CSCE 638 Natural Language Processing Foundation and Techniques

Lecture 3: Word Representations

Kuan-Hao Huang

Spring 2026

(Some slides adapted from Chris Manning, Dan Jurafsky, Richard Socher, Karthik Narasimhan, and Danqi Chen)

Course Staff

Instructor

Kuan-Hao Huang

- Email: khhuang@tamu.edu
- Office Hour: Wed. 2pm – 3pm
- Office: PETR 219

TA

Rusali Saha

- Email: rs0921@tamu.edu
- Office Hour: Tue. 11am – 12pm
- Office: PETR 330

For questions, send emails to csce638-ta-26s@lists.tamu.edu with "[CSCE 638] Subject …"

Assignment 0

Assignment 0

RELEASE DATE: 01/20/2026

DUE DATE: 01/29/2026 11:59pm on Gradescope

LaTeX Template: `https://www.overleaf.com/read/pzhhcsmdfyst#557346`

Name: First-Name Last-Name UIN: 000000000

This assignment consists of two parts: a writing section and a programming section. For the writing section, please use the provided LaTeX template to prepare your solutions and remember to fill in your name and UIN. For the programming section, please follow the instructions carefully.

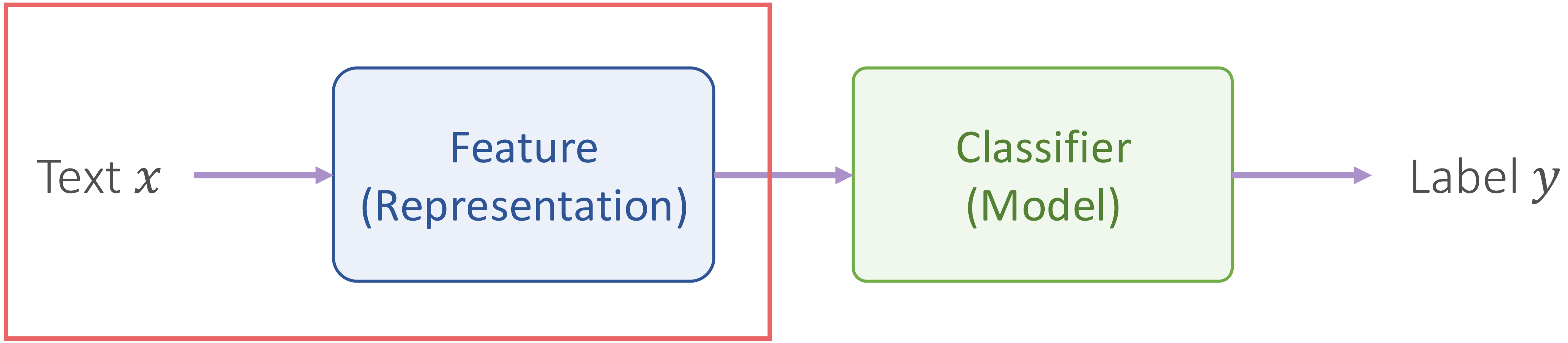
Discussions with others on course materials and assignment solutions are encouraged, and the use of AI tools as assistance is permitted. However, you must ensure that ***the final solutions are written in your own words****. It is your responsibility to avoid excessive similarity to others' work. Additionally, please clearly* ***indicate any parts where AI tools were used*** *as assistance.*

If you have any question, please send an email to `csce638-ta-26s@list.tamu.edu`

Lecture Plan

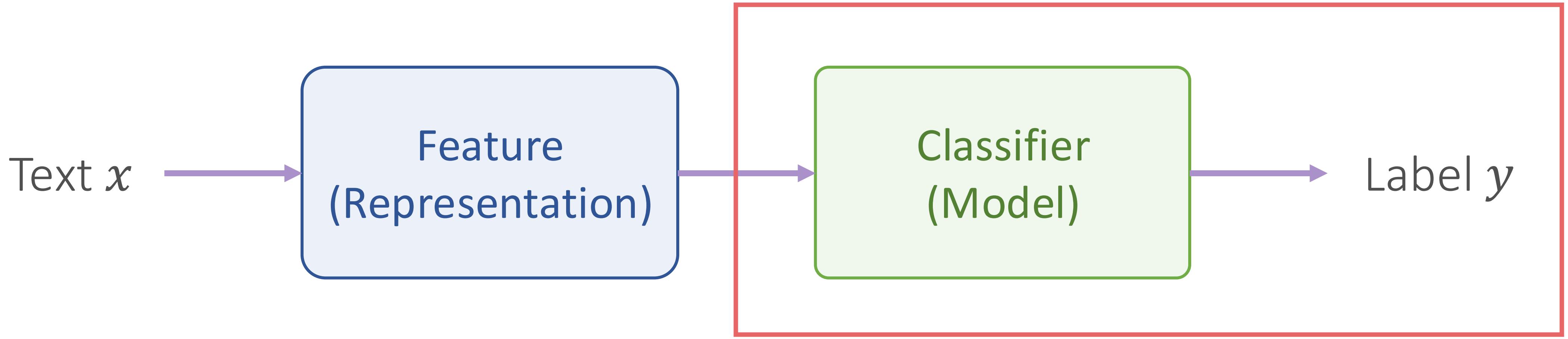
- Counting-Based Word Vectors
- Learning-Based Word Vectors
- Evaluation for Word Vectors

Recap: A General Framework for Text Classification



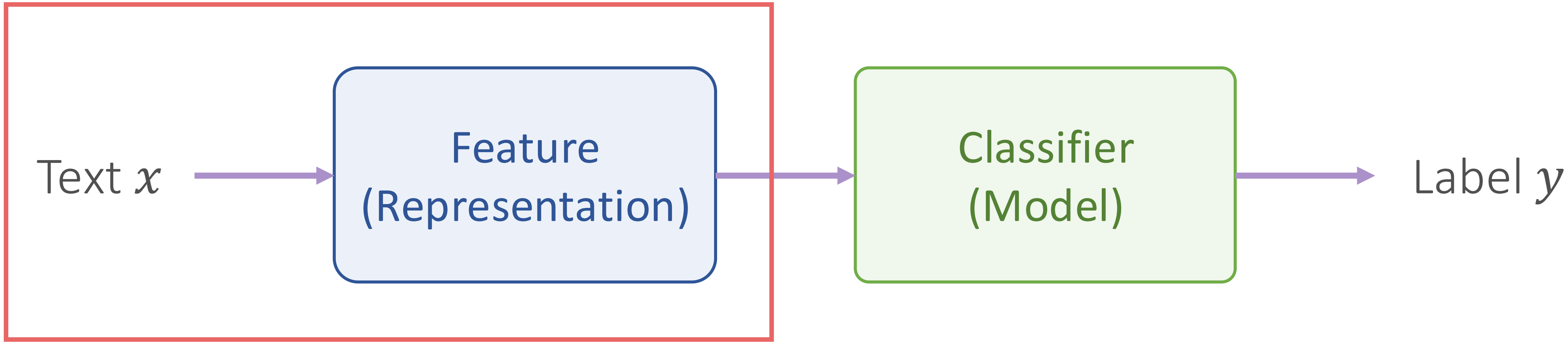
- Teach the model how to understand example x

Recap: A General Framework for Text Classification



- Teach the model how to make prediction y

Recap: Bag-of-Words and N-Grams



- Teach the model how to understand example x
- Convert the text to a mathematical form
  - The mathematical form captures essential characteristics of the text
- Bag-of-words and n-grams

We will discuss "learnable" features today!

Bag-of-Words and N-Gram Features

Bob likes Alice very much

$$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 1\ \ 1\ \ 0\ \ \ldots\ \ 0\ \ 1]$$

BoW (unigram) features

Bigram features

Alice likes Bob very much

$$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 0\ \ 0\ \ 1\ \ \ldots\ \ 1\ \ 1]$$

We can consider trigrams, 4-grams, …

Encode a text to ***one vector***

Word-Level Understanding

great 1 of 2 *adjective*

1 as in *excellent*

of the very best kind

this cake is *great*!

Synonyms & Similar Words

excellent	wonderful	terrific
awesome	fantastic	superb
lovely	beautiful	fabulous
marvelous	stellar	prime
fine	hot	neat
quality	classic	cool
famous	heavenly	good
splendid	exceptional	divine

Antonyms & Near Antonyms

terrible	poor	awful
lousy	pathetic	atrocious
bad	rotten	wretched
vile	unsatisfactory	inferior
substandard	execrable	low-grade
mediocre	second-class	middling

https://www.merriam-webster.com/thesaurus/great

Words as Vectors

Bob likes Alice very much

$$W = \begin{bmatrix} | & | & | & | & | \\ w_{bob} & w_{likes} & w_{Alice} & w_{very} & w_{much} \\ | & | & | & | & | \end{bmatrix}$$

Use ***one vector*** to represent ***each word***

Text = A list of vectors

Advantages?

How to Represent Words?

A simple solution: discrete symbols

One 1, the rest 0s

Words can be represented by one-hot vectors:

good = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]

great = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

bad = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

↑ good ↑ bad ↑ great

Vector dimension = number of words in vocabulary (e.g., 500,000+)

Any disadvantages?

Problem with Words as Discrete Symbols

Example: in web search, if a user searches for "good restaurant", we would like to match documents containing "great restaurant"

But

good = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
great = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

These two vectors are orthogonal

There is no way to encode similarity of words in these vectors!

Any solutions?

Previous Solution: Synonyms, Antonyms, and Hypernyms

Consider external resources like WordNet, a thesaurus containing lists of Synonyms, antonyms, and hypernyms

```
from nltk.corpus import wordnet as wn
poses = { 'n' : 'noun', 'v' : 'verb', 's' : 'adj (s)', 'a' : 'adj', 'r' : 'adv'}
for synset in wn.synsets("bad"):
   print("{}: {}".format(poses[synset.pos()],
              ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: bad, badness
adj: bad
adj (s): bad, big
adj (s): bad, tough
adj (s): bad, spoiled, spoilt
adj: regretful, sorry, bad
adj (s): bad, uncollectible
…
adj (s): bad, risky, high-risk, speculative
adj (s): bad, unfit, unsound
adj (s): bad, forged
adj (s): bad, defective
adv: badly, bad
```

Previous Solution: Synonyms, Antonyms, and Hypernyms

Consider external resources like WordNet, a thesaurus containing lists of Synonyms, antonyms, and hypernyms

welfare sorry

good = [0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0]

great = [0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0]

bad = [0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0]

good bad great

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

Similarity(good, great) > Similarity(good, bad)

Any disadvantages?

Problems with Resources Like WordNet

- Subjective
- A useful resource but missing nuance
  - e.g., “sorry” is listed as a synonym for “bad”
  - This is only correct in some contexts
- Requires human labor to create and adapt

Representing Words by Their Contexts

Distributional hypothesis: A word's meaning is given by the words that frequently appear close-by

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

Slide adapted from Chris Manning

Distributional Hypothesis: Example

C1: A bottle of ___ is on the table.

C2: Everybody likes ___.

C3: Don't have ___ before you drive.

C4: I bought ___ yesterday.

