baroni and **Georgiana Dinu** and **Germán Kruszewski**
Center for Mind/Brain Sciences (University of Trento, Italy)

(Baroni et al., 2014)

Word embeddings

Word embeddings

= **Learned** representations from text for representing words

- Input: a large text corpora, *V*, *d*
  - V: a pre-defined vocabulary
  - d: dimension of word vectors (e.g. 300)
  - Text corpora:
    - Wikipedia + Gigaword 5: 6B tokens
    - Twitter: 27B tokens
    - Common Crawl: 840B tokens
- Output: $f: V \rightarrow \mathbb{R}^d$

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Each word is represented by a low-dimensional (e.g., d = 300), real-valued vector

Each coordinate/dimension of the vector doesn't have a particular interpretation

Trained word embeddings available

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Differ in algorithms, text corpora, dimensions, cased/uncased…

Applied to many other languages

Word embeddings

- Basic property: similar words have similar vectors

word w*= "sweden"

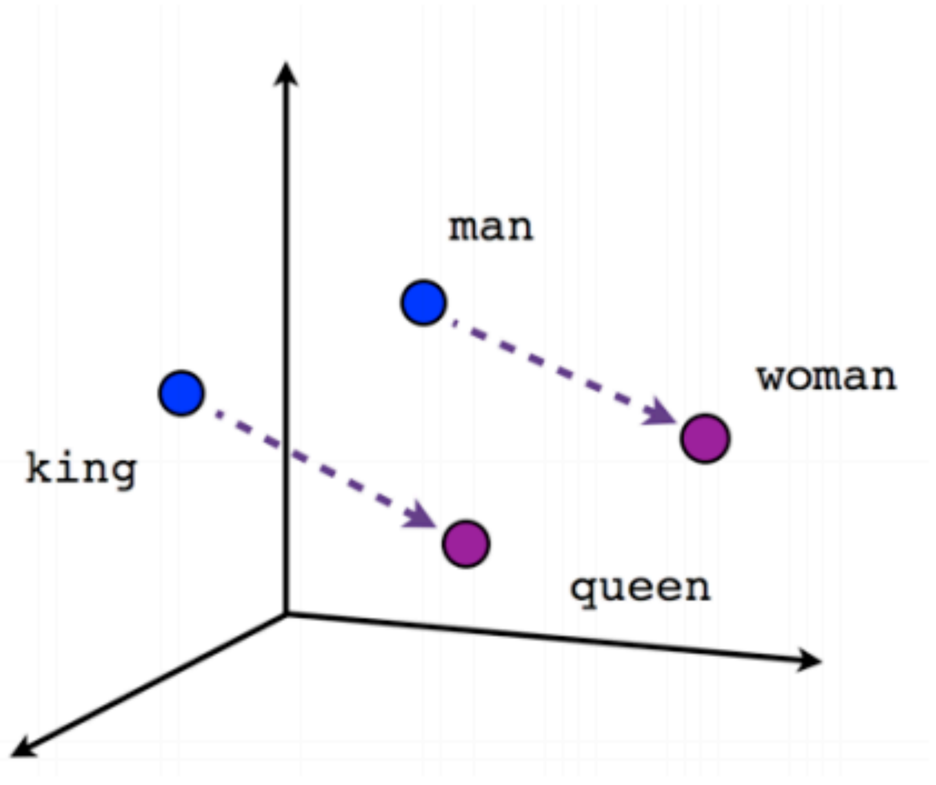
$$\arg\max_{w \in V} \cos(e(w), e(w^*))$$

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

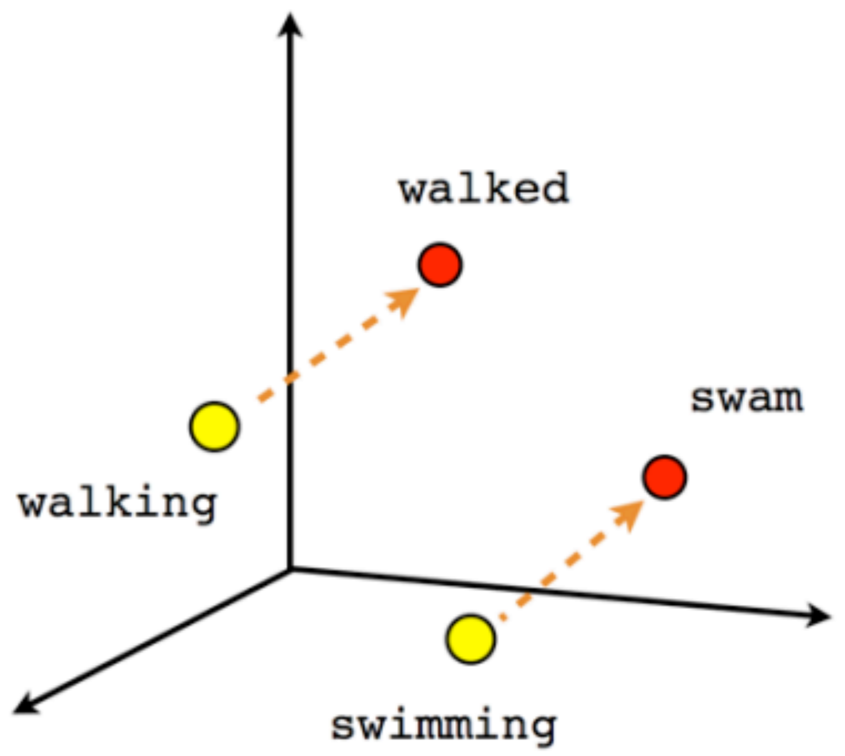
$\cos(u, v)$ ranges between -1 and 1

Word embeddings

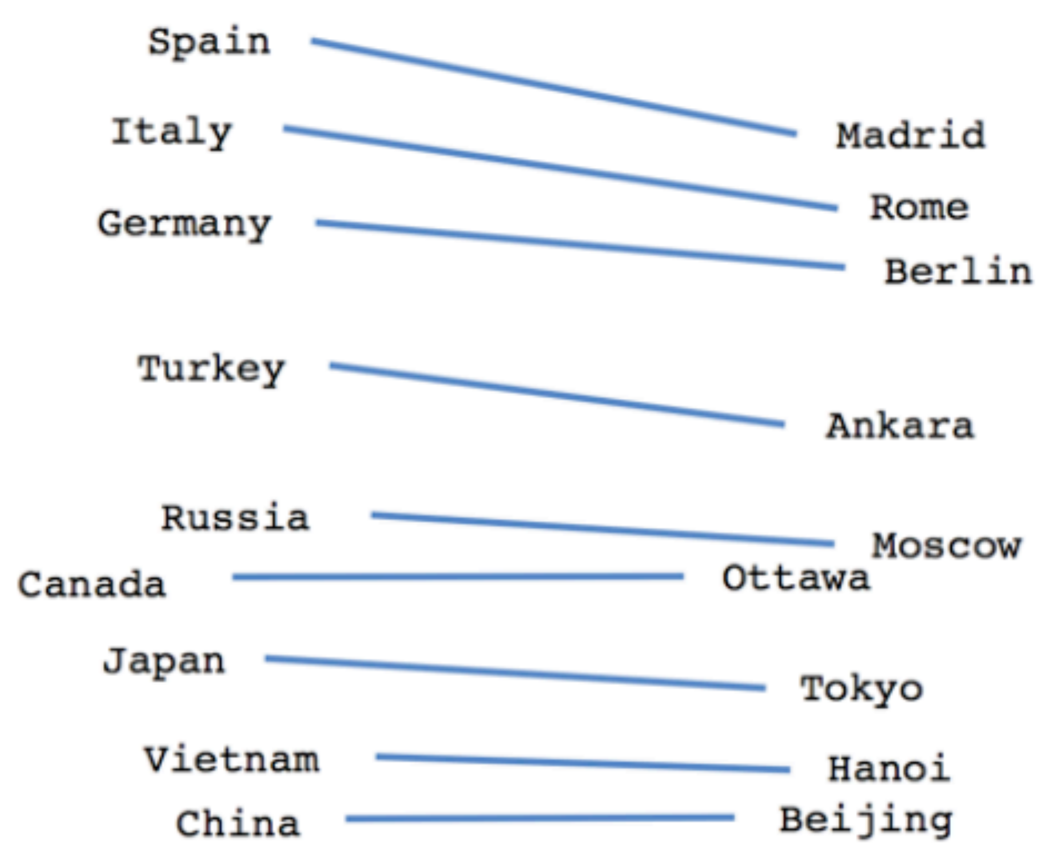
- They have some other nice properties too!



Male-Female



Verb tense



Country-Capital

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

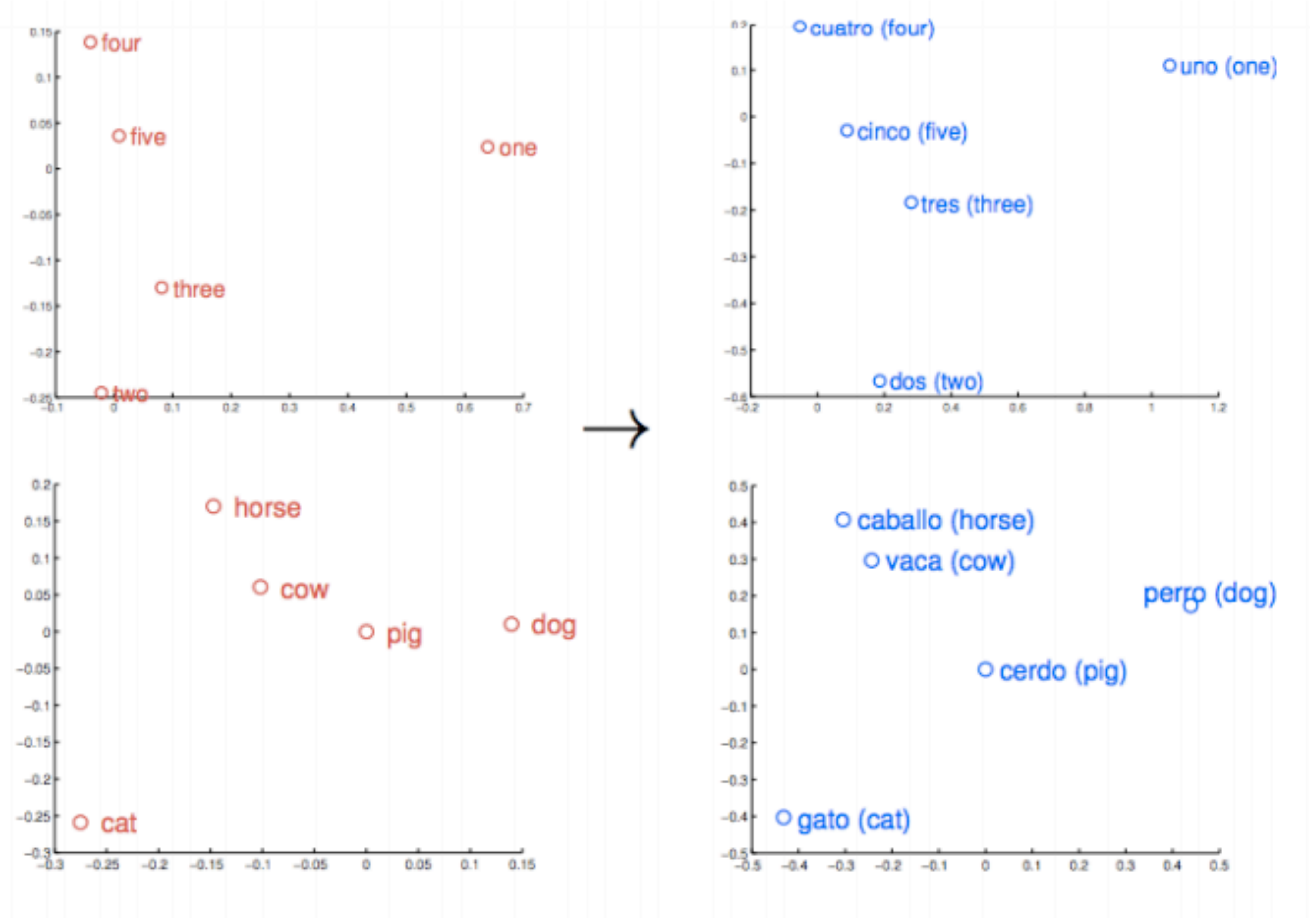
$$v_{\text{Paris}} - v_{\text{France}} \approx v_{\text{Rome}} - v_{\text{Italy}}$$

Word analogy test: $a : a* :: b : b*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

Word embeddings

- They have some other nice properties too!

