CSCE 689: Special Topics in Trustworthy NLP

Lecture 3: Natural Language Processing Basics (2)

Kuan-Hao Huang
khhuang@tamu.edu

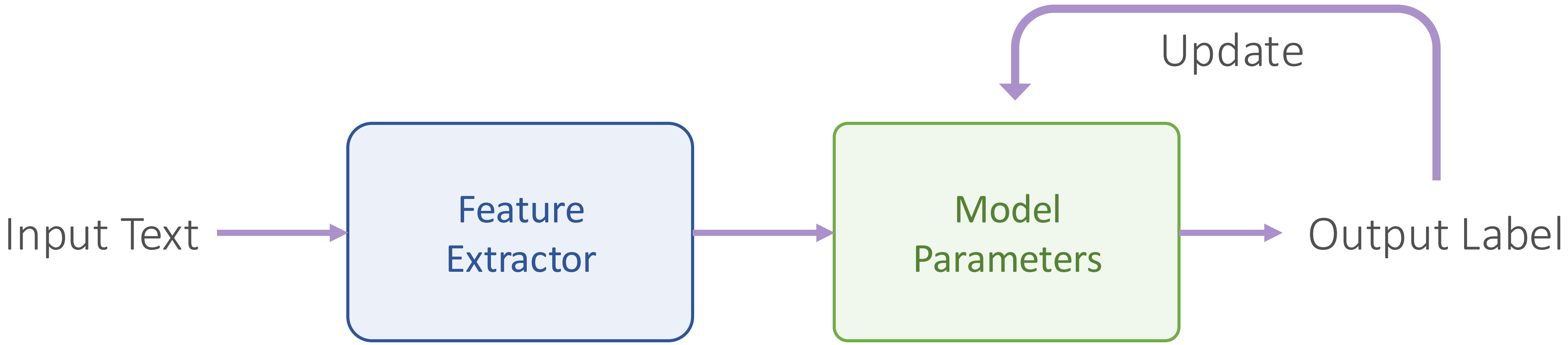
(Some slides adapted from Chris Manning, Dan Jurafsky, Danqi Chen, and Vivian Chen)

Lecture Plan

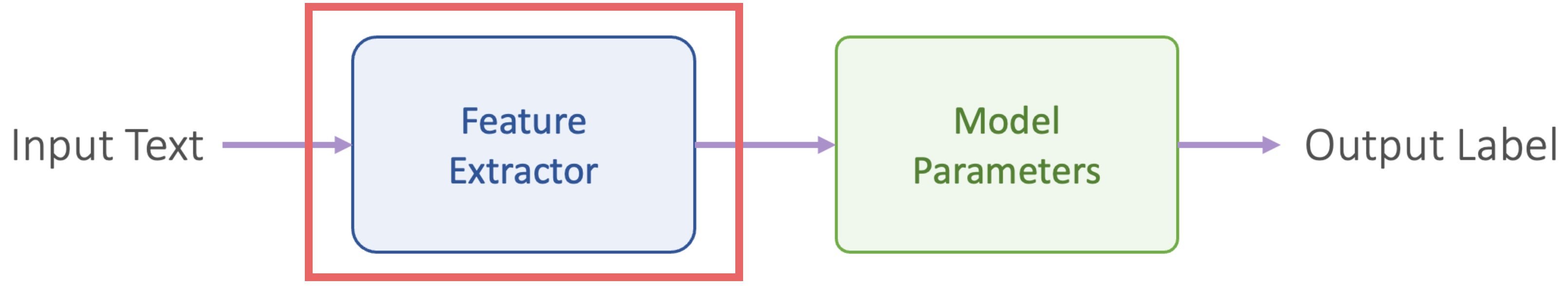
- Natural Language Processing Basics
- Word Embeddings
  - Word2Vec
- Tokenization
  - Byte-Pair Encoding

Recap: How to Learn an NLP model?