	C1	C2	C3	C4
wine	1	1	1	1
juice	1	1	0	1
loud	0	0	0	0
apples	0	1	0	1
choices	0	1	0	0
motor-oil	1	0	0	1

A word's meaning is given by the words that frequently appear close-by

Slide adapted from Danqi Chen

Word Vectors from Word-Word Co-Occurrence Matrix

- Main idea: Similar contexts → Similar word co-occurrence
- Collect a bunch of texts and compute co-occurrence matrix
- Words can be represented by row vectors

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

Word Vector

High cosine similarity!

	shark	computer	data	eat	result	sugar
apple	0	0	0	8	0	2
bread	0	0	0	9	0	1
digital	0	6	5	0	2	0
information	0	4	10	0	2	0

Low cosine similarity!

Most entries are 0s → sparse vectors

Issues with Word-Word Co-Occurrence Matrix

- Using raw frequency counts is not always very good (why?)
  - Some frequent words (e.g., the, it, or they) can have large counts

	shark	computer	data	eat	result	sugar	the	it
apple	0	0	0	8	0	2	104	67
bread	0	0	0	9	0	1	95	76
digital	0	6	5	0	2	0	101	65

Similarity(apple, bread) ≈ 0.994710

Similarity(apple, digital) ≈ 0.995545

Similarity is dominated by frequent words

Solution: use a ***weighted function*** instead of raw counts

Pointwise Mutual Information

Pointwise Mutual Information (PMI)

Do events x and y co-occur more or less than if they were independent?

$$\text{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI = 0 → x and y occur independently → co-occurrence is as expected
- PMI > 0 → x and y co-occur more often than expected
- PMI < 0 → x and y co-occur less often than expected

Co-Occurrence Matrix with Positive PMI

Positive Pointwise Mutual Information (PPMI)

$$\mathrm{PPMI}(x,y) = \max\left(\log_2 \frac{P(x,y)}{P(x)P(y)}, 0\right)$$

	shark	computer	data	eat	result	sugar	the	it
apple	0	0	0	1.80	0	0.35	0.08	0
bread	0	0	0	1.54	0	0.29	0	0.14
digital	0	1.47	1.22	0	0.61	0	0.10	0.06

Similarity(apple, bread) ≈ 0.995069

Similarity(apple, digital) ≈ 0.010795

Sparse Vectors vs. Dense Vectors

- The vectors in the word-word occurrence matrix are
  - **Long**: vocabulary size
  - **Sparse**: most are 0's
- Can we have short short (50-300 dimensional) and dense (real-valued) vectors?
  - Short vectors are easier to use as features in ML systems
  - Dense vectors may generalize better than explicit counts
  - Sparse vectors 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!

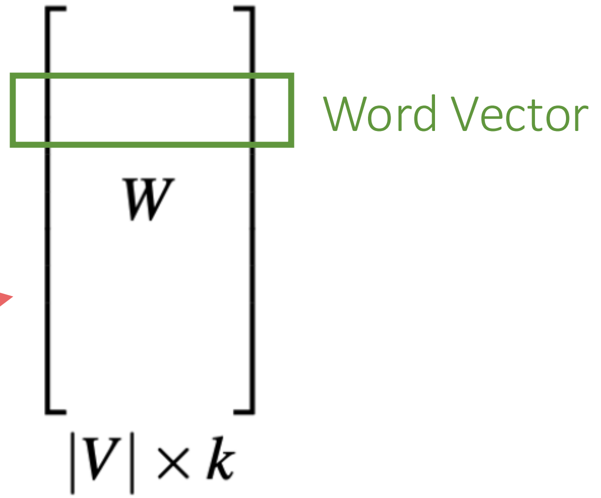
How to Get Dense Vectors?

- Singular value decomposition (SVD) of PPMI weighted co-occurrence matrix

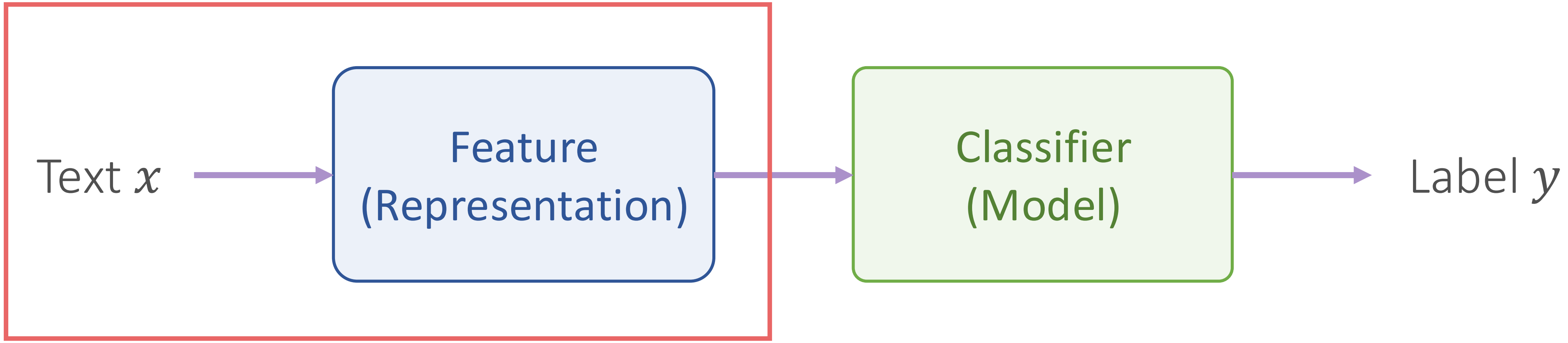
$$\underset{|V|\times|V|}{\begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} & & \\ & W & \\ & & \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots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

Only keep the top k singular values

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

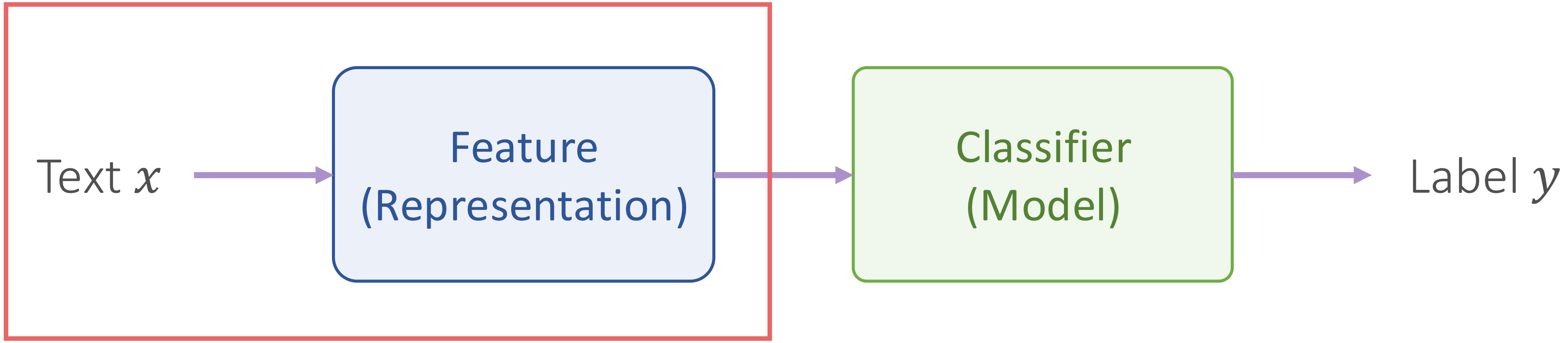


Counting-Based Word Vectors



- Use one vector to represent each word
- Get word vectors by singular value decomposition (SVD) of PPMI weighted co-occurrence matrix

Learning-Based Word Vectors



- Can we learn word vectors directly from text?

Word2Vec

- Efficient Estimation of Word Representations in Vector Space, 2013
  - 50000+ citations

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Learning Word Vectors

Map each word to a vector!

$$\text{similarity}(word1, word2) = \frac{\mathbf{v}_{word1} \cdot \mathbf{v}_{word2}}{\|\mathbf{v}_{word1}\| \times \|\mathbf{v}_{word2}\|}$$

$$\mathbf{v}_{great} = [0.12, 0.38, -0.91, 0.57, -0.64]$$

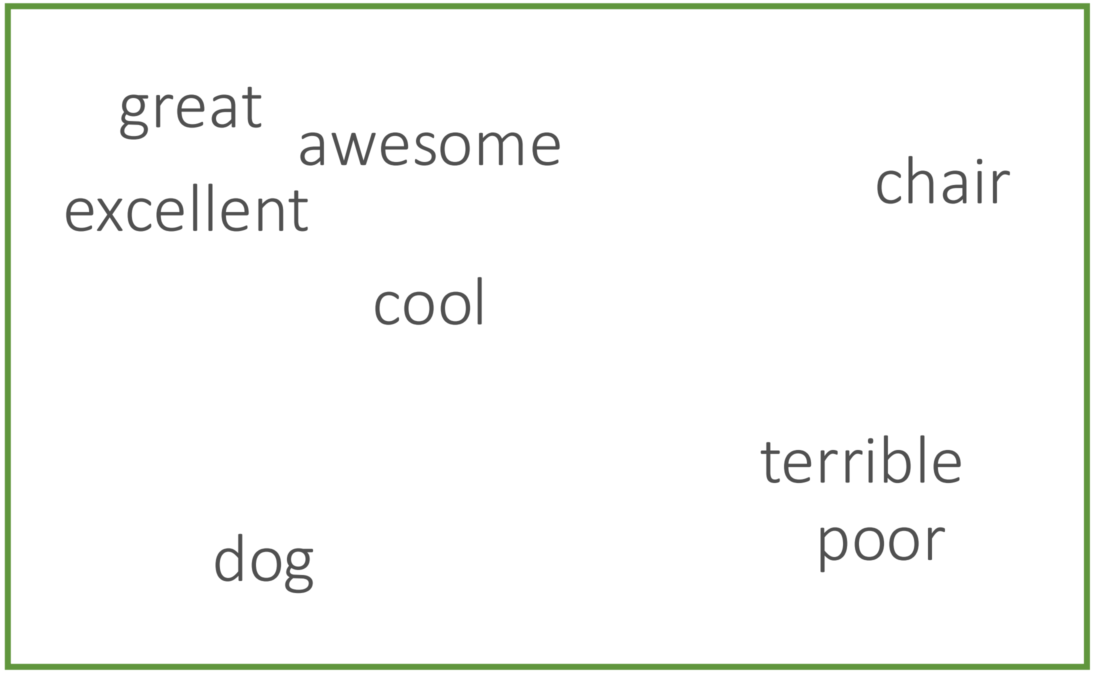
$$\mathbf{v}_{excellent} = [0.16, 0.47, -0.87, 0.50, -0.55]$$

$$\mathbf{v}_{awesome} = [0.08, 0.28, -0.90, 0.61, -0.54]$$

$$\mathbf{v}_{terrible} = [0.92, -0.36, 0.11, -0.24, 0.14]$$

$$\mathbf{v}_{poor} = [0.85, -0.40, 0.02, -0.31, 0.23]$$

Semantic meaning of words



Semantic relationship between words

How to learn those word vectors/embeddings/representations?