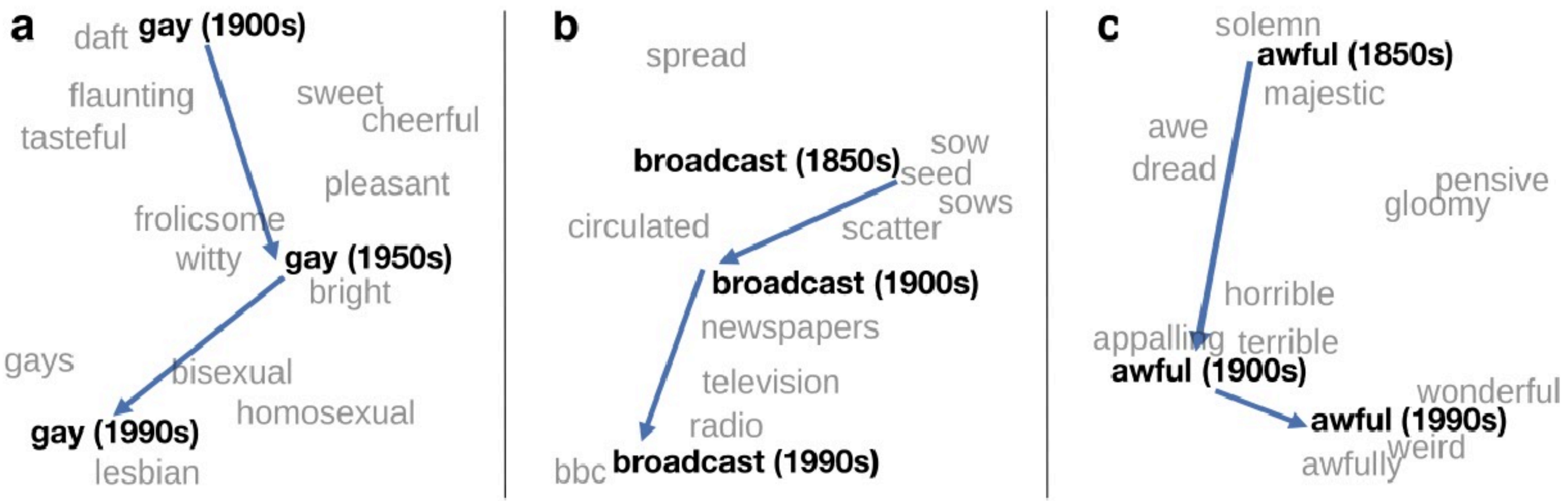


(Mikolov et al, 2013): Exploiting Similarities among Languages for Machine Translation

Embeddings as a window onto historical semantics

Train embeddings on different decades of historical text to see meanings shift

~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.

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 *NeurIPS*, 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

Algorithms that use embeddings as part of e.g., hiring searches for programmers, might lead to bias in hiring

Word embeddings: the learning problem

Learning vectors from text for representing words

- **Input**:
  - a large text corpus,
  - vocabulary V
  - vector dimension d (e.g., 300)
- **Output**: $f\colon\ V \to \mathbb{R}^d$

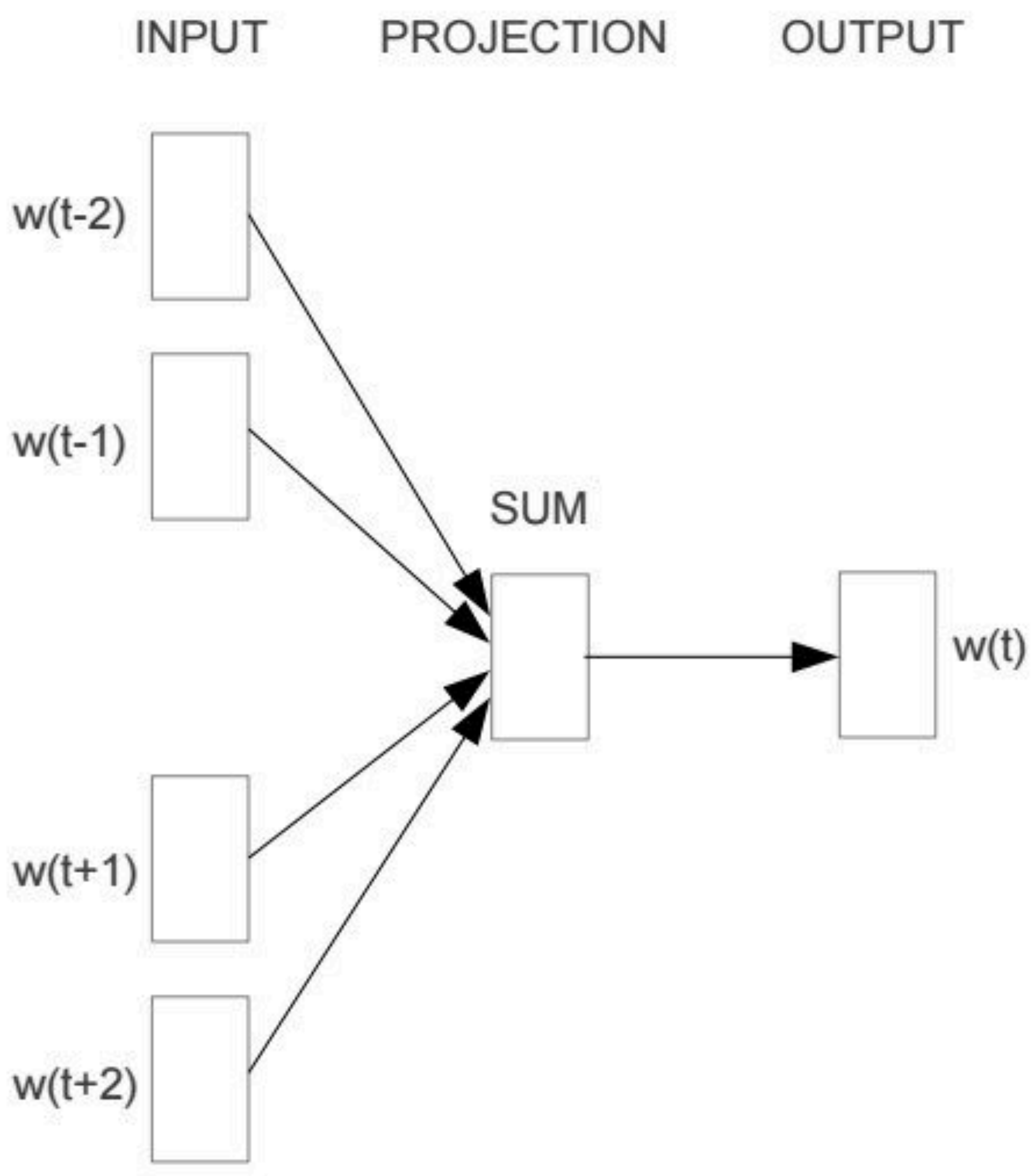
$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

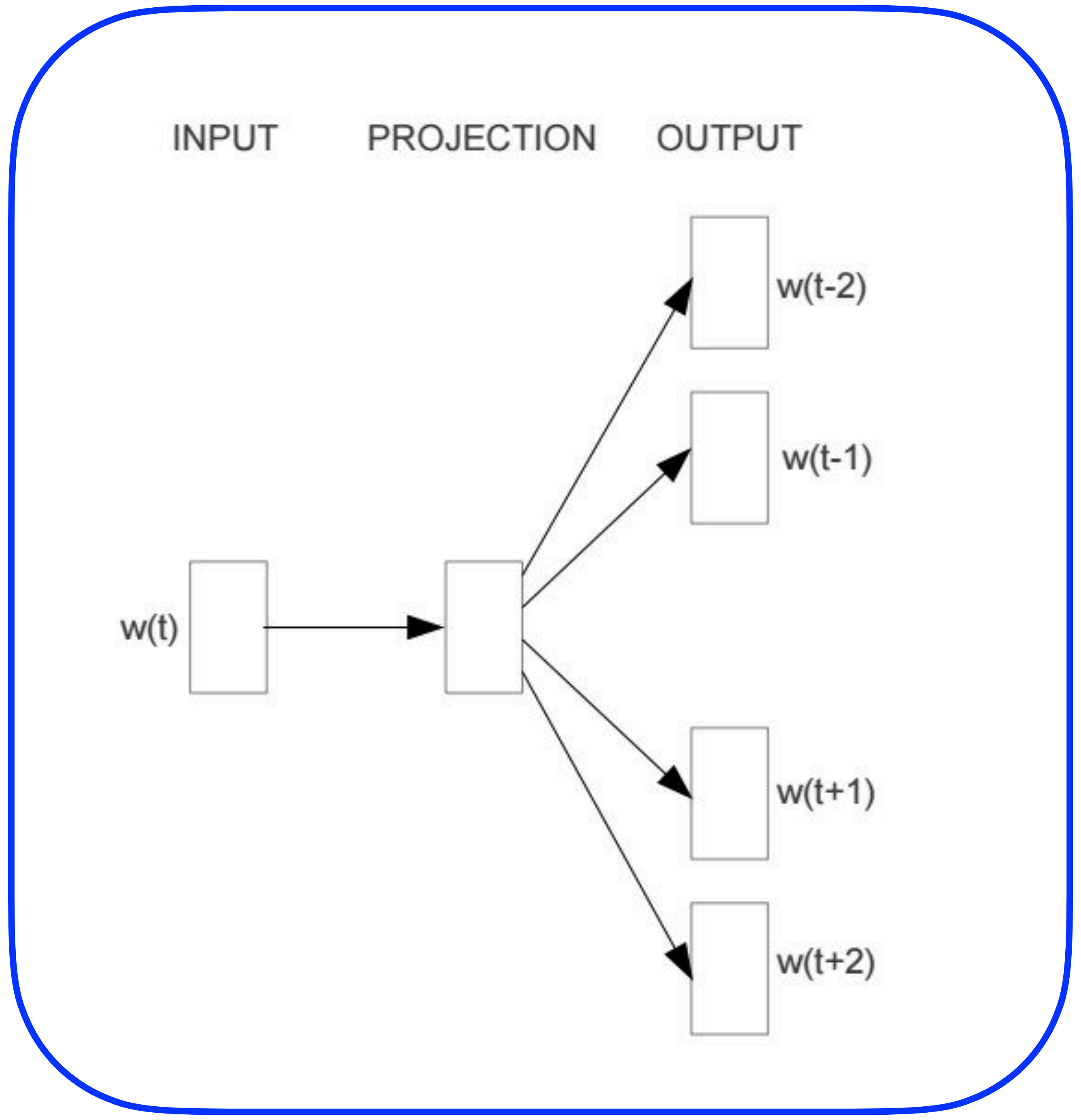
Word2vec

- (Mikolov et al 2013a): Efficient Estimation of Word Representations in Vector Space
- (Mikolov et al 2013b): Distributed Representations of Words and Phrases and their Compositionality

Tomáš Mikolov



Continuous Bag of Words (CBOW)



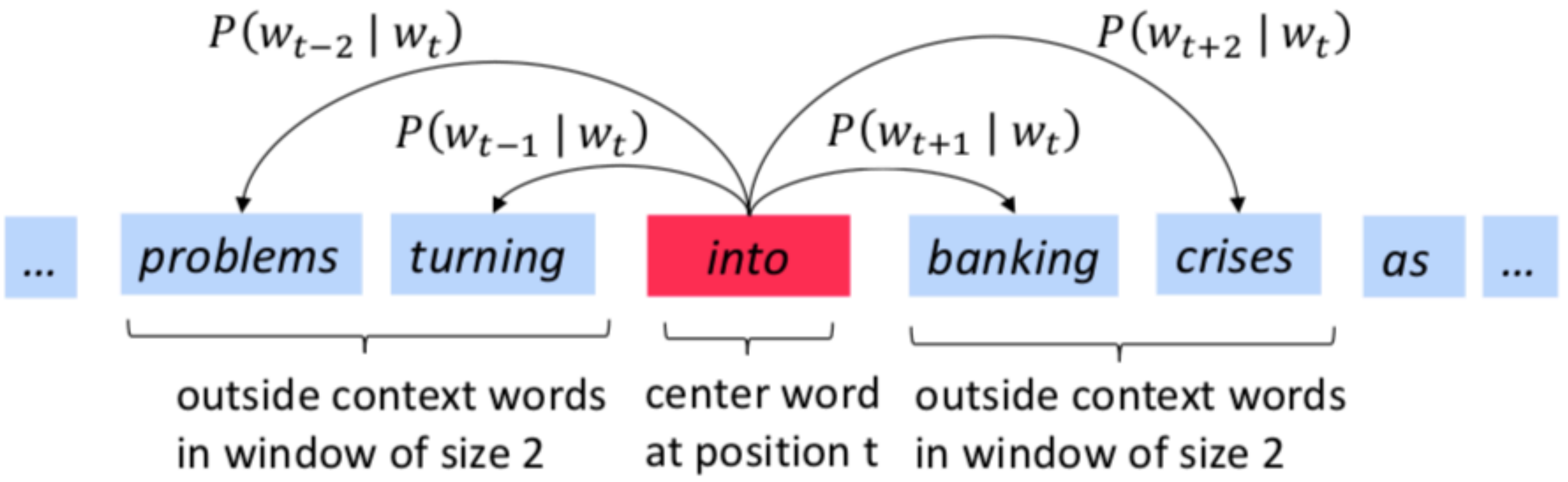
Skip-gram

Skip-gram

- Key idea: Use each word to predict other words in its context ← A classification problem!
- $P(b \mid a)$ = given the center word is a, what is the probability that b is a context word?
- Context: a fixed window of size 2m (m = 2 in the example)

$P(\cdot \mid a)$ is a probability distribution defined over V:

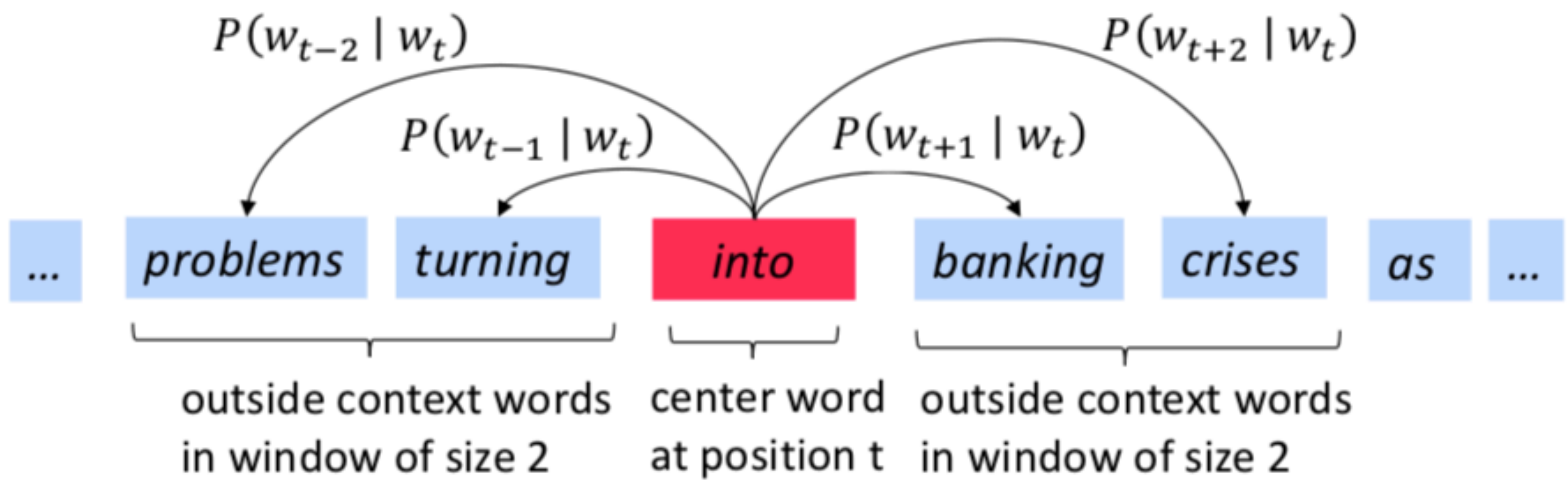
$$\sum_{w \in V} P(w \mid a) = 1$$



Key trick: We turn unlabeled text into supervised learning data

Skip-gram: Intuition (1/2)

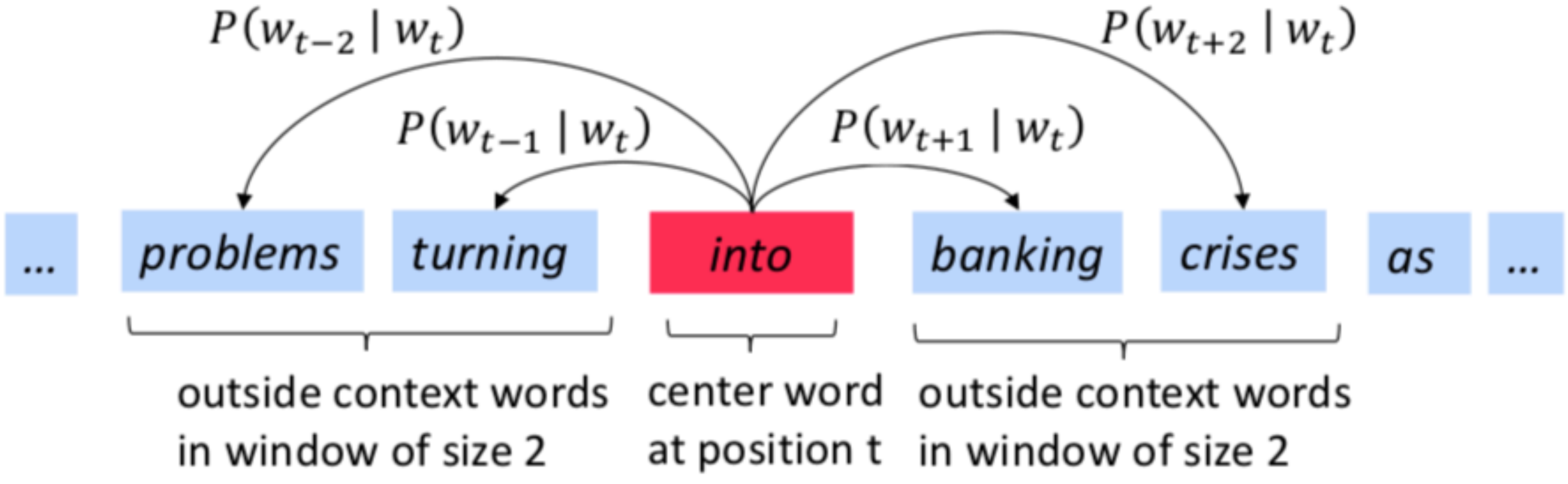
Key trick: We turn unlabeled text into supervised learning data

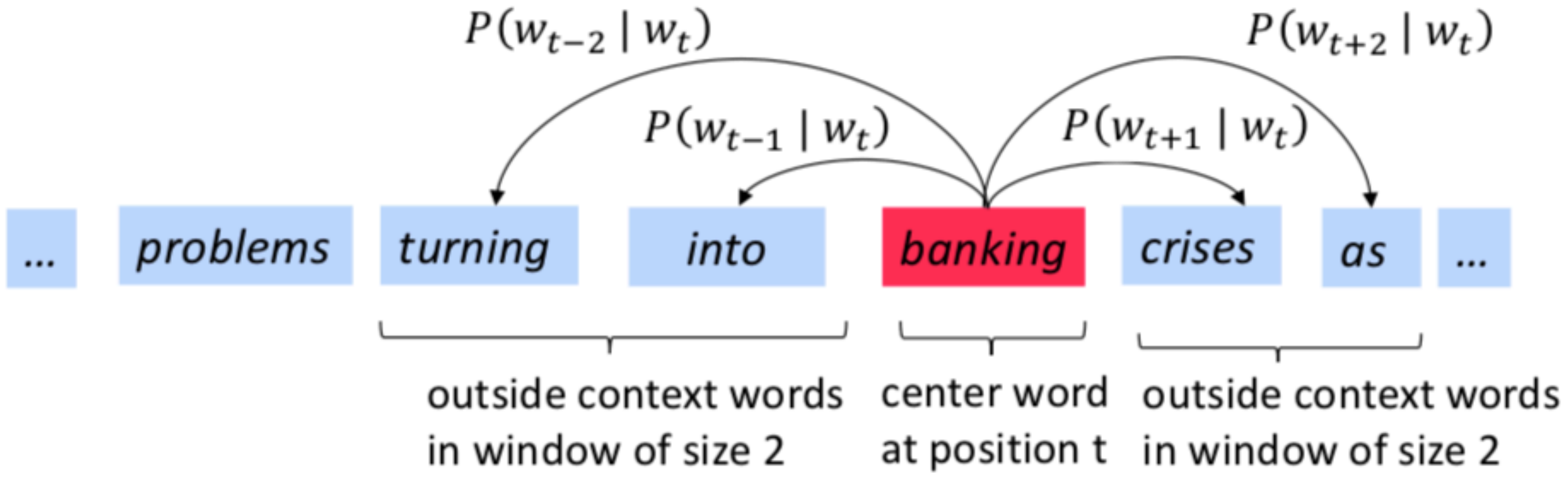


Convert into training data:
(into, problems)
(into, turning)
(into, banking)
(into, crises)

Skip-gram: Intuition (1/2)

Key trick: We turn unlabeled text into supervised learning data

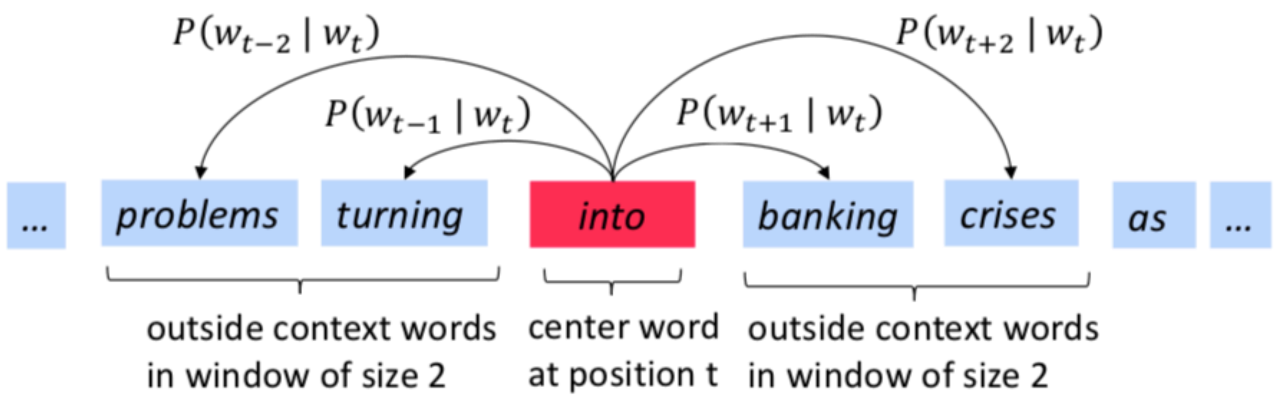


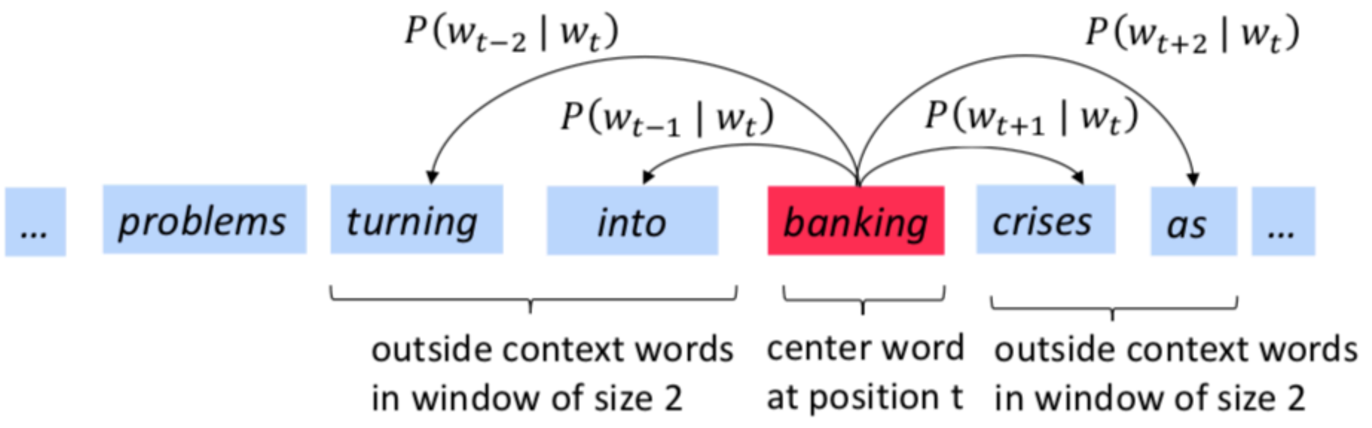


Convert into training data:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
…

Skip-gram: Intuition (2/2)

Key trick: We turn unlabeled text into supervised learning data





Convert into training data:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
...

Our goal is to find parameters that can maximize

$$\underline{P(\text{problems} \mid \text{into}) \times P(\text{turning} \mid \text{into}) \times P(\text{banking} \mid \text{into}) \times P(\text{crises} \mid \text{into}) \times}$$

$$\underline{P(\text{turning} \mid \text{banking}) \times P(\text{into} \mid \text{banking}) \times P(\text{crises} \mid \text{banking}) \times P(\text{as} \mid \text{banking}) \ldots}$$

Skim-gram: Objective function

- For each position $t = 1,2,\ldots T$, predict context words within context size m, given center word w_t:

$$\prod_{-m \le j \le m, j \ne 0} P(w_{t+j} \mid w_t; \theta)$$

Skim-gram: Objective function

- For each position $t = 1,2,\ldots T$, predict context words within context size m, given center word w_t:

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

all the parameters to be optimized

Skim-gram: Objective function

- For each position $t = 1,2,\ldots T$, predict context words within context size m, given center word w_t:

all the parameters to be optimized

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} P(w_{t+j} \mid w_t; \theta)$$

- It is equivalent to minimizing the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j} \mid w_t; \theta)$$

How to define $P(w_{t+j} \mid w_t; \theta)$?

- Use two sets of vectors for each word in the vocabulary

$$\mathbf{u}_a \in \mathbb{R}^d \quad \text{: vector for center word } a, \forall a \in V$$

$$\mathbf{v}_b \in \mathbb{R}^d \quad \text{: vector for context word } b, \forall b \in V$$

- Use inner product $\mathbf{u}_a \cdot \mathbf{v}_b$ to measure how likely word a appears with context word b

$$P(w_{t+j} \mid w_t) = \frac{\exp\left(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}}\right)}{\sum_{k \in V} \exp\left(\mathbf{u}_{w_t} \cdot \mathbf{v}_k\right)}$$

Does this term seem familiar?