- Machine learning method: supervised learning
  - Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
  - Learn model $F: \mathcal{X} \rightarrow \mathcal{Y}$

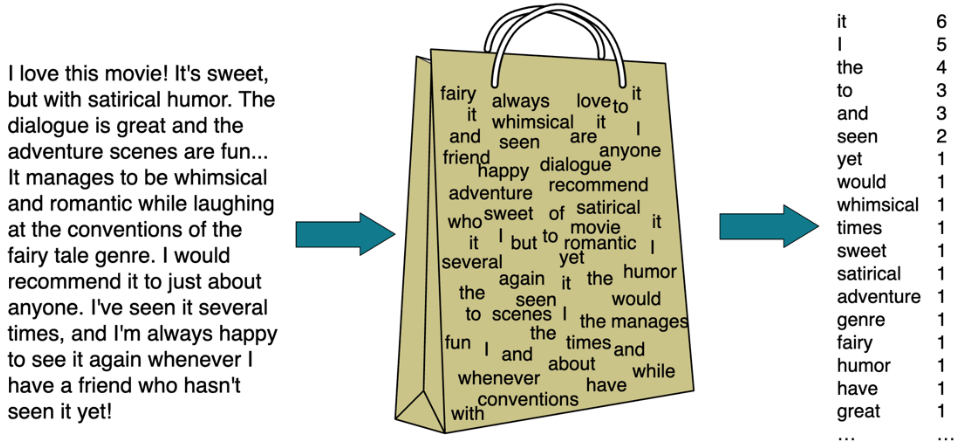


Recap: Human-Crafted Features



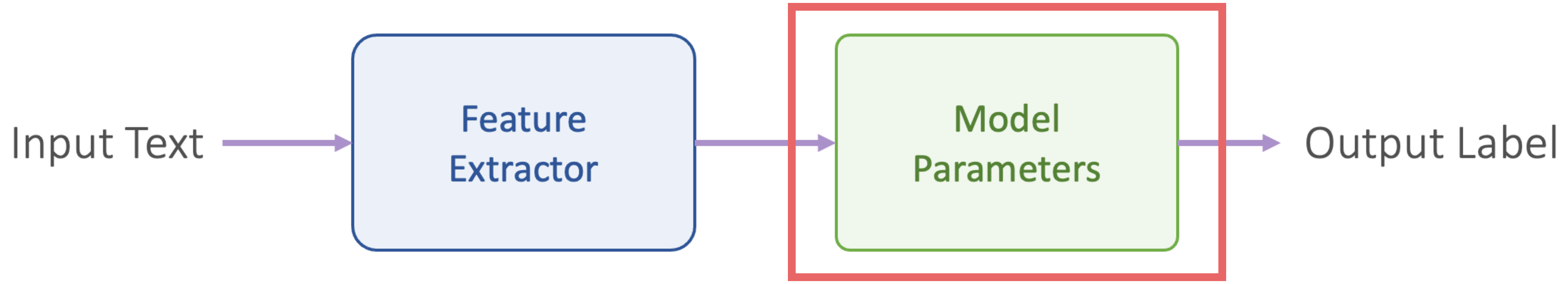
Input Text → A Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$

Bag of words (BoW)



Var	Definition
x_1	$\text{count}(\text{positive lexicon}) \in \text{doc})$
x_2	$\text{count}(\text{negative lexicon}) \in \text{doc})$
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	$\text{count}(\text{1st and 2nd pronouns} \in \text{doc})$
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	$\log(\text{word count of doc})$

Recap: Logistic Regression



- Logistic Regression for multiclass classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$ Label $y = 0,1, \dots, C-1$

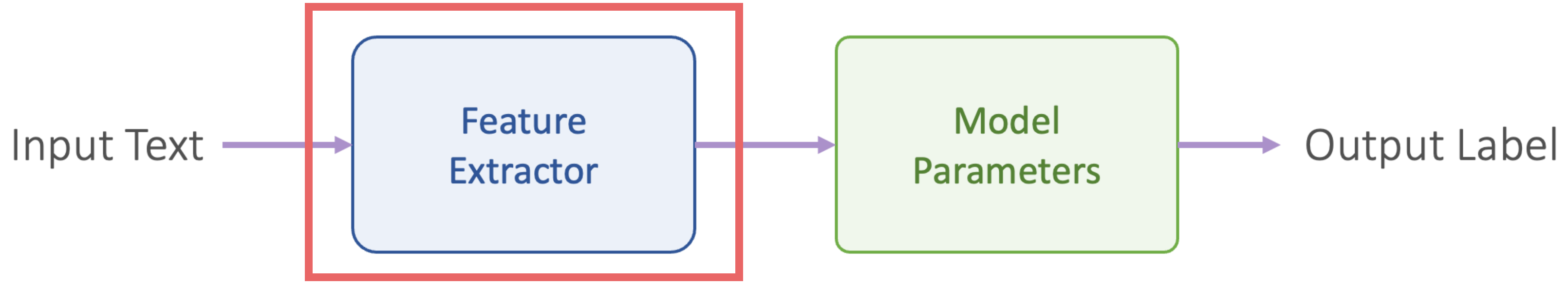
$\boxed{\text{Weight Vectors } \mathbf{w}_c = [w_{c,1}, w_{c,2}, w_{c,3}, \dots, w_{c,n}] \quad \text{Bias } b_c}$ Learnable Model Parameters