Word2Vec: Learning Problem

…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…*

Based on distributional hypothesis

Center word $\leftrightarrow$ Context words

Given the context words, we can predict the most likely center word

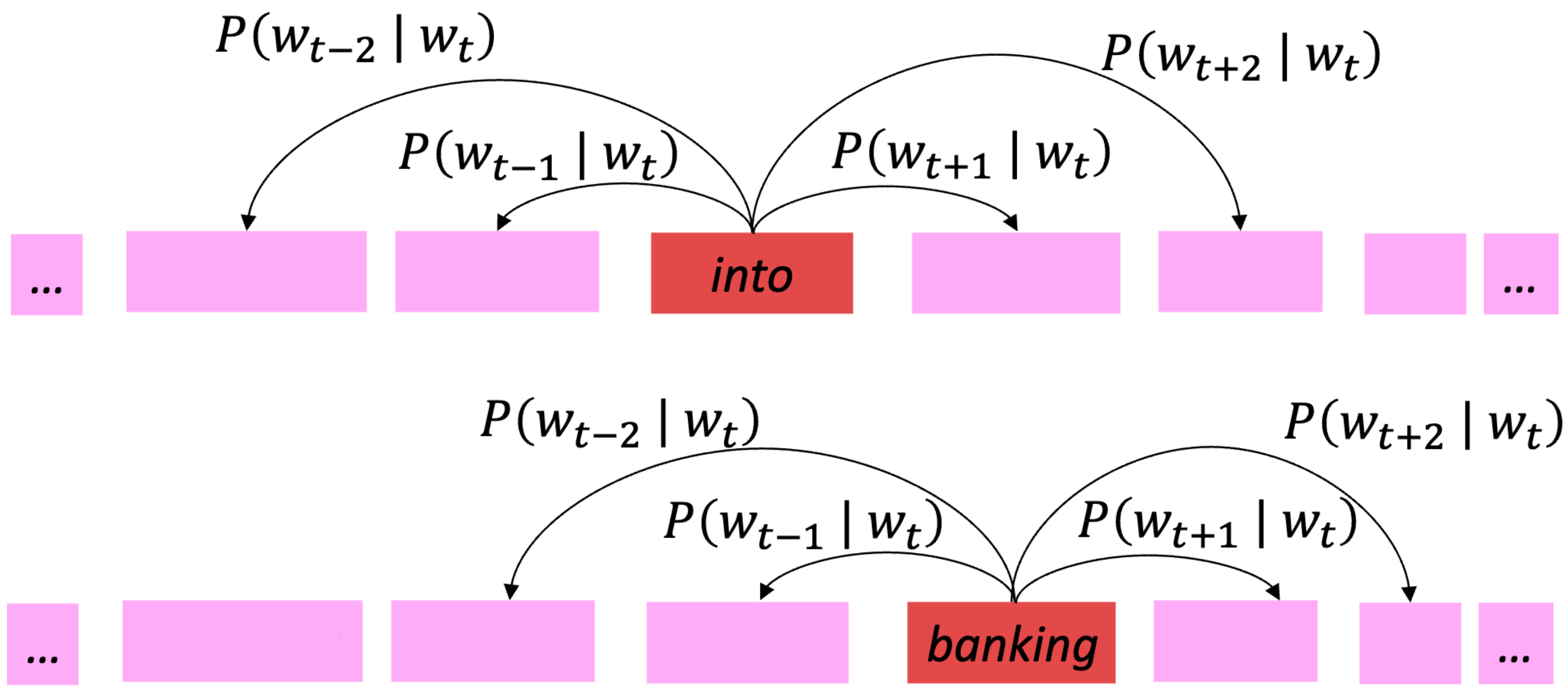
Given the center word, we can predict the most likely context words

Efficient Estimation of Word Representations in Vector Space, 2013

Word2Vec: Overview

- **Main idea:** we want to use words to predict their context words
- Context: a fixed window of size m

Use center word w_t to predict context words w_{t-m} to w_{t+m}



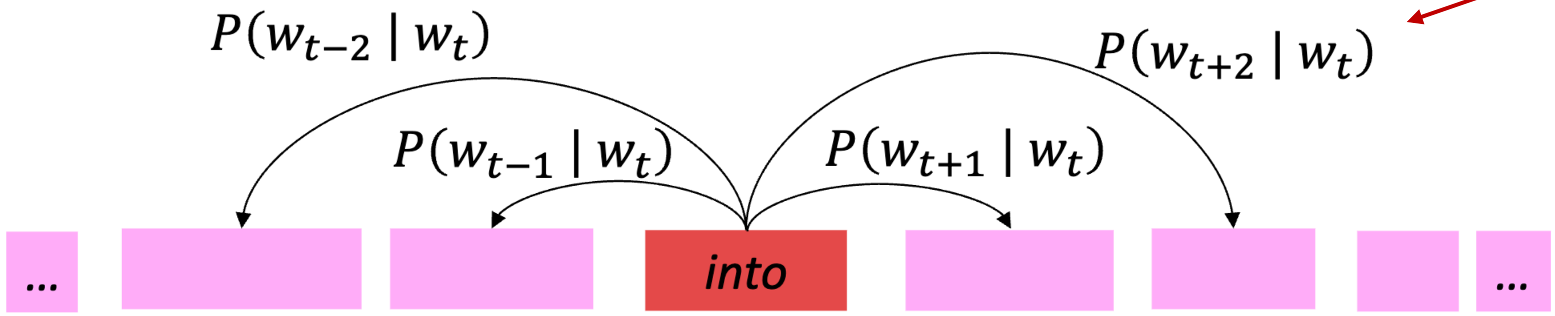
Word2Vec: Overview

- **Main idea:** we want to use words to predict their context words
- Context: a fixed window of size m

Classification Problem

Use center word w_t to predict context words w_{t-m} to w_{t+m}

$P(b|a)$ = given the center word is a, what is the probability that b is a context word?

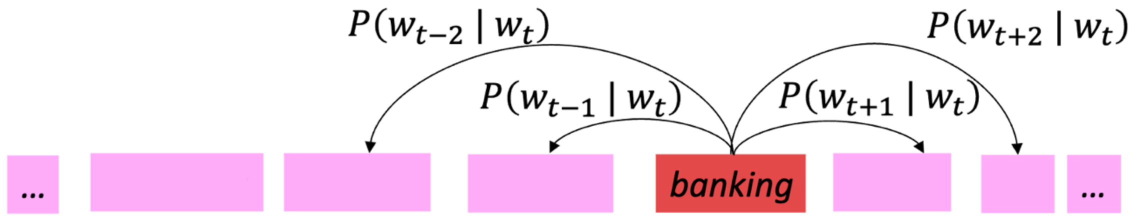


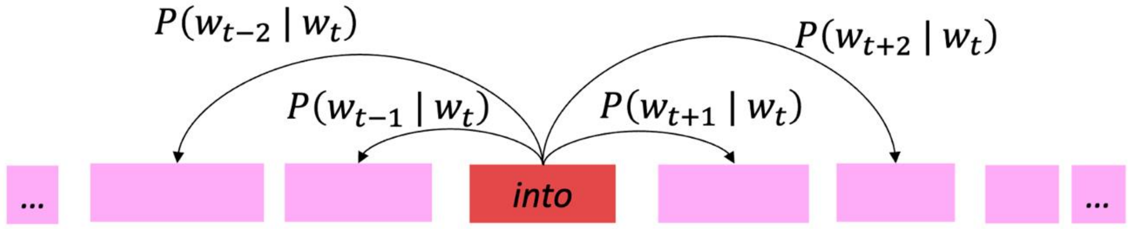
$P(\cdot\,|a)$ is a probability distribution defined over $\mathcal{V}$:

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

We will define the distribution soon!

Word2Vec: Overview





$P(\text{money} \mid \text{banking}) = 0.21$

$P(\text{crises} \mid \text{banking}) = 0.18$

$P(\text{deposit} \mid \text{banking}) = 0.15$

$P(\text{dog} \mid \text{banking}) = 0.01$

$P(\text{turning} \mid \text{banking}) = 0.06$

…

$P(\text{sunny} \mid \text{banking}) = 0.02$

$P(\text{money} \mid \text{banking}) = 0.06$

$P(\text{crises} \mid \text{banking}) = 0.19$

$P(\text{deposit} \mid \text{banking}) = 0.03$

$P(\text{dog} \mid \text{banking}) = 0.01$

$P(\text{turning} \mid \text{banking}) = 0.30$

…

$P(\text{sunny} \mid \text{banking}) = 0.01$

Word2Vec: Defining Probabilities (Simplified Version)

How to calculate $P(w_{context} \mid w_{center})$?

$$\frac{\mathbf{v}_{word1} \cdot \mathbf{v}_{word2}}{\|\mathbf{v}_{word1}\| \times \|\mathbf{v}_{word2}\|}$$

We consider Inner product $\mathbf{v}_{w_{center}} \cdot \mathbf{v}_{w_{context}}$ as the score

If $\mathbf{v}_{w_{center}} \cdot \mathbf{v}_{w_{context}}$ is higher, $P(w_{context} \mid w_{center})$ is higher

$$P(w_{context} \mid w_{center}) = \frac{\exp(\mathbf{v}_{w_{center}} \cdot \mathbf{v}_{w_{context}})}{\sum_{k \in V} \exp(\mathbf{v}_{w_{center}} \cdot \mathbf{v}_k)}$$

Scores can be asymmetric! It is less likely that a word appears in its own context

Normalize scores to probabilities

Word2Vec: Defining Probabilities (Final Version)

How to calculate $P(w_{context} | w_{center})$?

$$\frac{\mathbf{v}_{word1} \cdot \mathbf{v}_{word2}}{\|\mathbf{v}_{word1}\| \times \|\mathbf{v}_{word2}\|}$$

We consider Inner product $\mathbf{u}_{w_{center}} \cdot \mathbf{v}_{w_{context}}$ as the score

If $\mathbf{u}_{w_{center}} \cdot \mathbf{v}_{w_{context}}$ is higher, $P(w_{context} | w_{center})$ is higher

$$P(w_{context} | w_{center}) = \frac{\exp(\mathbf{u}_{w_{center}} \cdot \mathbf{v}_{w_{context}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_{center}} \cdot \mathbf{v}_k)}$$

Normalize scores to probabilities

Word2Vec: Defining Probabilities (Final Version)