… vs multinominal logistic regression

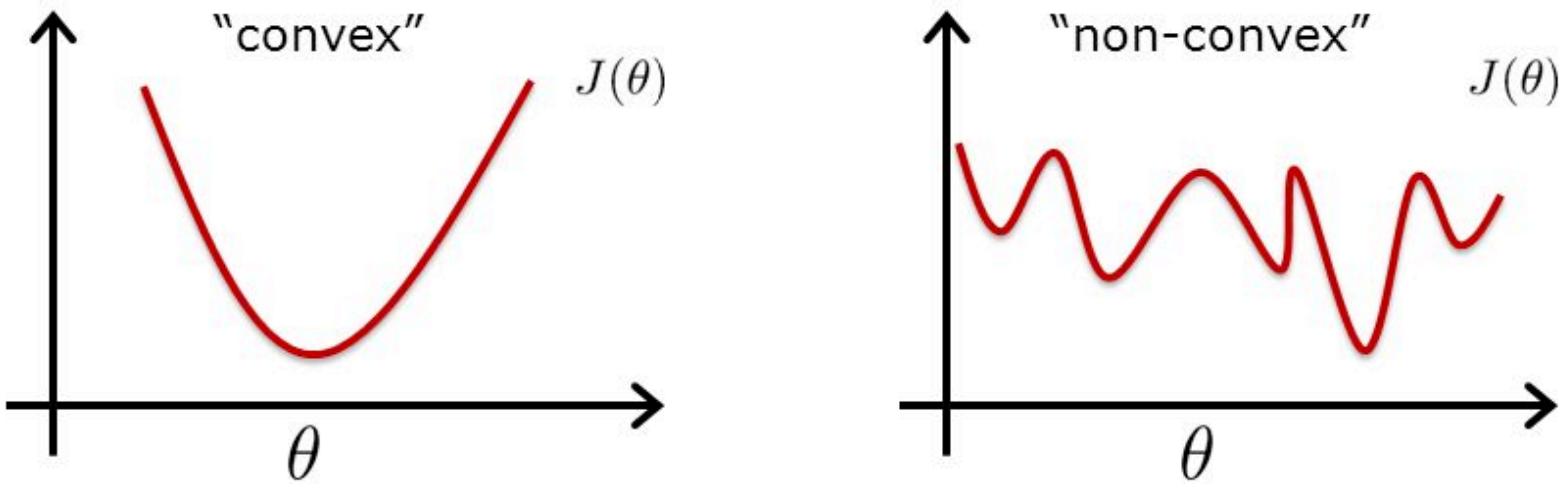
- Essentially a |V|-way classification problem
- Recall: multinomial logistic regression:

$$P(y = c \mid x) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x} + b_c)}{\sum_{j=1}^{m} \exp(\mathbf{w}_j \cdot \mathbf{x} + b_j)}$$

$$\boxed{P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}}$$

- If we fix $\mathbf{u}_{w_t}$, it is reduced to a multinomial logistic regression problem.
- However, since we have to learn both $\mathbf{u}$ and $\mathbf{v}$ together, the training objective is non-convex.

… vs multinominal logistic regression



- It is hard to find a global minimum
- But can still use stochastic gradient descent to optimize θ:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta J(\theta)$$

Important note

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

- In this formulation, we don't care about the classification task itself like we do for the logistic regression model we saw previously.
- The key point is that the parameters used to optimize this training objective—when the training corpus is large enough—can give us very good representations of words (following the principle of **distributional hypothesis**)!

Quick quiz

How many parameters does this model have (i.e. what is size of θ)?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

(a) $d|V|$

(b) $2d|V|$

(c) $2m|V|$

(d) $2md|V|$

V := Vocabulary

d := dimension of embedding

m := size of context window

The answer is (b).
Each word has two d-dimensional vectors, so it is $2 \times |V| \times d$.

Word2vec formulation

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

Q: Why do we need two vectors for each word?

- Because one word is not likely to appear in its own context window, e.g., $P(\text{dog} \mid \text{dog})$ should be low. If we use one set of vectors only, it essentially needs to minimize $\mathbf{u}_{\text{dog}} \cdot \mathbf{u}_{\text{dog}}$

Q: Which set of vectors are used as word embeddings?

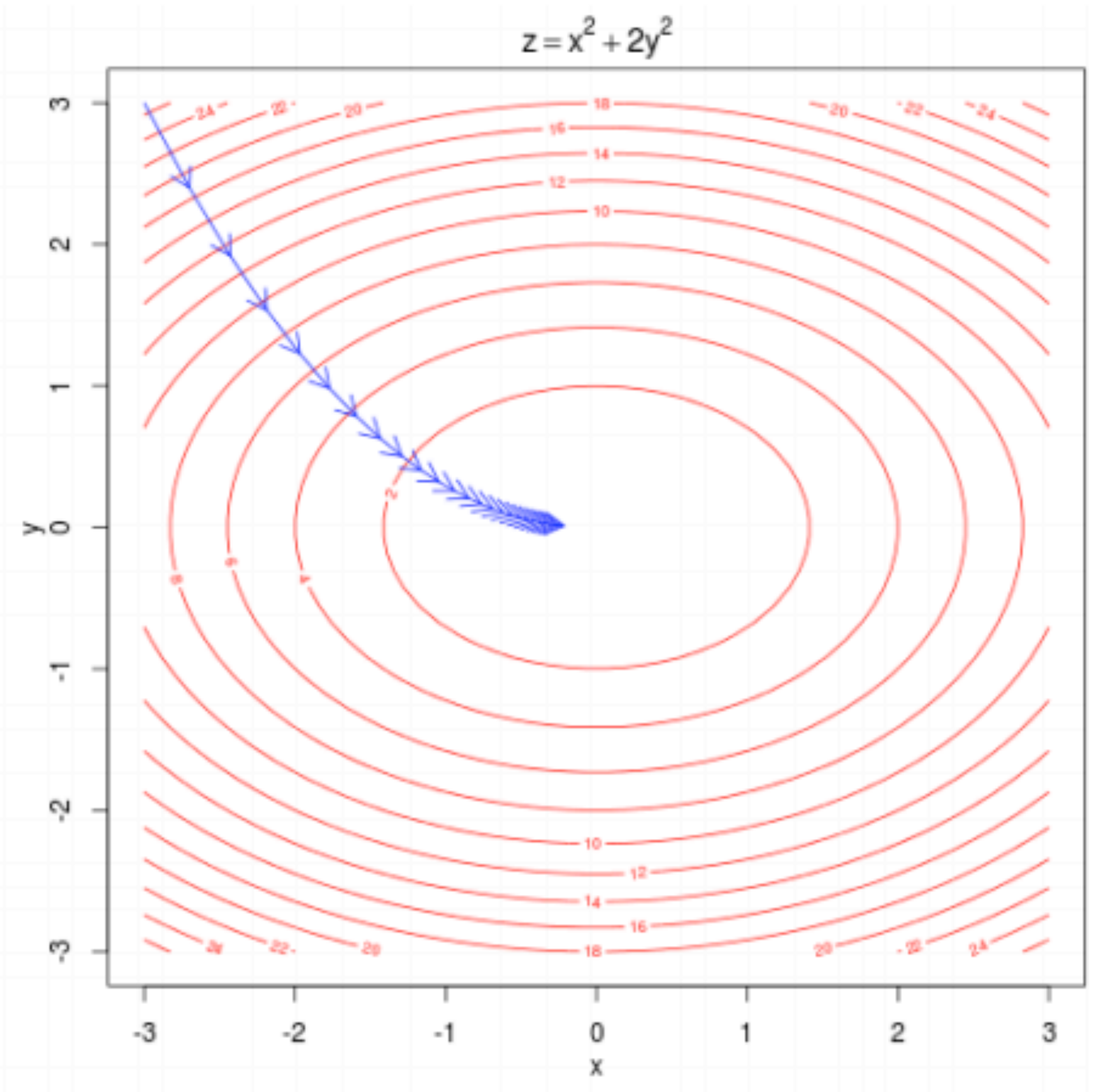
- This is an empirical question. Typically just $\mathbf{u}_w$ but you can also concatenate the two vectors..

How to train this model?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

- To train such a model, we need to compute the vector gradient $\nabla_\theta J(\theta) = ?$
- Remember that θ represents all $2d|V|$ model parameters, in one vector.

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix}$$



Warm up: Vectorized gradients

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{a}$$

$$\mathbf{x}, \mathbf{a} \in \mathbb{R}^n$$

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{a}$$

$$f = x_1 a_1 + x_2 a_2 + \ldots + x_n a_n$$

$$\frac{\partial f}{\partial \mathbf{x}} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}]$$

Warm up: Vectorized gradients exercises

Let $f = \exp(\mathbf{w} \cdot \mathbf{x})$, what is the value of $\dfrac{\partial f}{\partial \mathbf{x}}$? (Assume $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$)

(a) $\mathbf{w}$

(b) $\exp(\mathbf{w} \cdot \mathbf{x})$

(c) $\exp(\mathbf{w} \cdot \mathbf{x})\mathbf{w}$

(d) $\mathbf{x}$

Hints:

- The derivative of $\exp(\mathbf{z})$ is $\exp(\mathbf{z})$
- The derivative of $\mathbf{w} \cdot \mathbf{x}$ with respect to $\mathbf{x}$ is $\mathbf{w}$

The answer is (c).

$$\frac{\partial}{\partial x_i} = \frac{\exp(\sum_{k=1}^{n} w_i x_i)}{\partial x_i} = \exp(\sum_{k=1}^{n} w_i x_i) w_i$$

Let's compute gradients for word2vec

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

Consider one pair of center/context words (t, c):

$$y = -\log\left(\frac{\exp(\mathbf{u}_t\cdot\mathbf{v}_c)}{\sum_{k\in V}\exp(\mathbf{u}_t\cdot\mathbf{v}_k)}\right)$$

We need to compute the gradient of y with respect to

$\mathbf{u}_t$ and $\mathbf{v}_k, \forall k \in V$

Overall algorithm

- Input: text corpus, embedding size d, vocabulary V, context size m
- Initialize $\mathbf{u}_i, \mathbf{v}_i$ randomly $\forall i \in V$
- Run through the training corpus and for each training instance (*t*, *c*):
  - Update $\mathbf{u}_t \leftarrow \mathbf{u}_t - \eta \frac{\partial y}{\partial \mathbf{u}_t}$; $\quad \frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k$
  - Update $\mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v}_k}, \forall k \in V$; $\quad \frac{\partial y}{\partial \mathbf{v}_k} = \begin{cases} (P(k \mid t) - 1)\mathbf{u}_t & k = c \\ P(k \mid t)\mathbf{u}_t & k \neq c \end{cases}$

Q: Can you think of any issues with this algorithm?

Overall algorithm: Problem

Problem: every time you get one pair of (t, c), you need to update v_k with all the words in the vocabulary! This is very expensive computationally.

$$\frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k \quad ; \quad \frac{\partial y}{\partial \mathbf{v}_k} = \begin{cases} (P(k \mid t) - 1)\ \mathbf{u}_t & k = c \\ P(k \mid t)\ \mathbf{u}_t & k \neq c \end{cases}$$

$$P(k \mid t) = \frac{\exp\left(\mathbf{u}_t \cdot \mathbf{v}_k\right)}{\sum_{j \in V} \exp\left(\mathbf{u}_t \cdot \mathbf{v}_j\right)}$$

Key question: Do we really need to compare the center word against every word in the vocabulary every time?

Jan 27 lecture starts from here

CS 288 Advanced Natural Language Processing

Course website: cal-cs288.github.io/sp26

Ed: edstem.org/us/join/XvztdK

- Class starts at 15:40!

Announcements

- Enrollment:
  - Batch 1 codes sent. Please use your code by **tomorrow (Wednesday)**.
  - Batch 2 will be sent on **Thursday**.
- Everyone (including students not enrolled yet) should now have access to Ed and Gradescope
- **Assignment 1 is out! (Due in two weeks)**
  - Part 1: *n*-gram LM (our previous lecture)
  - Part 2: Text classification (today's lecture) — involves hidden test set
  - Submit on Gradescope (max 5 submissions)
- Please start thinking about the course project!
  - Check out the descriptions on the course website.
  - Team registration / match request is due 02/10. Teams must have 3 students.
  - Feel free to use Ed to find teammates.
- Today's lecture plan: Finish Word embeddings (20min) → Text classification (60min)

Recap: Representing a word

One-hot encoding

$$v_{\text{cat}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{dog}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{the}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \cdots \end{bmatrix} \quad v_{\text{language}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \cdots \end{bmatrix}$$

Count vectors

	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	...

Dense vectors

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix} \quad v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

In particular, learned dense vectors (through shallow neural networks) are called **word embeddings**!

Recap: Distributional semantics

Words that occur in similar **contexts** tend to have similar meanings

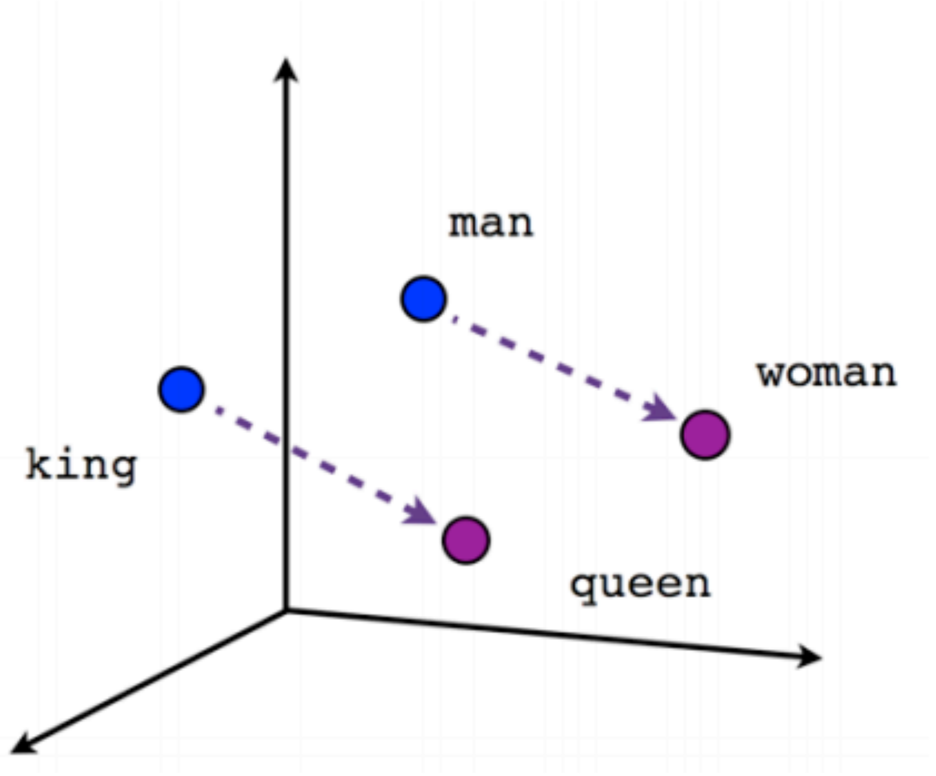
J.R.Firth 1957

- “You shall know a word by the company it keeps”
- One of the most successful ideas of modern statistical NLP!