$$z_c = \mathbf{w_c} \cdot \mathbf{x} + b_c$$

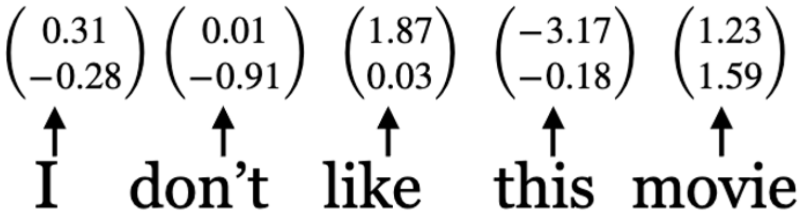
$$P(y = c | \mathbf{x}) = \mathrm{softmax}(z_c) \qquad \mathrm{softmax}(t) = \frac{e^{z_c}}{\sum_c e^{z_c}}$$

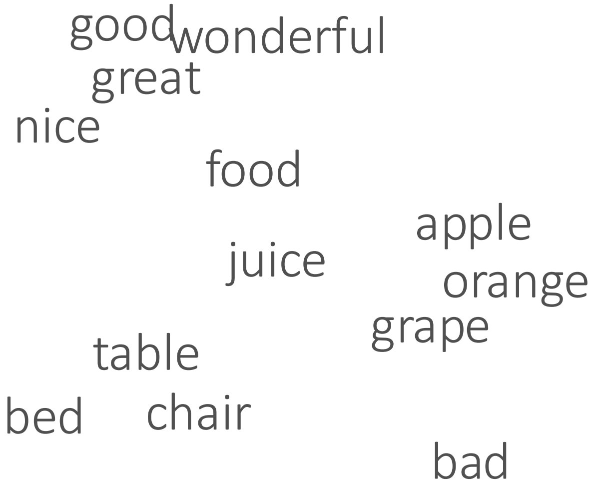
Softmax Function

Recap: Word Embeddings



Input Text → A Sequence of Word Vectors





Representing Words by Their Contexts

Distributional hypothesis: 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

Distributional Hypothesis

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
juice	1	1	0	1
loud	0	0	0	0
motor-oil	1	0	0	1
chips	0	1	0	1
choices	0	1	0	0
wine	1	1	1	1

Words that occur in similar contexts tend to have similar meanings

Word2Vec

- Efficient Estimation of Word Representations in Vector Space, 2013
  - 40000+ 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

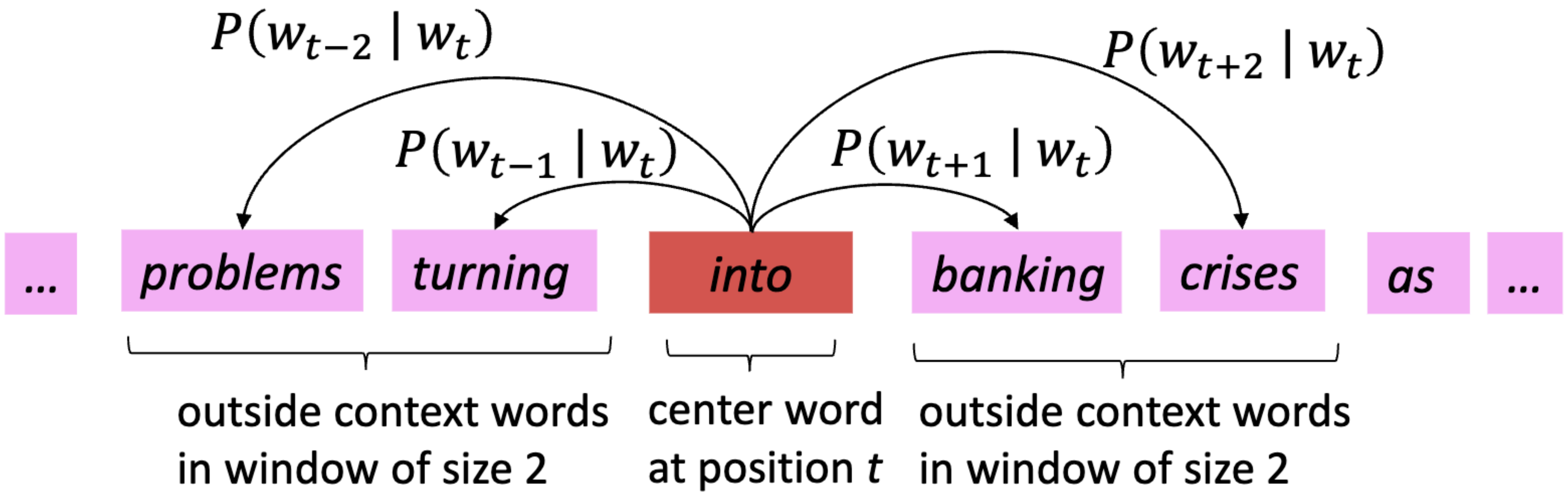
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