We have two sets of vectors for each word in the vocabulary

$\mathbf{u}_w \in \mathbb{R}^d$: word vector when w is a center word

$\mathbf{v}_w \in \mathbb{R}^d$: word vector when w is a context word

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

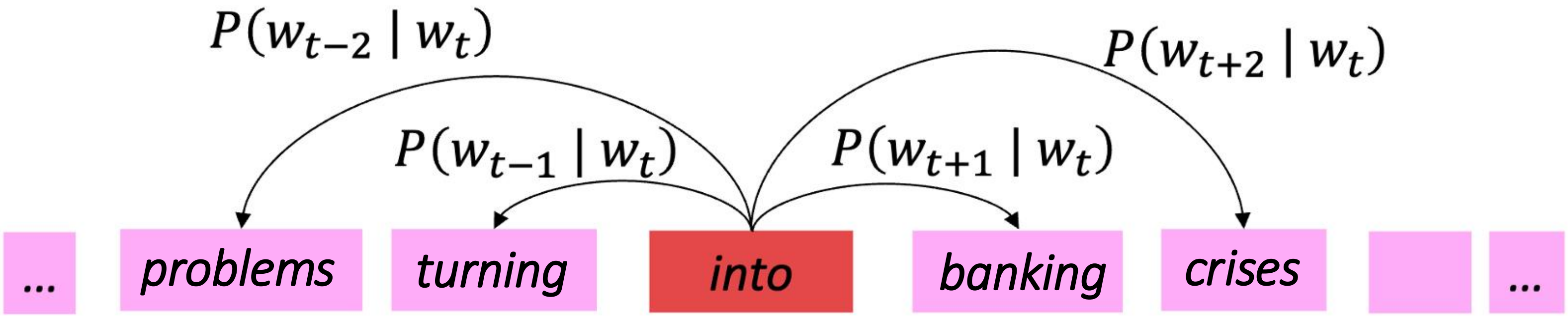
Normalize over entire vocabulary to give probability distribution

The score to indicate how likely the context word w_{t+j} appears with the center word w_t

Softmax function: mapping arbitrary values to a probability distribution

$$\text{softmax}(t) = \frac{e^t}{\sum_c e^c}$$

Word2Vec: Training Intuition



$P(\text{problems} \mid \text{into})$ ↑

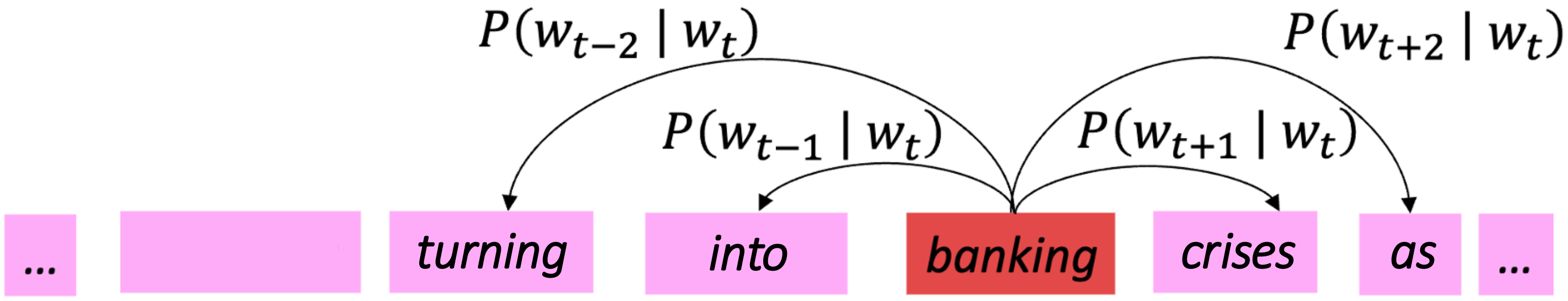
$P(\text{turning} \mid \text{into})$ ↑

$P(\text{banking} \mid \text{into})$ ↑

$P(\text{crises} \mid \text{into})$ ↑

$P(\text{other words} \mid \text{into})$ ↓

Word2Vec: Training Intuition



$P(\text{turning} \mid \text{banking}) \uparrow$

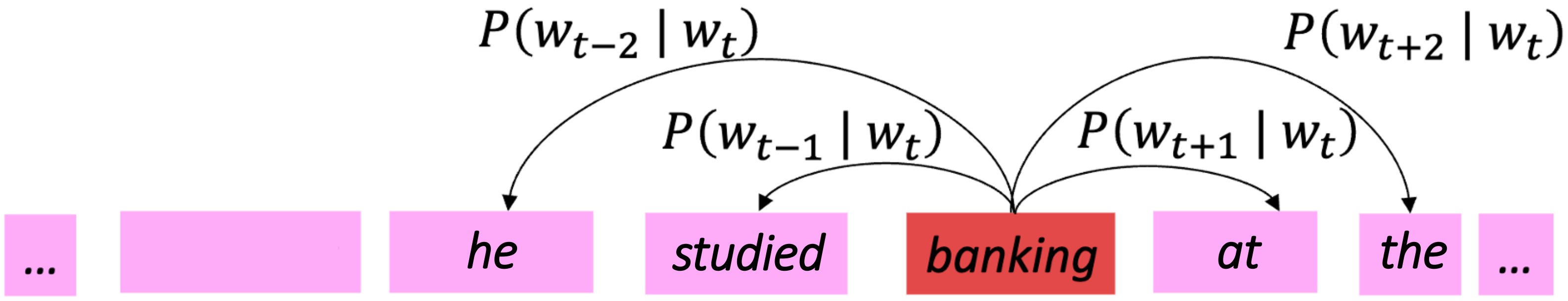
$P(\text{into} \mid \text{banking}) \uparrow$

$P(\text{crises} \mid \text{banking}) \uparrow$

$P(\text{as} \mid \text{banking}) \uparrow$

$P(\text{other words} \mid \text{banking}) \downarrow$

Word2Vec: Training Intuition



$P(\text{he} \mid \text{banking}) \uparrow$

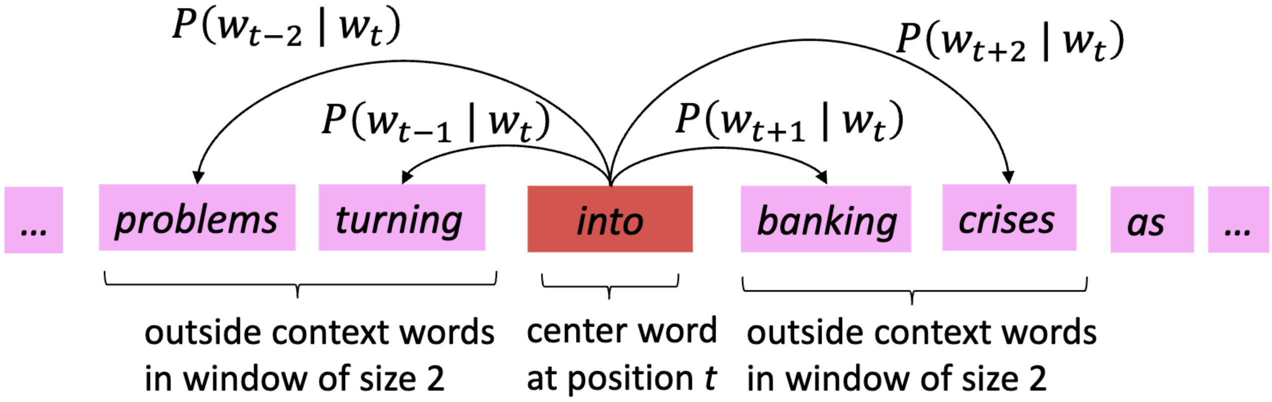
$P(\text{studied} \mid \text{banking}) \uparrow$

$P(\text{at} \mid \text{banking}) \uparrow$

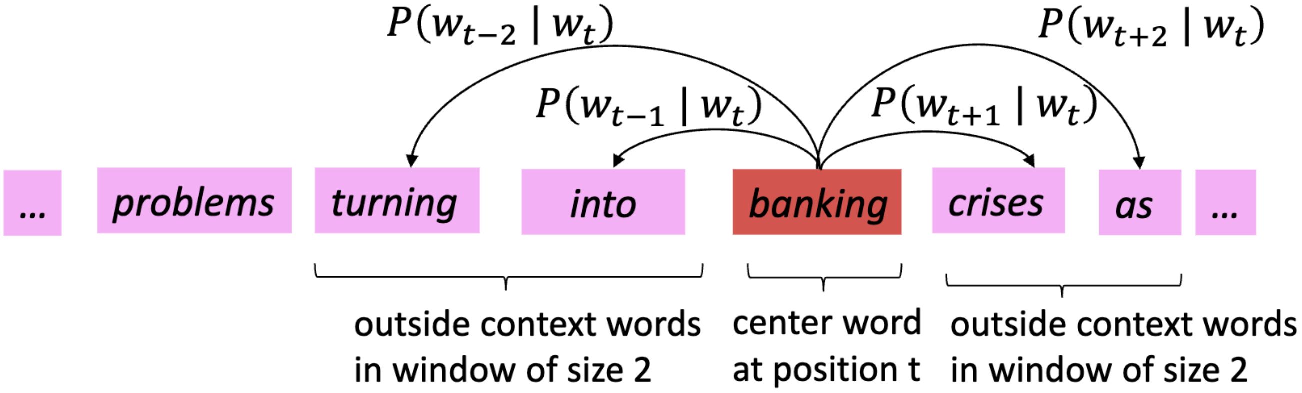
$P(\text{the} \mid \text{banking}) \uparrow$

$P(\text{other words} \mid \text{banking}) \downarrow$

Word2Vec: Training Data



Collect into training data
(into, problems)
(into, turning)
(into, banking)
(into, crises)



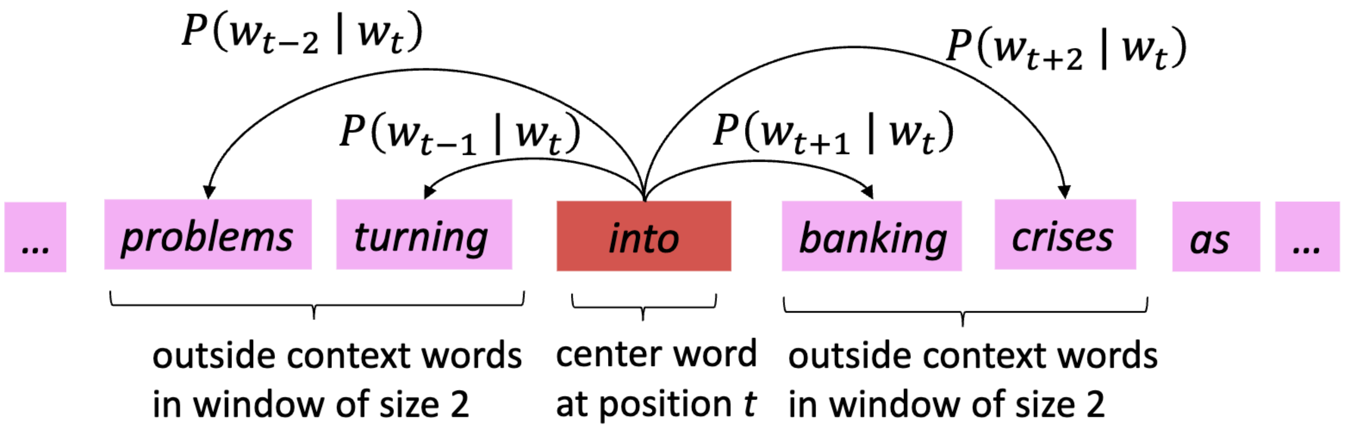
Collect into training data
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)