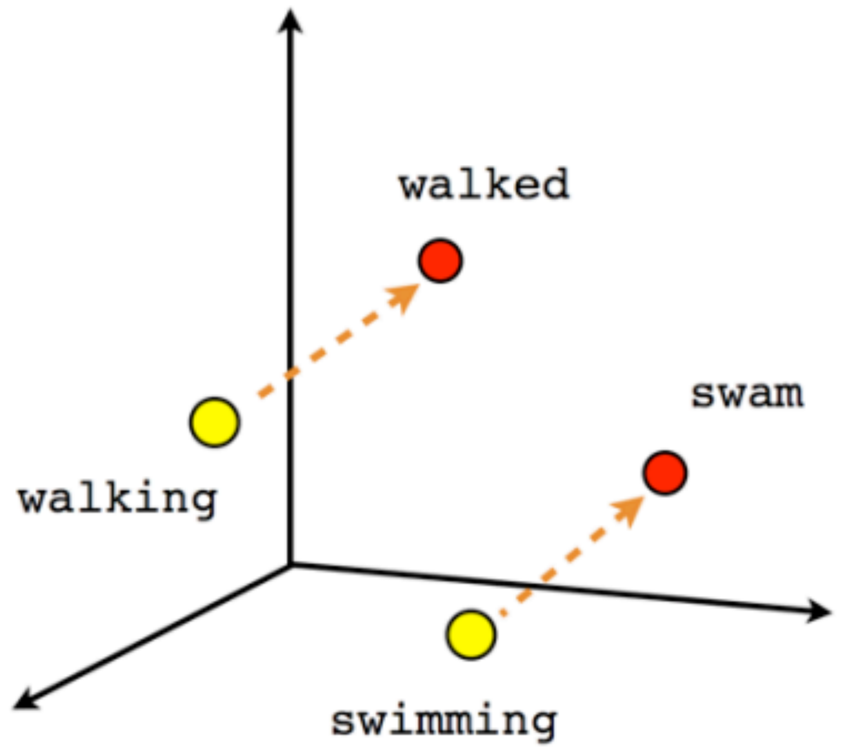
…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

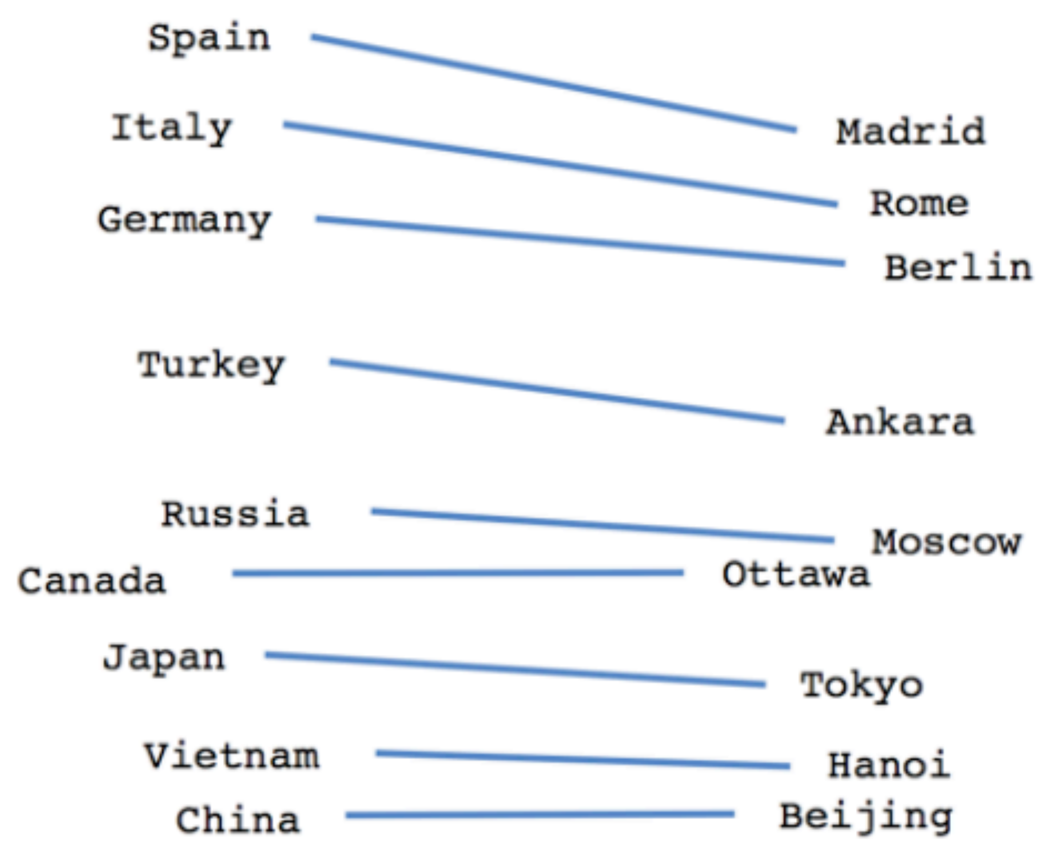
Recap: Nice properties of word embeddings



Male-Female



Verb tense



Country-Capital

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

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

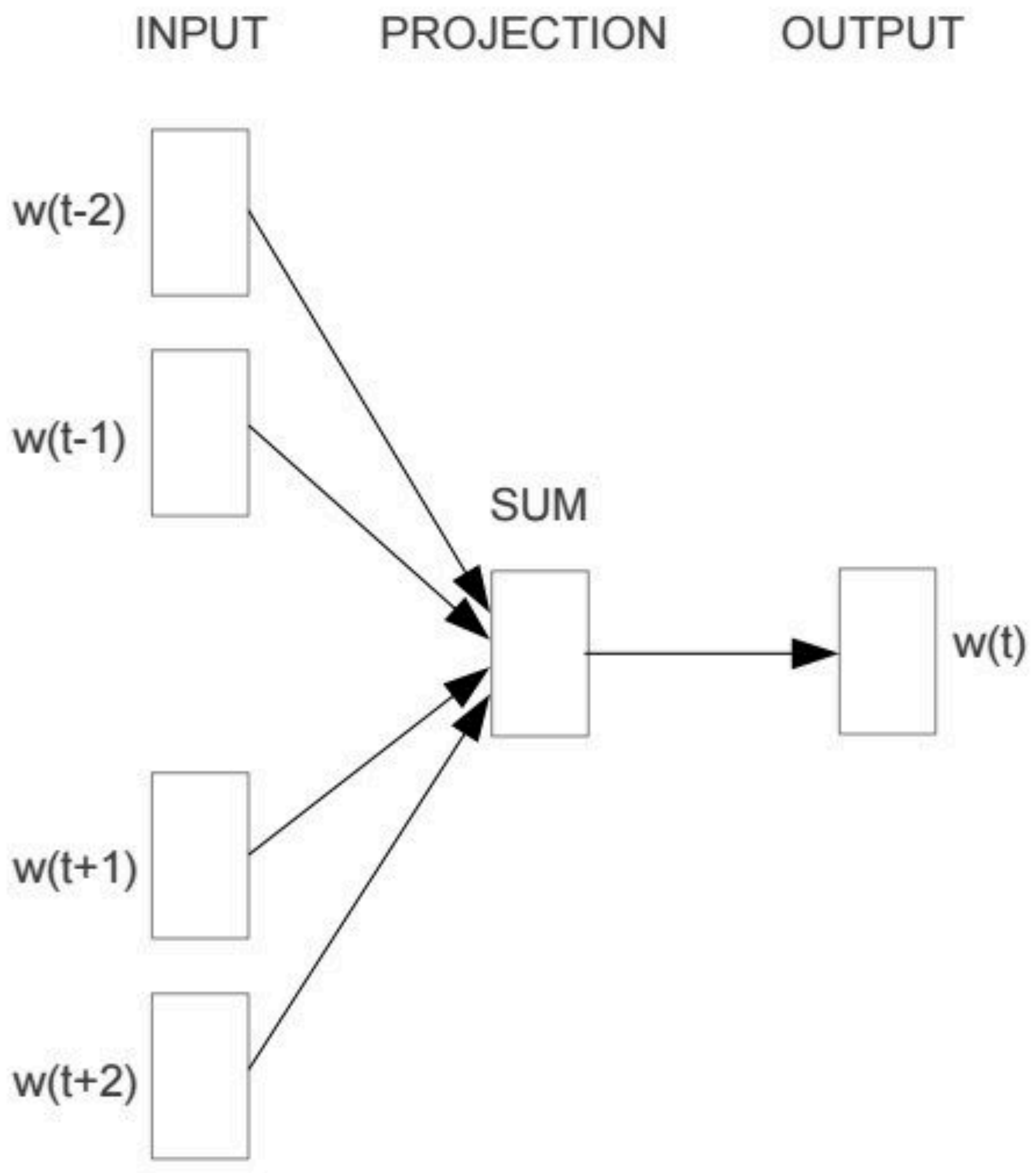
Word analogy test: $a : a* :: b : b*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

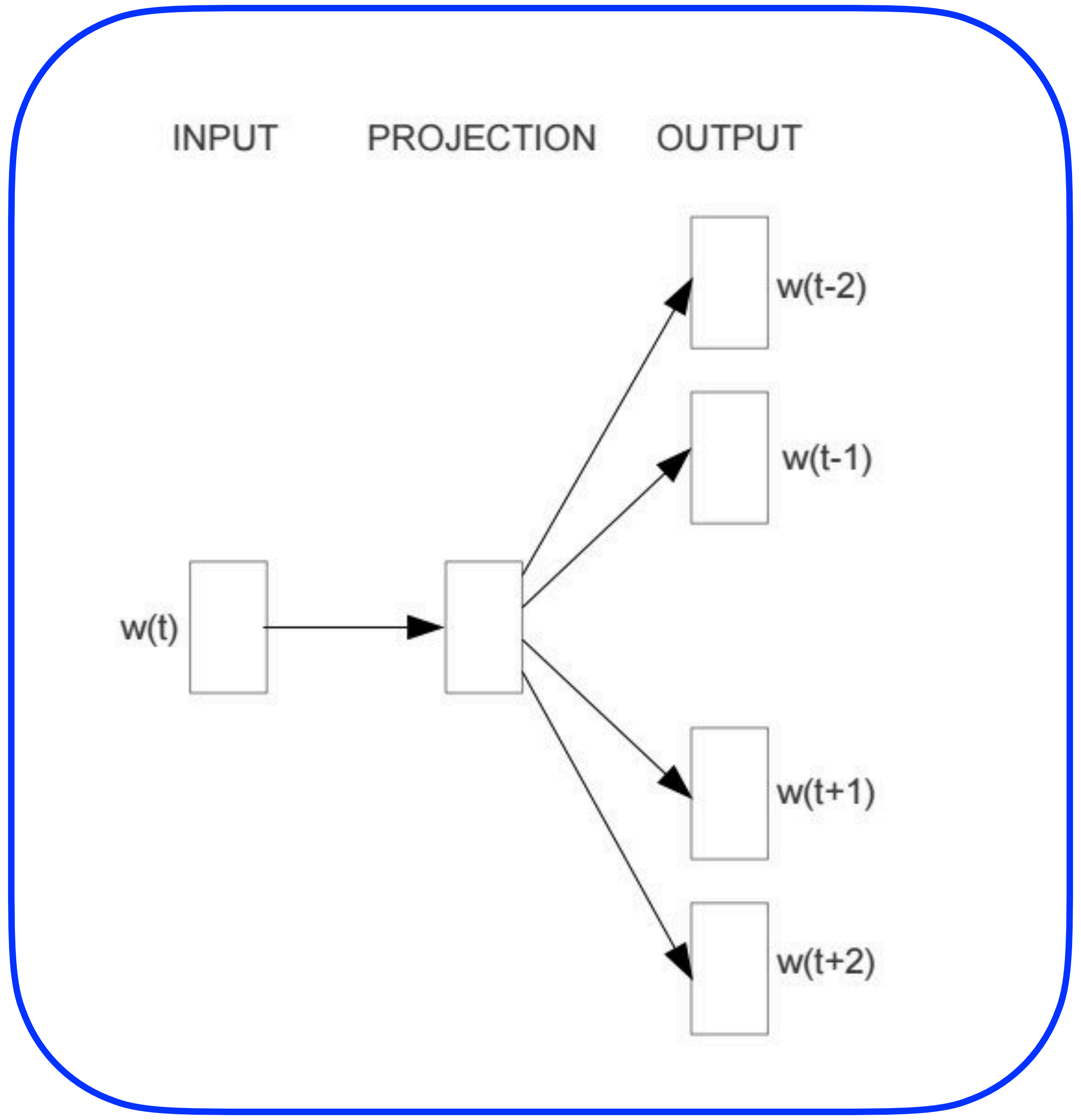
Recap: Word2vec

- (Mikolov et al 2013a): Efficient Estimation of Word Representations in Vector Space
- (Mikolov et al 2013b): Distributed Representations of Words and Phrases and their Compositionality