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

Word2Vec: Overview

- **The 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}

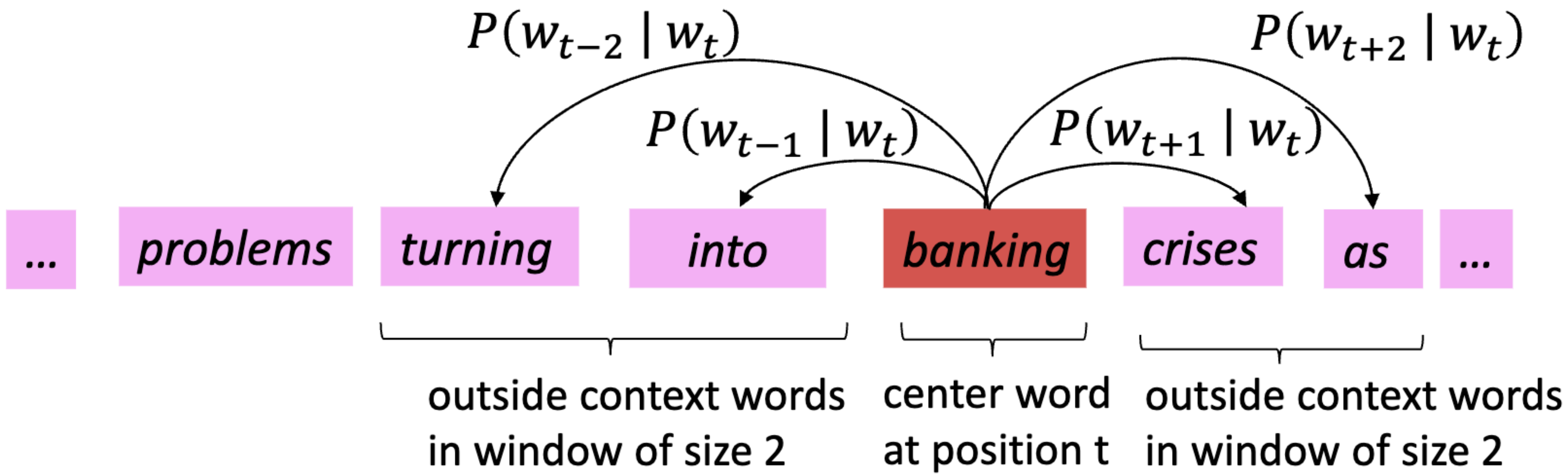


Words that occur in similar contexts tend to have similar meanings

Word2Vec: Overview

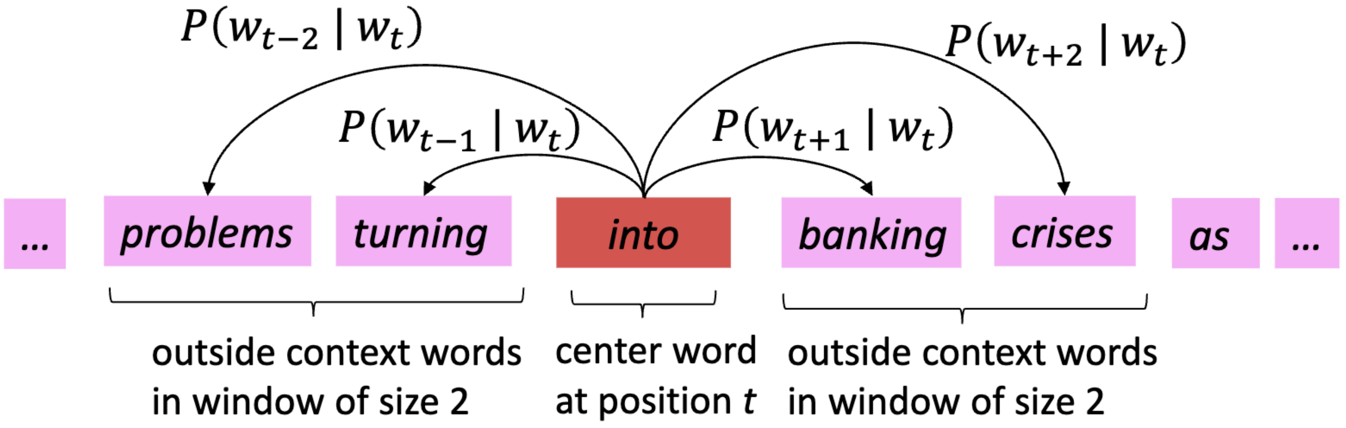
- **The idea:** we want to use words to predict their context words
- Context: a fixed window of size m