Maximize the likelihood

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

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

Word2Vec: Likelihood



For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t

θ all parameters to be optimized

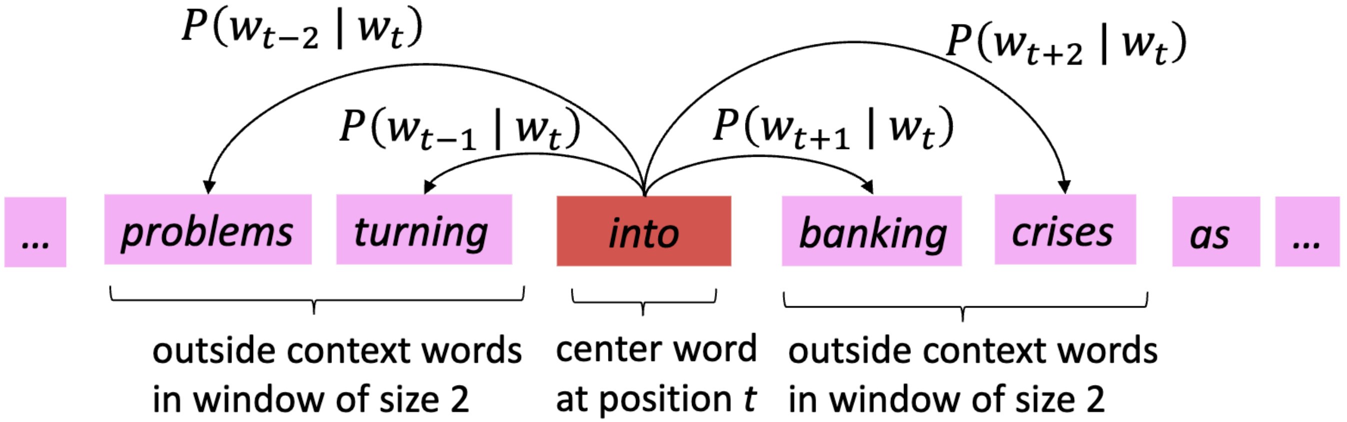
$$\text{Likelihood} = \mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t ; \theta)$$

Probability over all vocabulary V

For each position $t = 1, \ldots, T$

Likelihood for all context words given center word w_t

Word2Vec: Objective Function



The objective function $J(\theta)$ is the (average) negative log likelihood

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

We minimize the objective function (also called cost or loss function)

Word2Vec: How to Train Word Vectors?

Parameters: $\theta = \{\{\mathbf{u}_k\}, \{\boldsymbol{v}_k\}\}$

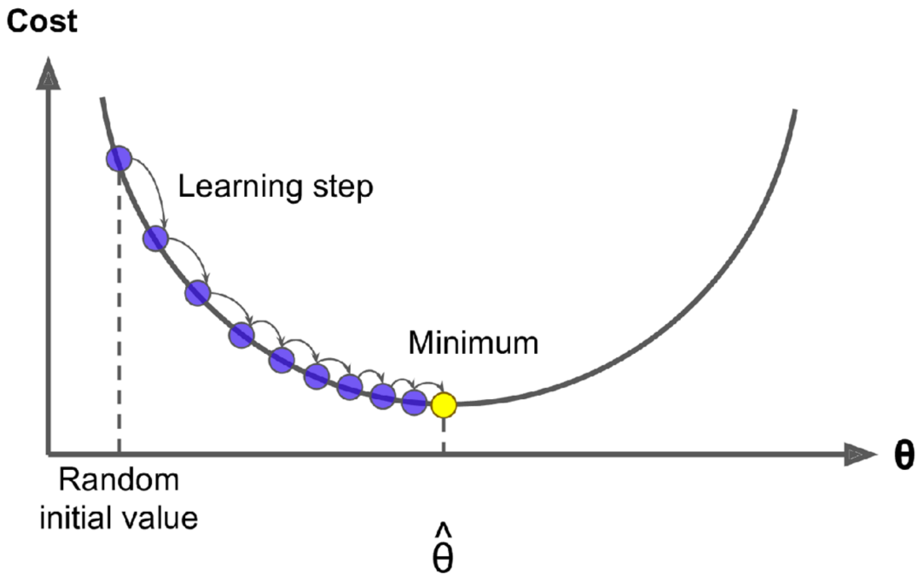
Objective function: $J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-m\leq j\leq m, j\neq 0} \log P\left(w_{t+j} \middle| w_t ; \theta\right)$

Our goal: find parameters θ that minimize the objective function $J(\theta)$

Solution: stochastic gradient descent (SGD)

- Randomly initialize parameters θ
- For each iteration $\theta \longleftarrow \theta - \eta \nabla_\theta J(\theta)$

Learning step → η Gradient → ∇_θ



https://insightfultscript.com/collections/programming/machine-learning/sgd/

Word2Vec: Computing the Gradients

Objective function

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log P\left(w_{t+j} \middle| w_t ; \theta\right)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \boxed{-\log P\left(w_{t+j} \middle| w_t ; \theta\right)}$$

The gradients can be calculated separately!

For simplicity, we consider one pair of center/context words (o, c)

$$y = -\log P(c|o ; \theta) = -\log\left(\frac{\exp(\mathbf{u}_o \cdot \mathbf{v}_c)}{\sum_{k\in V} \exp(\mathbf{u}_o \cdot \mathbf{v}_k)}\right)$$

$$\boxed{\frac{\partial y}{\partial \mathbf{u}_o} \quad \frac{\partial y}{\partial \boldsymbol{v}_c}}$$

We need to compute this!

Word2Vec: Computing the Gradients

$$y = -\log P(c|o) = -\log\left(\frac{\exp(\mathbf{u}_o \cdot \mathbf{v}_c)}{\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)}\right) = \boxed{-\log(\exp(\mathbf{u}_o \cdot \mathbf{v}_c))} + \log\left(\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)\right)$$

$$= -\mathbf{u}_o \cdot \mathbf{v}_c$$

$$\frac{\partial \log(x)}{\partial x} = \frac{1}{x} \qquad \frac{\partial \exp(x)}{\partial x} = \exp(x)$$

$$\frac{\partial y}{\partial \mathbf{u}_o} = \frac{\partial(-\mathbf{u}_o \cdot \mathbf{v}_c + \log(\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)))}{\partial \mathbf{u}_o} = -\mathbf{v}_c + \frac{\sum_{k\in V}\frac{\partial \exp(\mathbf{u}_o \cdot \mathbf{v}_k)}{\partial \mathbf{u}_o}}{\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)}$$

$$= -\mathbf{v}_c + \frac{\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)\,\mathbf{v}_k}{\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)} = -\mathbf{v}_c + \sum_{k\in V}\frac{\exp(\mathbf{u}_o \cdot \mathbf{v}_k)\,\mathbf{v}_k}{\sum_{k\in V}\exp(\mathbf{u}_o \cdot \mathbf{v}_k)}$$

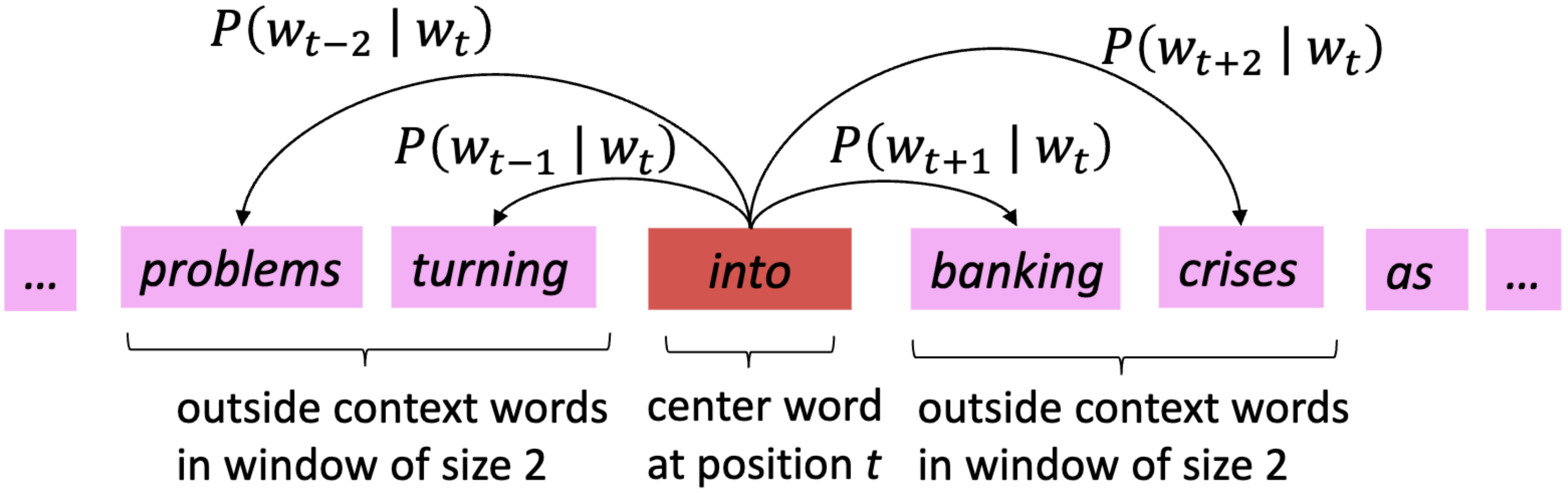
$$= -\mathbf{v}_c + \sum_{k\in V} P(k|o)\,\mathbf{v}_k$$

$$\boxed{\frac{\partial y}{\partial \mathbf{v}_k} = -1(k = c)\mathbf{u}_o + P(k|o)\mathbf{u}_o}$$

Similar calculation step

Word2Vec: Training Process

- Randomly initialize parameters $\mathbf{u}_i, \mathbf{v}_i$
- Walk through the training corpus and collect training data (o, c)



$$\mathbf{u}_o \leftarrow \mathbf{u}_o - \eta \frac{\partial y}{\partial \mathbf{u}_o} \qquad \mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v}_k} \quad \forall k \in V$$

Word2Vec: Negative Sampling

Issue: every time we get one pair of (o, c), we have to update $\mathbf{v}_k$ with all the words in the vocabulary.

$$\mathbf{u}_o \leftarrow \mathbf{u}_o - \eta \frac{\partial y}{\partial \mathbf{u}_o} \qquad \mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v}_k} \quad \forall k \in V$$