Tomáš Mikolov



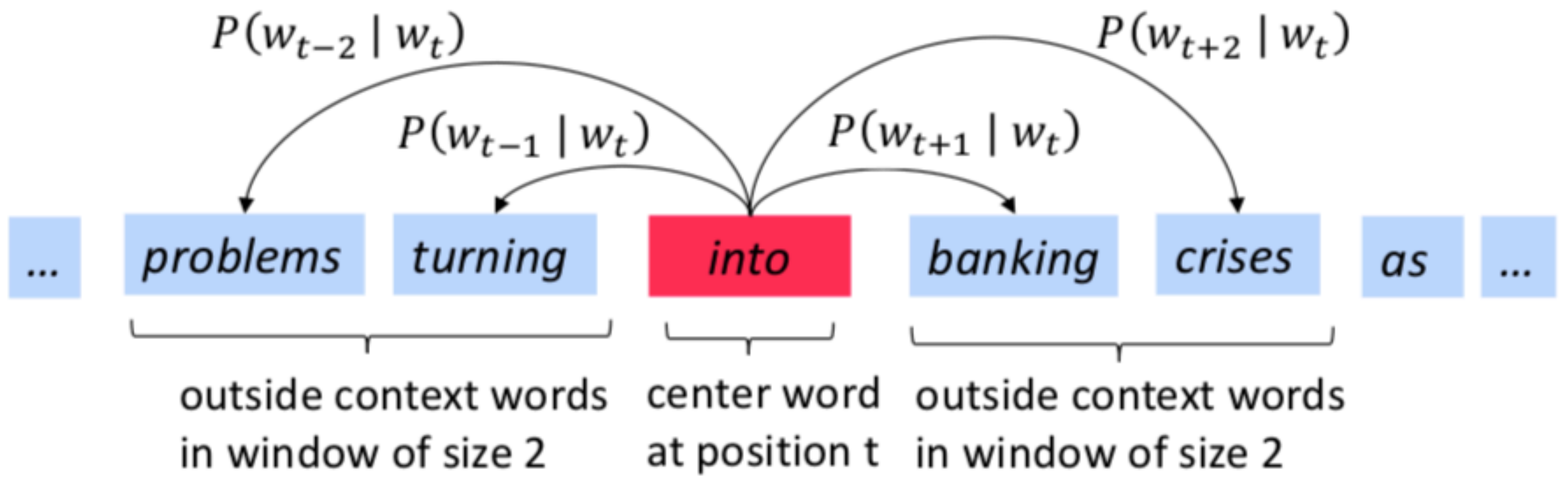
Continuous Bag of Words (CBOW)



Skip-gram

Recap: Skip-gram

- Key idea: Use each word to predict other words in its context



Our goal is to find parameters that can maximize

$$P(\text{problems} \mid \text{into}) \times P(\text{turning} \mid \text{into}) \times P(\text{banking} \mid \text{into}) \times P(\text{crises} \mid \text{into})$$

Recap: Skip-gram

- Key idea: Use each word to predict other words in its context

Key tricks:

- We turn unlabeled text into supervised learning data
- We train a model on a prediction task, not for the sake of the prediction, but to learn high-quality representations (word embeddings) — *recurring themes in pre-training*

Our goal is to find parameters that can maximize

$$\underline{P(\text{problems} \mid \text{into}) \times P(\text{turning} \mid \text{into}) \times P(\text{banking} \mid \text{into}) \times P(\text{crises} \mid \text{into}) \times}$$

$$\underline{P(\text{turning} \mid \text{banking}) \times P(\text{into} \mid \text{banking}) \times P(\text{crises} \mid \text{banking}) \times P(\text{as} \mid \text{banking})\ldots}$$

Skim-gram: Objective function

- We maximize $\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$

- Alternatively, we minimize $J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$

- Here, the conditional probability is represented as $P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$

Problem: We need to compute the gradient of y with respect to $\mathbf{u}_t$ and $\mathbf{v}_k, \forall k \in V$

Why? Because we are essentially comparing the center word against every word in the vocabulary → Is it necessary?

Skip-gram with negative sampling (SGNS): Intuition

Instead of doing $|V|$-way classification, let's do binary classification

- Previously: Given a pair of words <t, c>, where c is the true context word for t, let's make sure the model predicts c from t among among all $|V|$ possible candidates.
- Now: Given a pair of words <t, c> where c may or may not be a true context word for t, let's classify whether c is a correct context word or not.

Similar to **binary logistic regression**, but we need to optimize $\mathbf{u}$ and $\mathbf{v}$ together.

$$P(y = 1 \mid t, c) = \sigma(\mathbf{u}_t \cdot \mathbf{v}_c) \qquad p(y = 0 \mid t, c') = 1 - \sigma(\mathbf{u}_t \cdot \mathbf{v}_{c'}) = \sigma(-\mathbf{u}_t \cdot \mathbf{v}_{c'})$$

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

$$\boxed{y = -\log\left(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)\right) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log\left(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j)\right)}$$

P(w): sampling according to the frequency of words

Understanding SGNS

In skip-gram with negative sampling (SGNS), how many parameters need to be updated in θ for every (t, c) pair? (K: # of negatives, d: dimension)

(a) Kd

(b) $2Kd$

(c) $(K+1)d$

(d) $(K+2)d$

$$y = -\log\left(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)\right) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log\left(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j)\right)$$

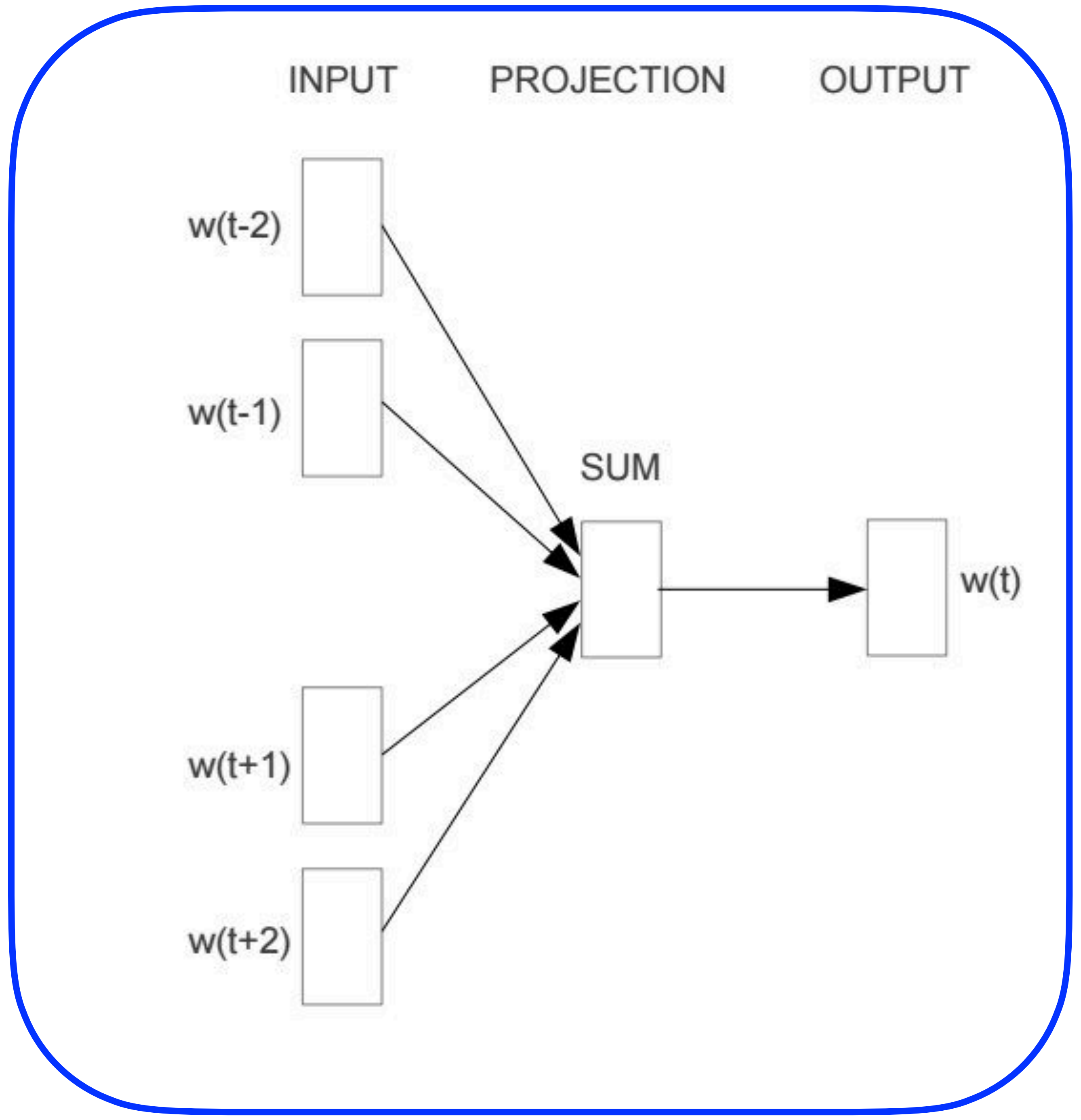
The answer is (d).

We need to calculate gradients with respect to $\mathbf{u}_t$ and (K + 1) $\mathbf{v}_i$ (one positive and K negatives).

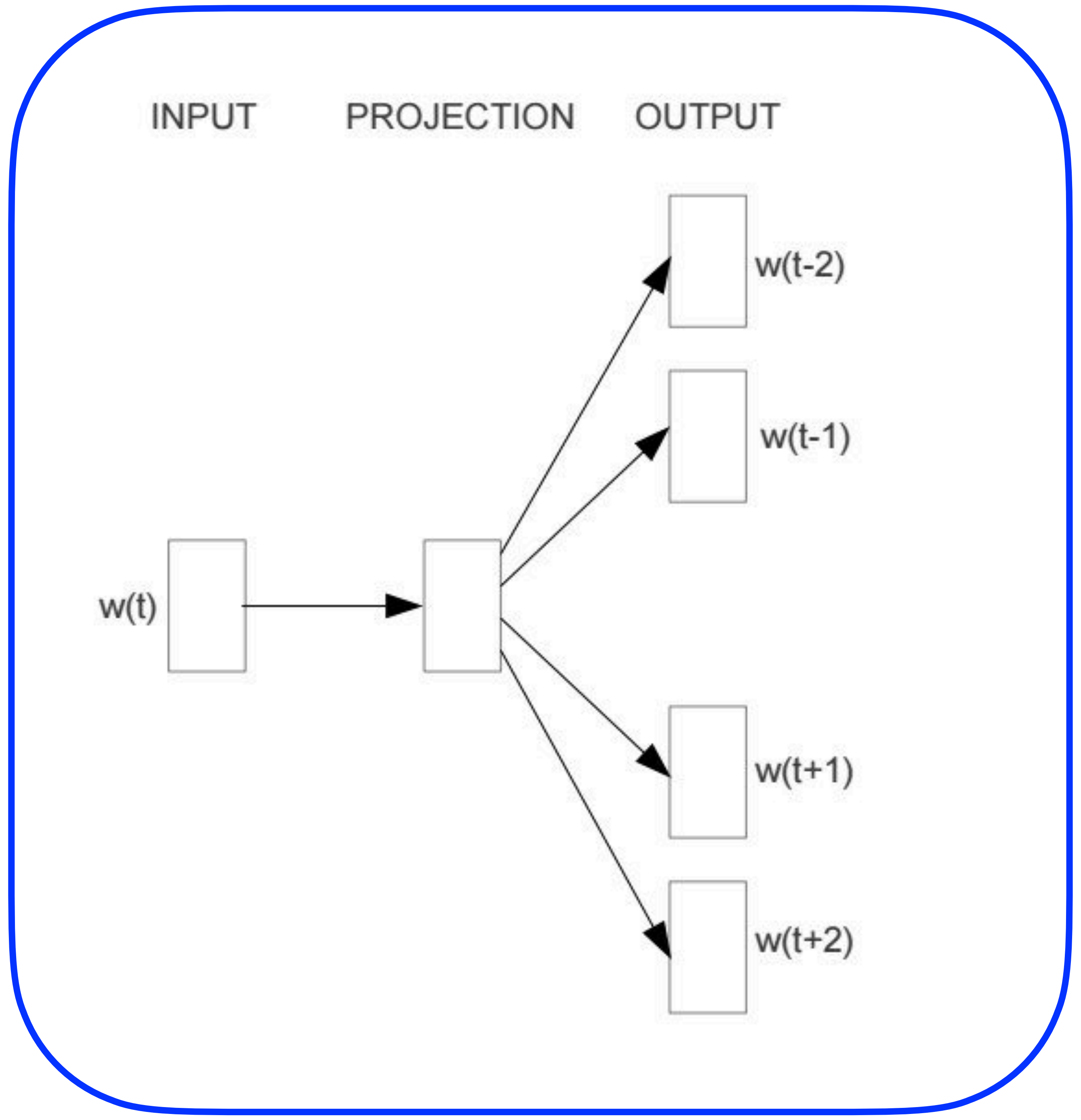
Word2vec

- (Mikolov et al 2013a): Efficient Estimation of Word Representations in Vector Space
- (Mikolov et al 2013b): Distributed Representations of Words and Phrases and their Compositionality