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



Words that occur in similar contexts tend to have similar meanings

Word2Vec: Likelihood



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

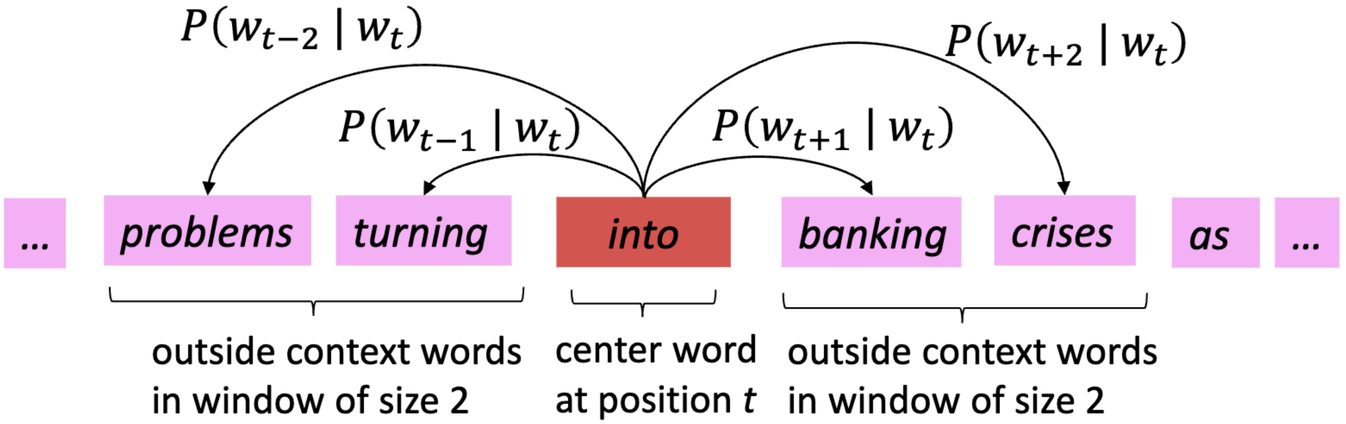
θ all parameters to be optimized

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

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

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

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

How to Define Probability?

Question: how to calculate $P\left(w_{t+j}\middle|\, w_t\,; \theta\right)$?

Answer: 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

We consider Inner product $\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}}$ as the score to measure how likely the context word w_{t+j} appears with the center word w_t, the larger the more likely!

$$P\left(w_{t+j}\middle|\, w_t\,; \theta\right) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k\in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)} \qquad \theta = \{\{\mathbf{u}_k\}, \{\boldsymbol{v}_k\}\} \text{ all parameters}$$

How to Define Probability?

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

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

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

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

Why Two Sets of Vectors?

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

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

- Scores can be asymmetric
- It is not likely that a word appears in its own context

How to Train Word Vectors?