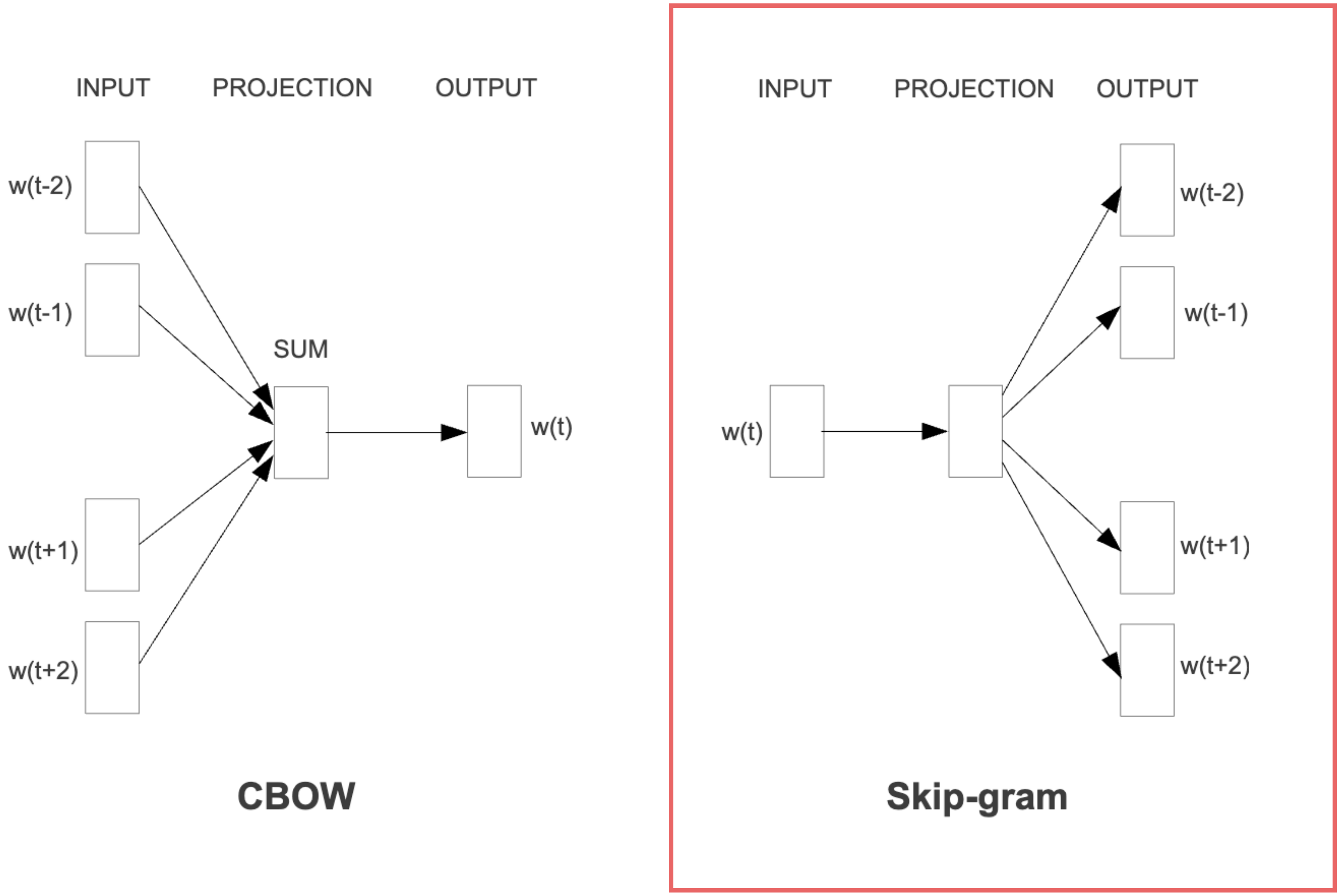
Negative sampling: instead of considering all the words in V, we randomly sample K (5-20) negative examples

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

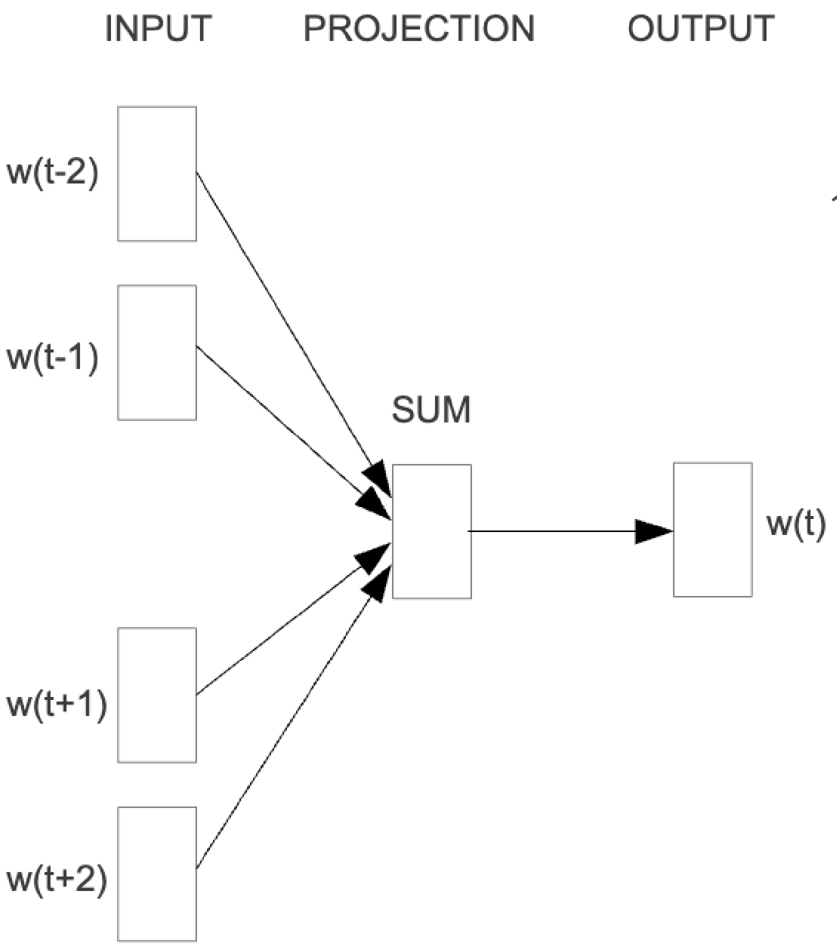
Negative sampling $$y = -\log(\sigma(\mathbf{u}_o \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_o \cdot \mathbf{v}_j))$$

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

Continuous Bag of Words (CBOW) vs Skip-Grams



Continuous Bag of Words (CBOW)



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

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

$$\bar{\mathbf{v}}_t = \frac{1}{2m} \sum_{-m \le j \le m, j \ne 0} \mathbf{v}_{t+j}$$

GloVe: Global Vectors

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

Idea: capture ratios of co-occurrence probabilities as linear meaning components in a word vector space

Log-bilinear model $w_i \cdot w_j = \log P(i|j)$

Vector difference $w_i \cdot (w_a - w_b) = \dfrac{\log P(x|a)}{\log P(x|b)}$

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_i^{\top}\tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Global co-occurrence statistics

Training faster and scalable to very large corpora!

FastText: Sub-Word Embeddings

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

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

where

3-grams: <wh, whe, her, ere, re>

4-grams: <whe, wher, here, ere>

5-grams: <wher, where, here>

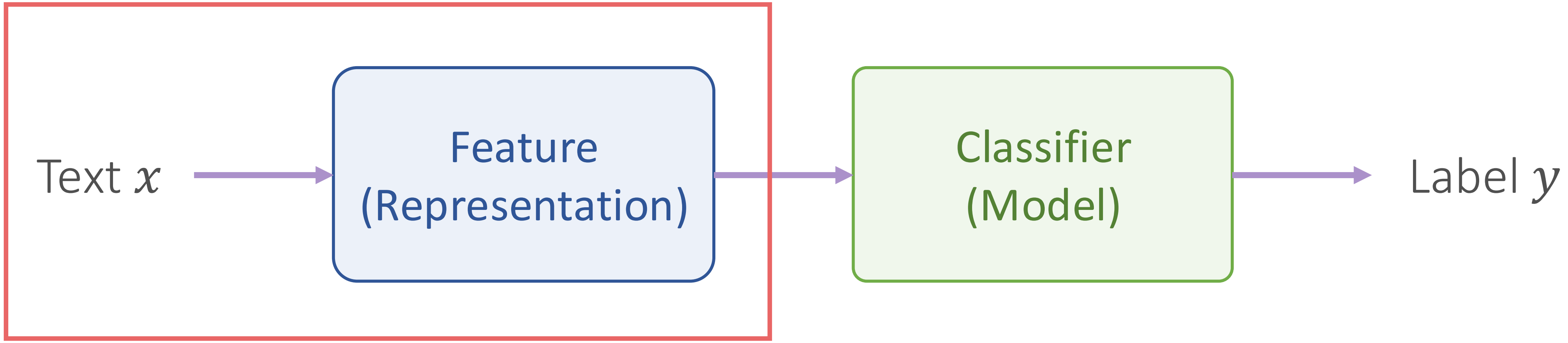
6-grams: <where, where>

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

Trained Word Vectors Are Available

- Word2Vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Learning-Based Word Vectors



- Learn word vectors directly from text
  - Word2Vec (Skip-Gram and CBOW)
  - GloVe
  - FastText

How to Evaluate the Quality of Word Embeddings?

- Intrinsic evaluation
  - Measures the quality of word embeddings by assessing their performance on specific linguistic or semantic tasks
- Extrinsic evaluation
  - Measures the quality of word embeddings by testing their impact on downstream and real-world 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

Cosine similarity:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Metric: Spearman rank correlation

Pearson's Correlation Coefficient

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

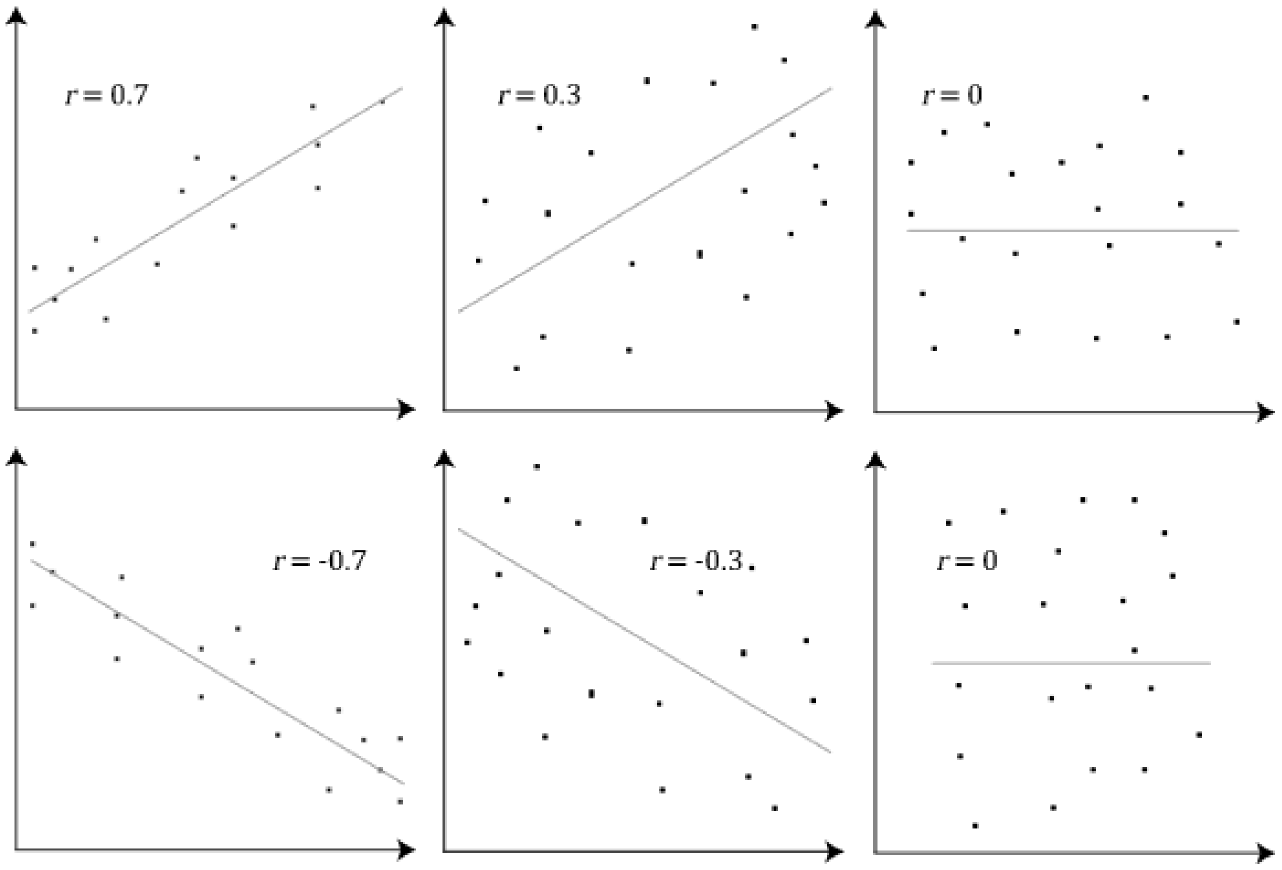
r = correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