Tomáš Mikolov



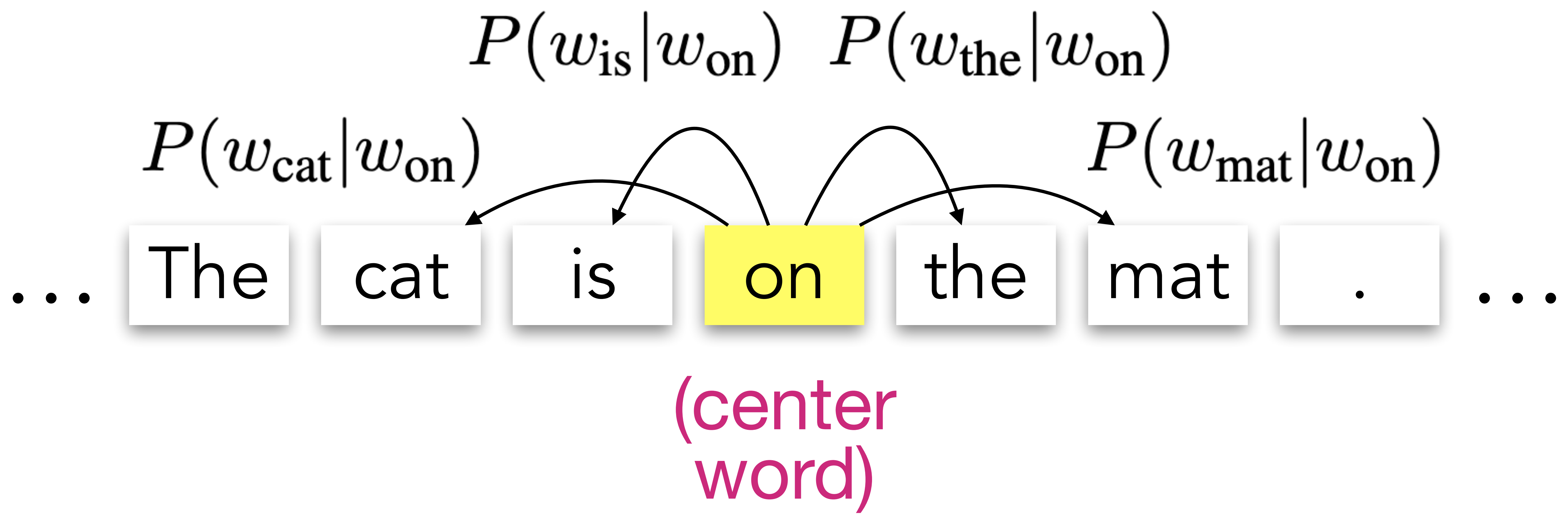
Continuous Bag of Words (CBOW)



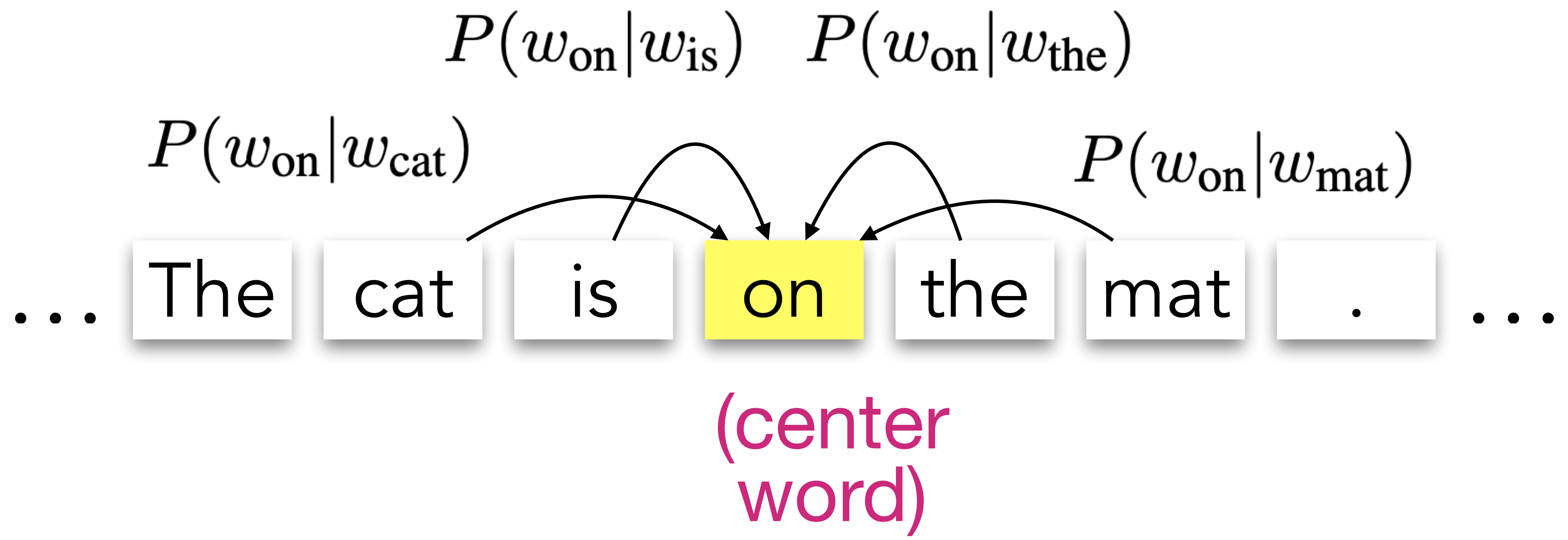
Skip-gram

Skip-gram vs. CBOW

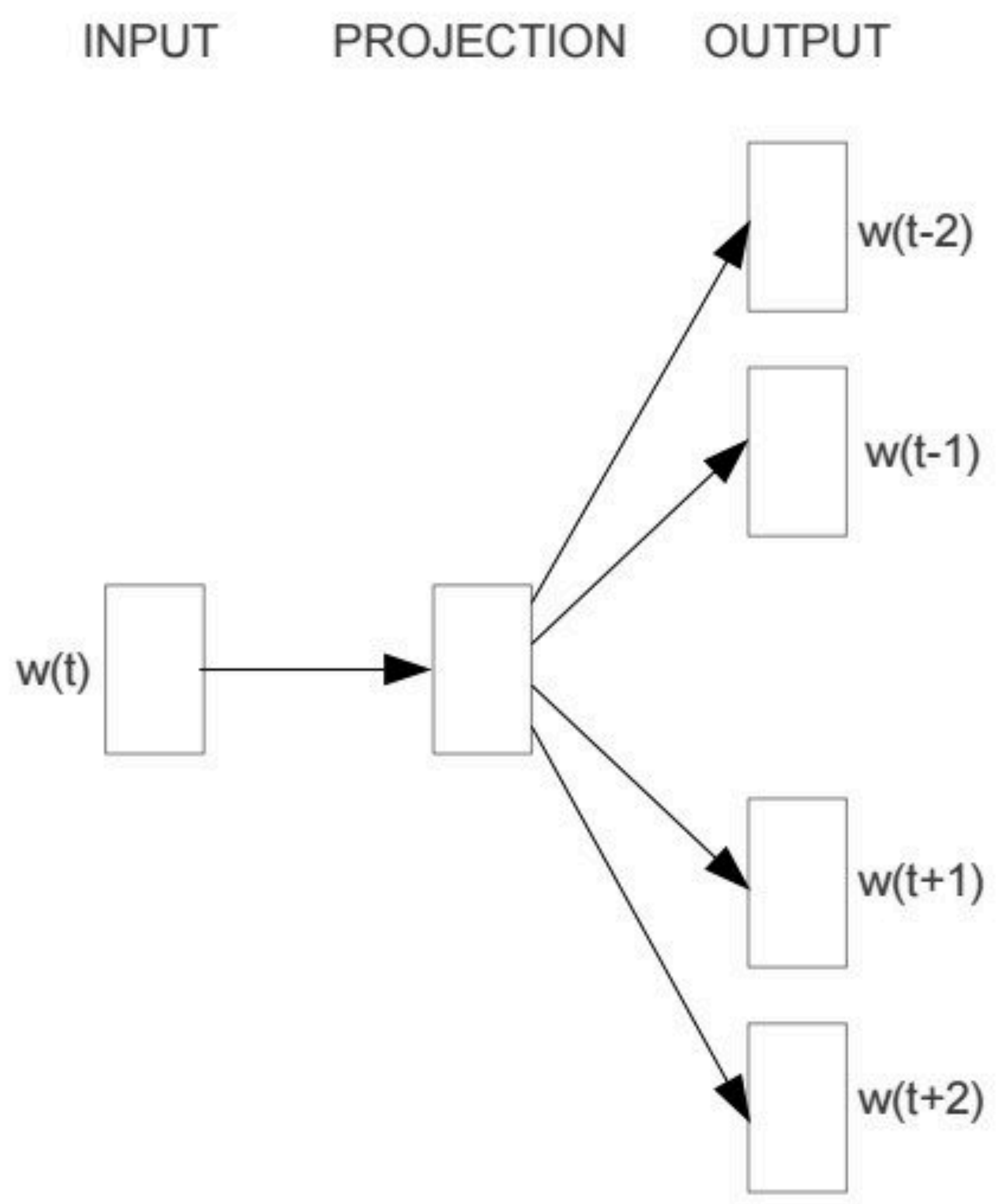
Skip-gram: Trained to predict the context words given a target word.



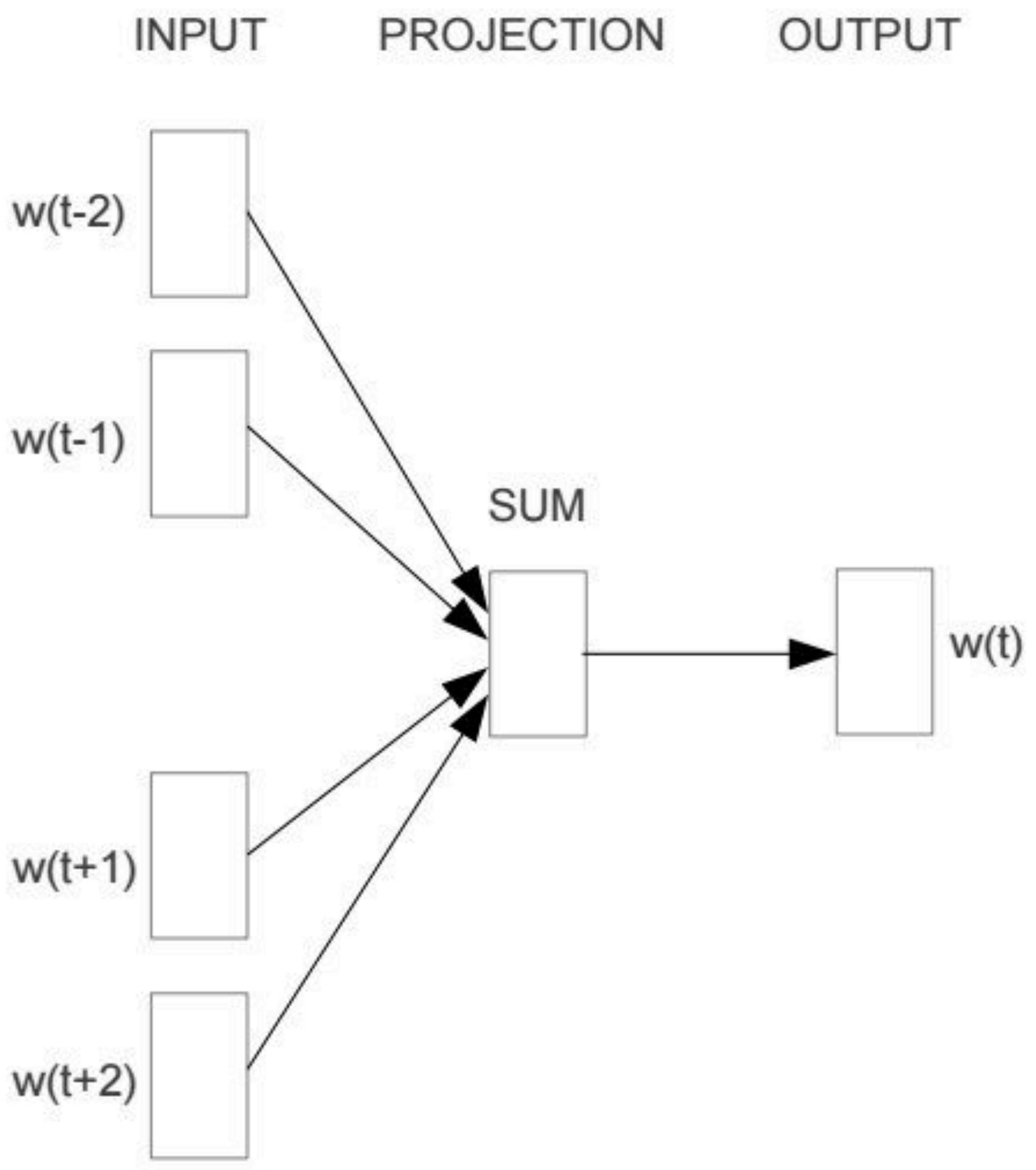
CBOW: Trained to predict the target word given its context words.



Skip-gram vs. CBOW



Skip-gram



Continuous Bag of Words (CBOW)

CBOW Objective

$$L(\theta) = \prod_{t=1}^{T} \boxed{P\left(w_t \mid \{w_{t+j}\}, -m \leq j \leq m, j \neq 0\right)}$$

$$=$$

$$P(w_t | w_{t-m}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+m})$$

Skim-gram used: $P(w_{t+j}|w_t) = \dfrac{\exp(\mathbf{v}_{w_{t+j}}^{\mathsf{T}} \mathbf{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{v}_{w'}{}^{\mathsf{T}} \mathbf{v}_{w_t})}$

where

$$\bar{\mathbf{v}}_c = \frac{1}{2m} \sum_{-m \leq j \leq m, j \neq 0} \mathbf{v}_{w_{t+j}}$$

Other word embeddings?