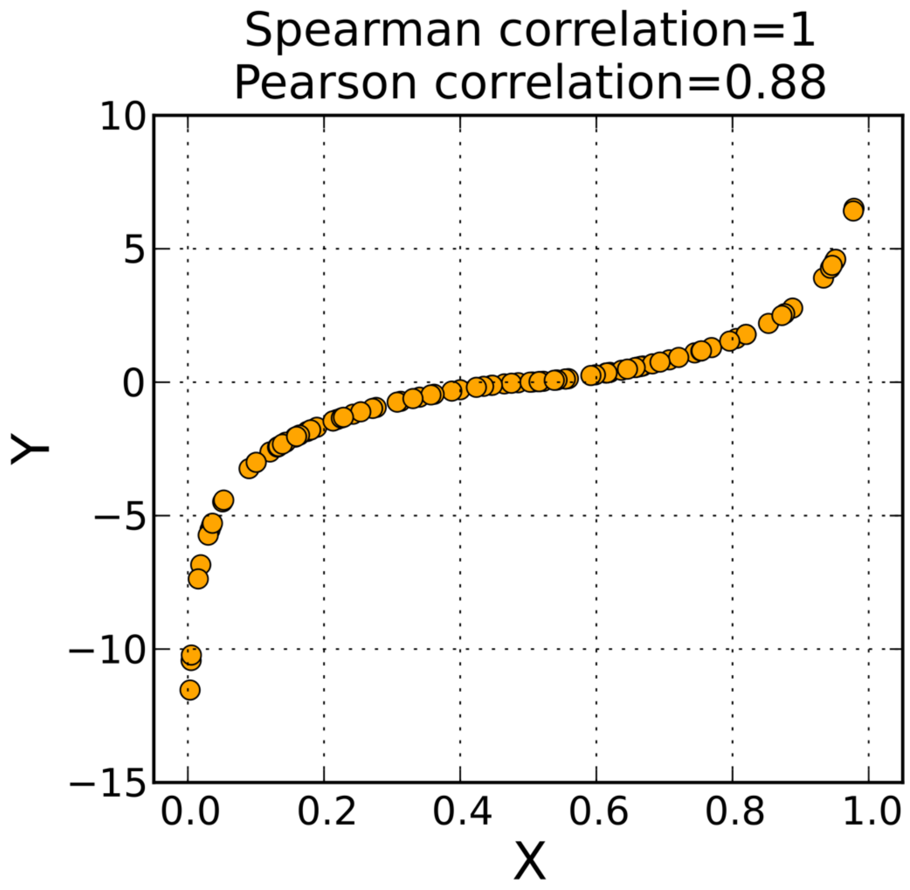
$\bar{y}$ = mean of the values of the y-variable



https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php

Spearman's Correlation Coefficient

- Pearson's correlation coefficient on rank
- Score
  - Human: [1.2, 3.4, 2.5, 0.7, 4.0]
  - Machine: [0.5, 3.3, 1.0, 1.2, 3.4]
- Rank
  - Human: [4, 2, 3, 5, 1]
  - Machine: [5, 2, 4, 3, 1]
- Assesses monotonic relationships
  - whether linear or not



https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

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

GloVe: Global Vectors for Word Representation, EMNLP 2014

Intrinsic Evaluation: Word Analogy

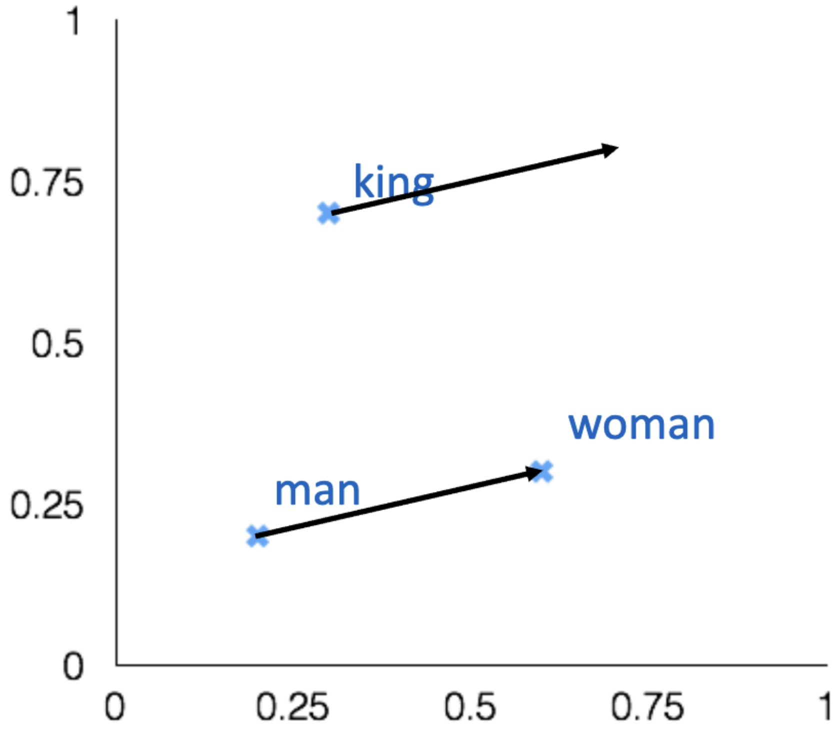
Word analogy

man: woman ≈ king: ?

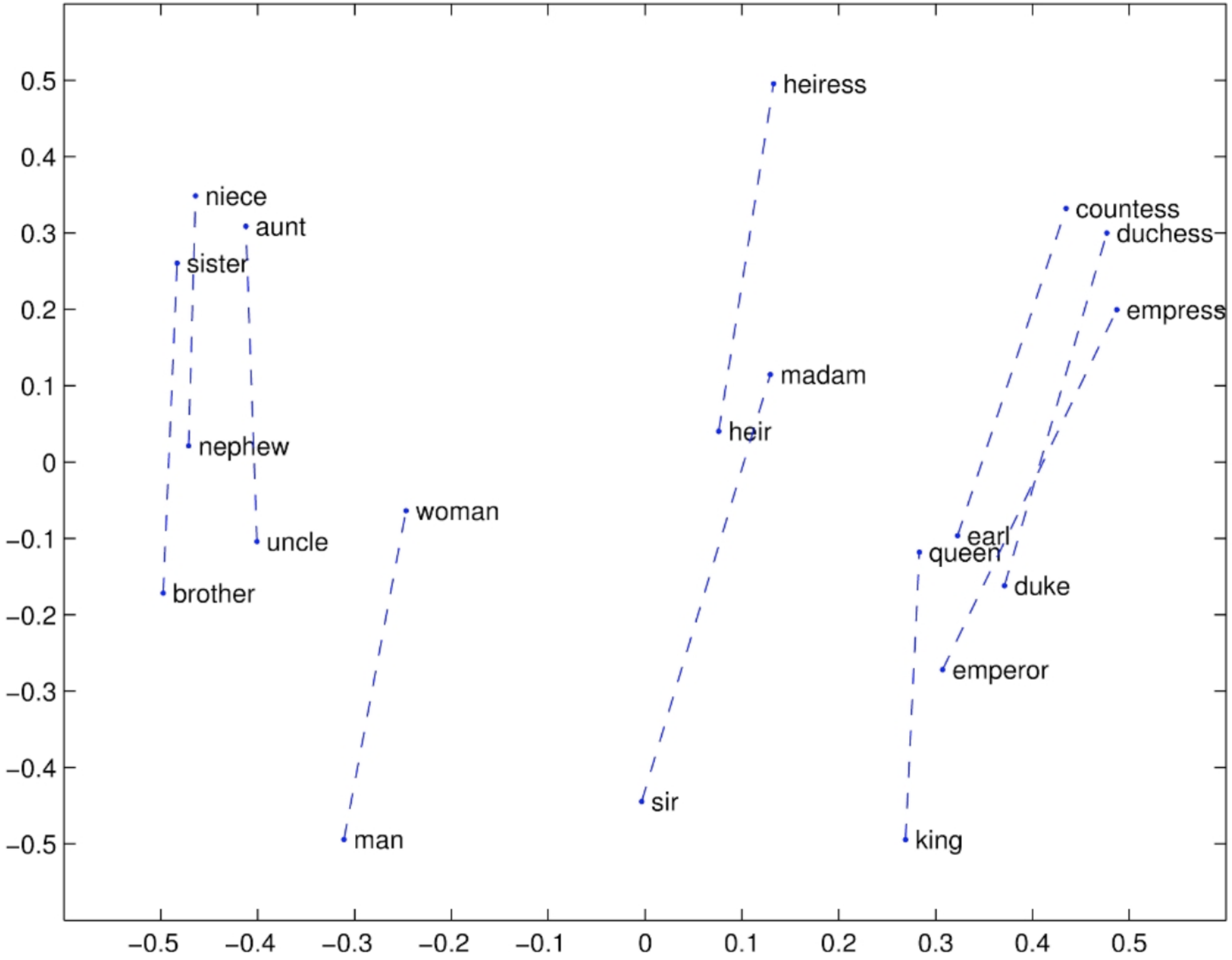
Paris: France ≈ London: ?

bad: worst ≈ cool: ?

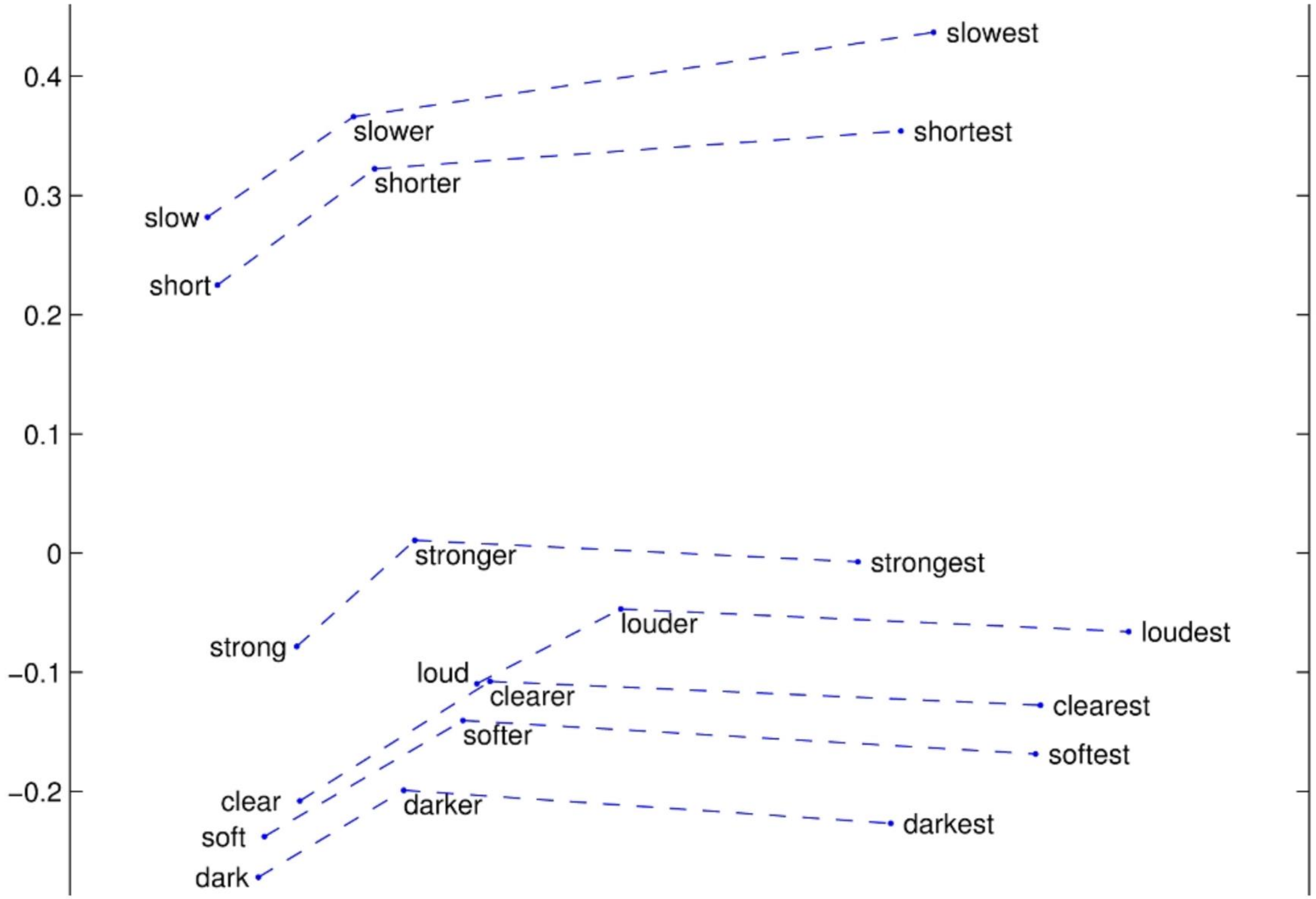
$$\arg\max_{w}\left(\cos(\mathbf{u}_w, \mathbf{u}_{woman} - \mathbf{u}_{man} + \mathbf{u}_{king})\right)$$



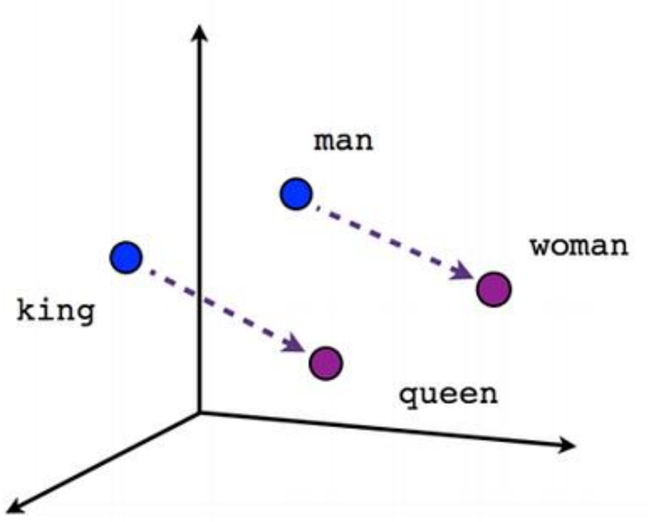
Intrinsic Evaluation: Word Analogy



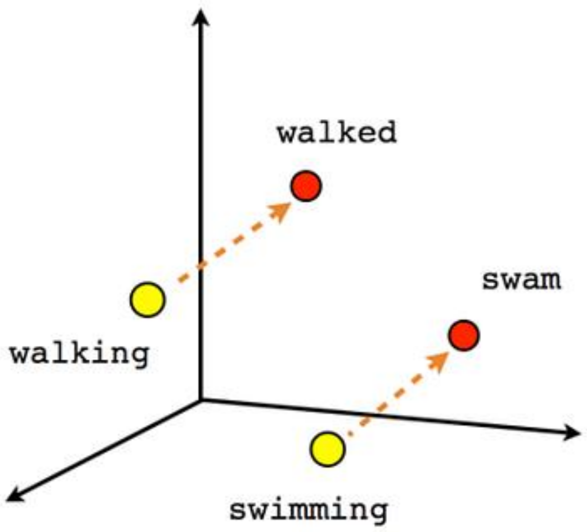
Intrinsic Evaluation: Word Analogy



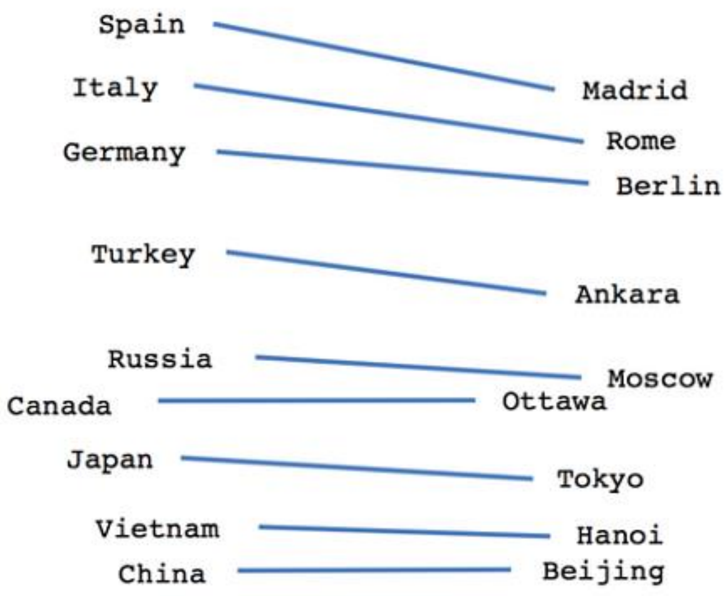
Intrinsic Evaluation: Word Analogy



Male-Female



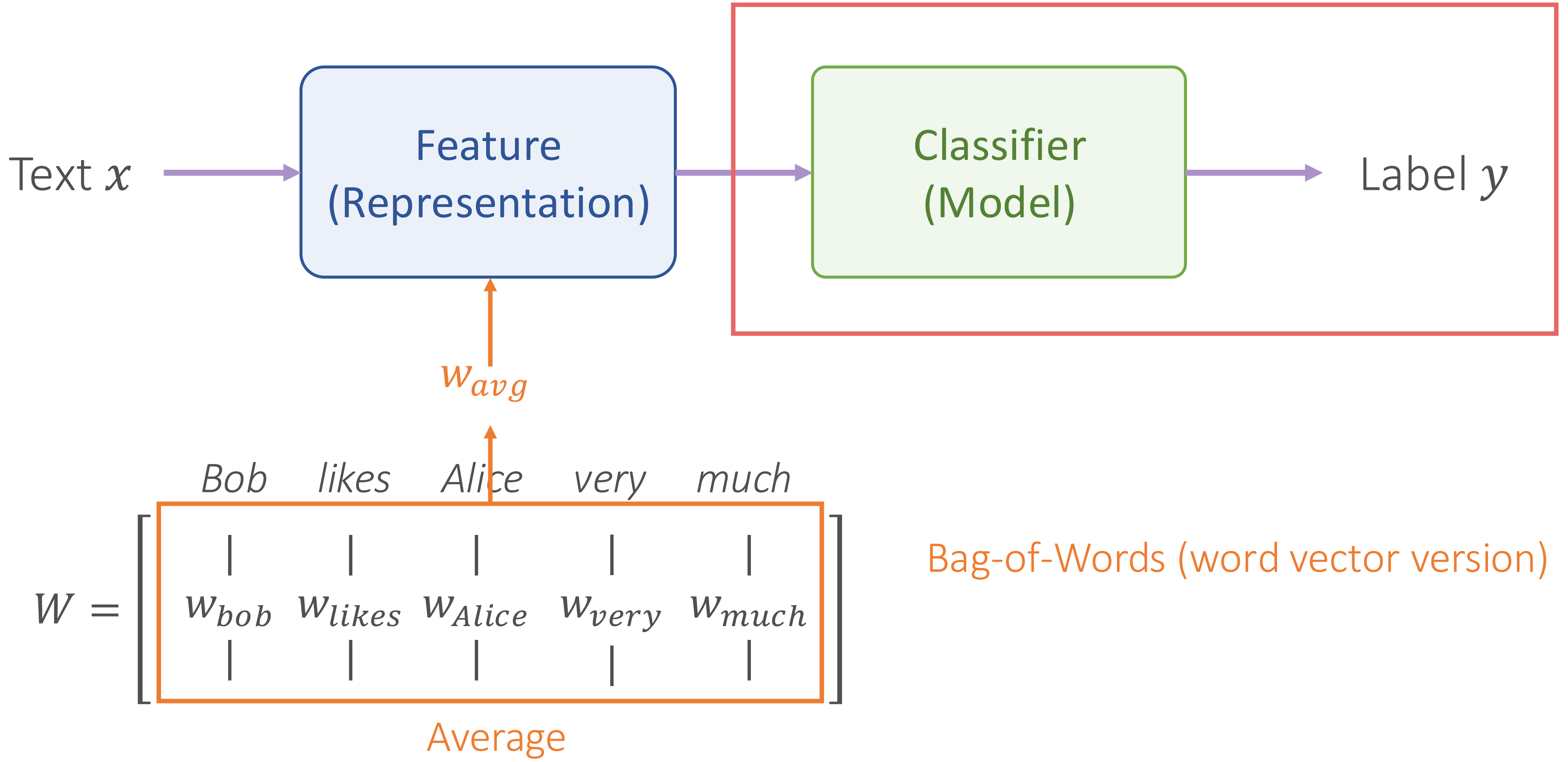
Verb tense



Country-Capital

https://sanjayc.medium.com/word2vec-analogical-reasoning-d47d3a66b9fb

Extrinsic Evaluation: Downstream Performance



Lecture Plan

- Counting-Based Word Vectors
- Learning-Based Word Vectors
- Evaluation for Word Vectors