GloVe: **Glo**bal **Ve**ctors

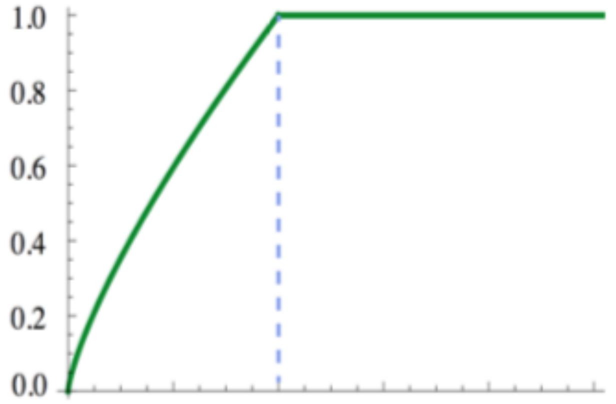
- Take the global co-occurrence statistics: $X_{i,j}$
- Key idea: let's approximate $\mathbf{u}_i \cdot \mathbf{v}_j$ using their co-occurrence counts directly

$$\mathbf{u}_i \cdot \mathbf{v}_j \approx \log X_{i,j}$$

- Formal objective:

$$J(\theta) = \sum_{i,j \in V} f(X_{i,j}) \left(\mathbf{u}_i \cdot \mathbf{v}_j + b_i + \tilde{b}_j - \log X_{i,j} \right)$$

$f \sim$



- f : Weighting function, giving more importance to more common pairs, but capped at a certain point

Advantages: Training faster, Scalable to very large corpora

(Pennington et al, 2014): GloVe: Global Vectors for Word Representation

FastText: Subword Embeddings

- Similar to Skip-gram, but break words into character-level n-grams with n = 3 to 6

3-grams: <wh, whe, her, ere, re>
4-grams: <whe, wher, here, ere>
5-grams: <wher, where, here>
6-grams: <where, where>

- Replace $\mathbf{u}_i \cdot \mathbf{v}_j$ by $\sum_{g \in n\text{-grams}(w_i)} \mathbf{u}_g \cdot \mathbf{v}_j$

- Why? Handle rare & out-of-vocabulary words better, captures morphology (plurals, tenses, derivations, misspellings)

(Bojanowski et al, 2017): Enriching Word Vectors with Subword Information

Trained word embeddings available

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Differ in algorithms, text corpora, dimensions, cased/uncased…

Applied to many other languages

Easy to use!

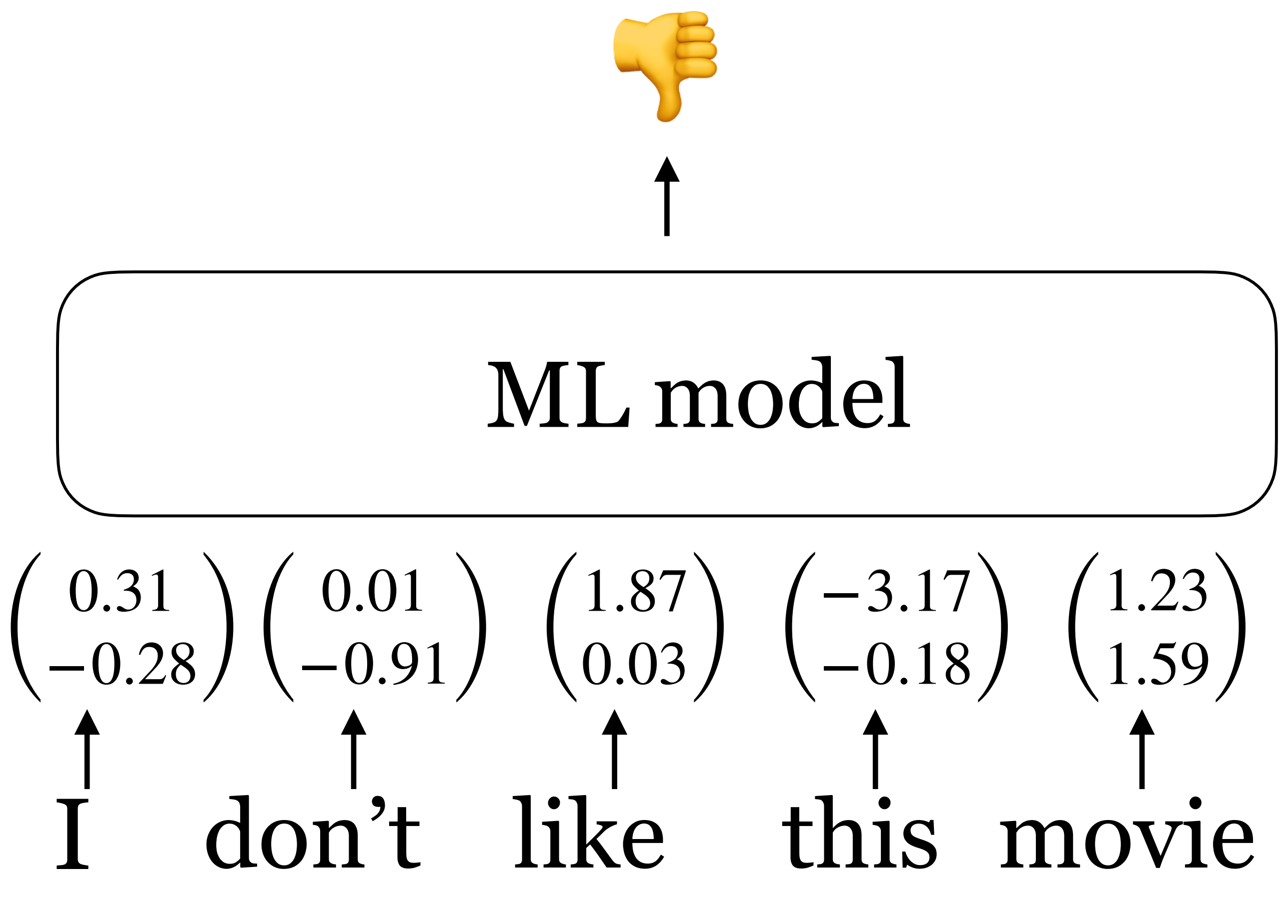
```python
from gensim.models import KeyedVectors
# Load vectors directly from the file
model = KeyedVectors.load_word2vec_format('data/GoogleGoogleNews-vectors-negative300.bin', binary=True)
# Access vectors for specific words with a keyed lookup:
vector = model['easy']
```

```
In [17]: model.similarity('straightforward', 'easy')

Out[17]: 0.5717043285477517

In [18]: model.similarity('simple', 'impossible')

Out[18]: 0.29156160264633707

In [19]: model.most_similar('simple')

Out[19]: [('straightforward', 0.7460169196128845),
          ('Simple', 0.7108174562454224),
          ('uncomplicated', 0.6297484636306763),
          ('simplest', 0.6171397566795349),
          ('easy', 0.5990299582481384),
          ('fairly_straightforward', 0.5893306732177734),
          ('deceptively_simple', 0.5743066072463989),
          ('simpler', 0.5537199378013611),
          ('simplistic', 0.5516539216041565),
          ('disarmingly_simple', 0.5365327000617981)]
```

Evaluating word embeddings

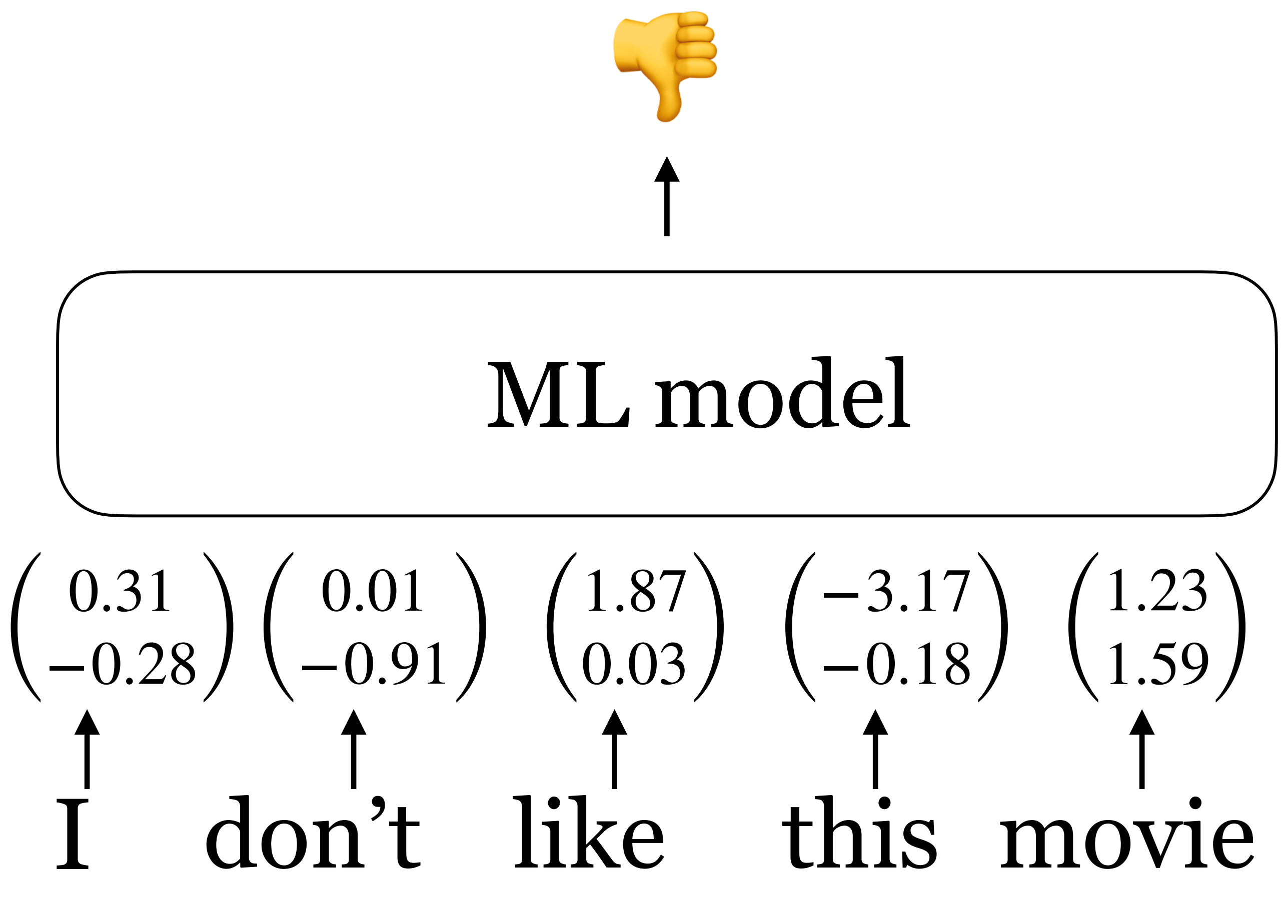
Extrinsic vs. intrinsic evaluation

- **Extrinsic evaluation**
  - Let's plug these word embeddings into a real NLP system and see whether this improves performance
  - Could take a long time but still the most important evaluation metric



Extrinsic vs. intrinsic evaluation

- **Extrinsic evaluation**
  - Let's plug these word embeddings into a real NLP system and see whether this improves performance
  - Could take a long time but still the most important evaluation metric
- **Intrinsic evaluation**
  - Evaluate on a specific/intermediate subtask
  - Fast to compute
  - Not clear if it really helps downstream tasks



Intrinsic evaluation: word similarity

Word similarity

Example dataset: wordsim-353: 353 pairs of words with human judgement

http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2} \sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

Metric: Spearman rank correlation

Intrinsic evaluation: word similarity

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	<u>72.7</u>	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	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	<u>**75.9**</u>	<u>**83.6**</u>	<u>**82.9**</u>	<u>**59.6**</u>	<u>**47.8**</u>
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

SG: Skip-gram

Intrinsic evaluation: word analogy

Word analogy test: $a : a* :: b : b*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

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
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

semantic

Chicago:Illinois≈Philadelphia: ?

syntactic

bad:worst ≈ cool: ?

More examples at

http://download.tensorflow.org/data/questions-words.txt

Metric: accuracy

Word Embeddings in Practice?

We've covered

- What word embeddings are
- How to train them
- How to evaluate them intrinsically and extrinsically

Next question

- How do we actually use word embeddings inside modern NLP models?

Questions?

Acknowledgement

Princeton COS 484 by Danqi Chen, Tri Dao, Vikram Ramaswamy
Liang Huang AI 534 (400/401), Machine Learning
Stanford CS224N NLP with Deep Learning by Diyi Yang and Tatsunori Hashimoto (+ older versions by Chris Manning)
New York University NLP by He He

FAQ

- **Enrollment related**
  - **I was already enrolled but got informed that I was removed because my enrollment was due to a system error. What happened?** Sorry about that. We were informed that you did not meet the eligibility criteria and were enrolled due to a system error, and the system automatically corrected it. We do not have control over this process. Please join the waitlist via the Google Form on the website. Qualified students will be selected using the same criteria.
  - **Does the time I join the waitlist matter?** It does not matter until the end of today. Students added to the waitlist starting tomorrow will not be considered.
  - **When will I hear back? What are my chances?** Sorry, we do not know until the 5th class. The capacity increase is made by the department, not the course staff. Most students have not received enrollment codes yet—we only sent a few to test the system and found that the codes are currently not working. We are waiting for the department to resolve this.
  - **I received an enrollment code today, but I can only waitlist.** Apologies for the confusion. The department needs to increase the course capacity before enrollment codes can be used. We will notify you once codes become active.
- **Ed/Gradescope access related**: Please contact our GSI, Zineng Tang (email address on the website).
- **Slides/recording availability**: Slides will be uploaded within 24 hours after each class. Recordings will be available starting next week—please stay tuned.