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

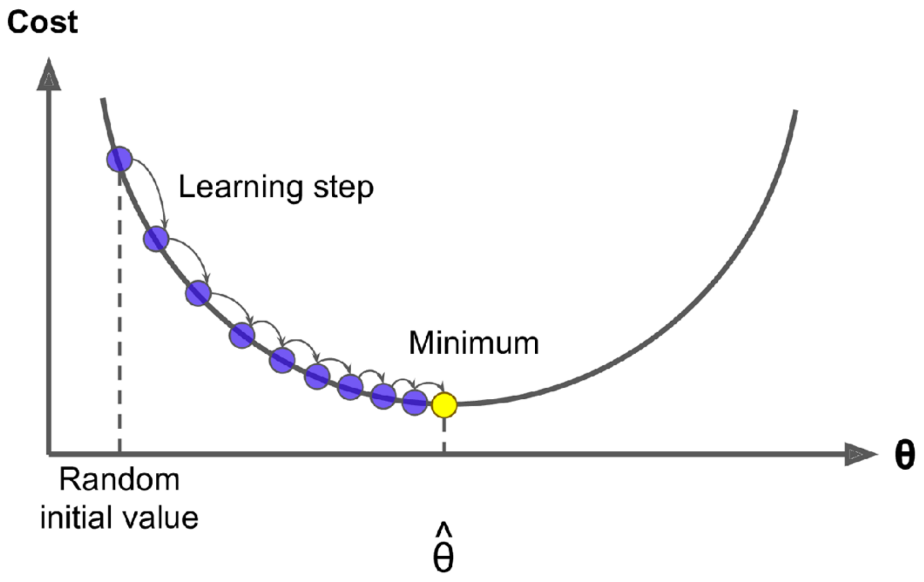
Objective function: $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} \log P\left(w_{t+j} \middle| w_t ; \theta\right)$$

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 Gradient



Warm-Up

$$f(x) = \exp(x) \qquad \frac{df}{dx} = \exp(x)$$

$$f(x) = \log(x) \qquad \frac{df}{dx} = \frac{1}{x}$$

Chain Rule

$$f(x) = f_1(f_2(x)) \qquad \frac{df}{dx} = \frac{df_1(z)}{dz}\frac{df_2(x)}{dx} \qquad z = f_2(x)$$

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

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

Computing the Gradients

Objective function

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

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

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!

Computing the Gradients

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

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

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

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

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

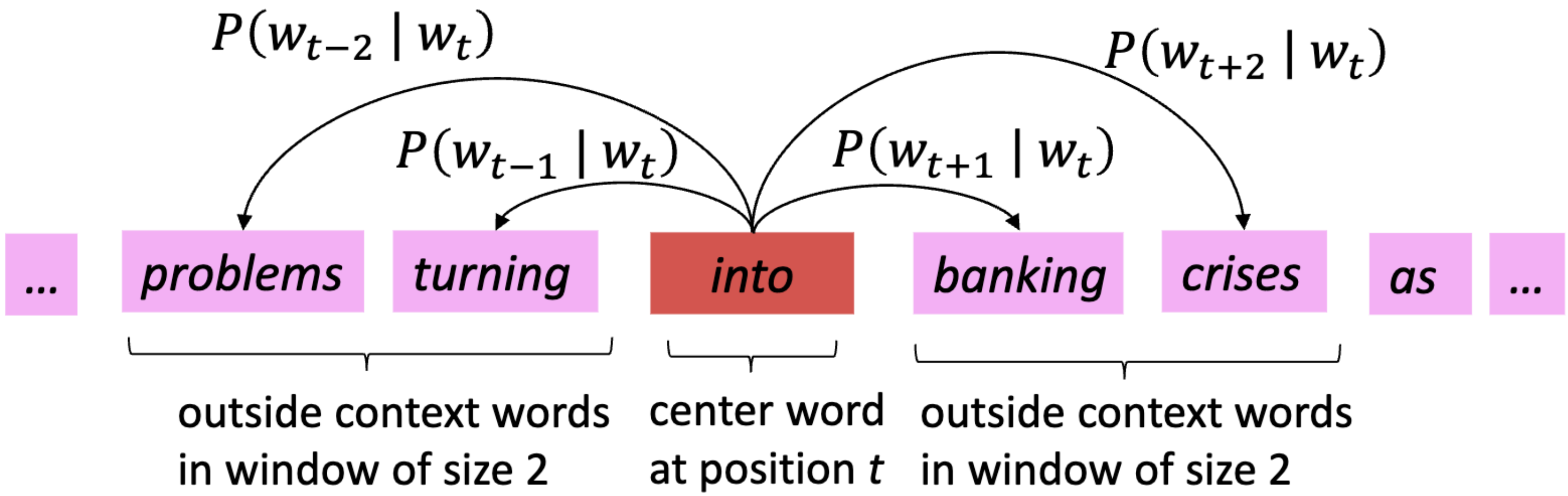
$$= -\mathbf{v}_c + \sum_{k\in V}P(k|o)\,\mathbf{v}_k$$

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

Similar calculation step

Training Process

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



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

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.

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

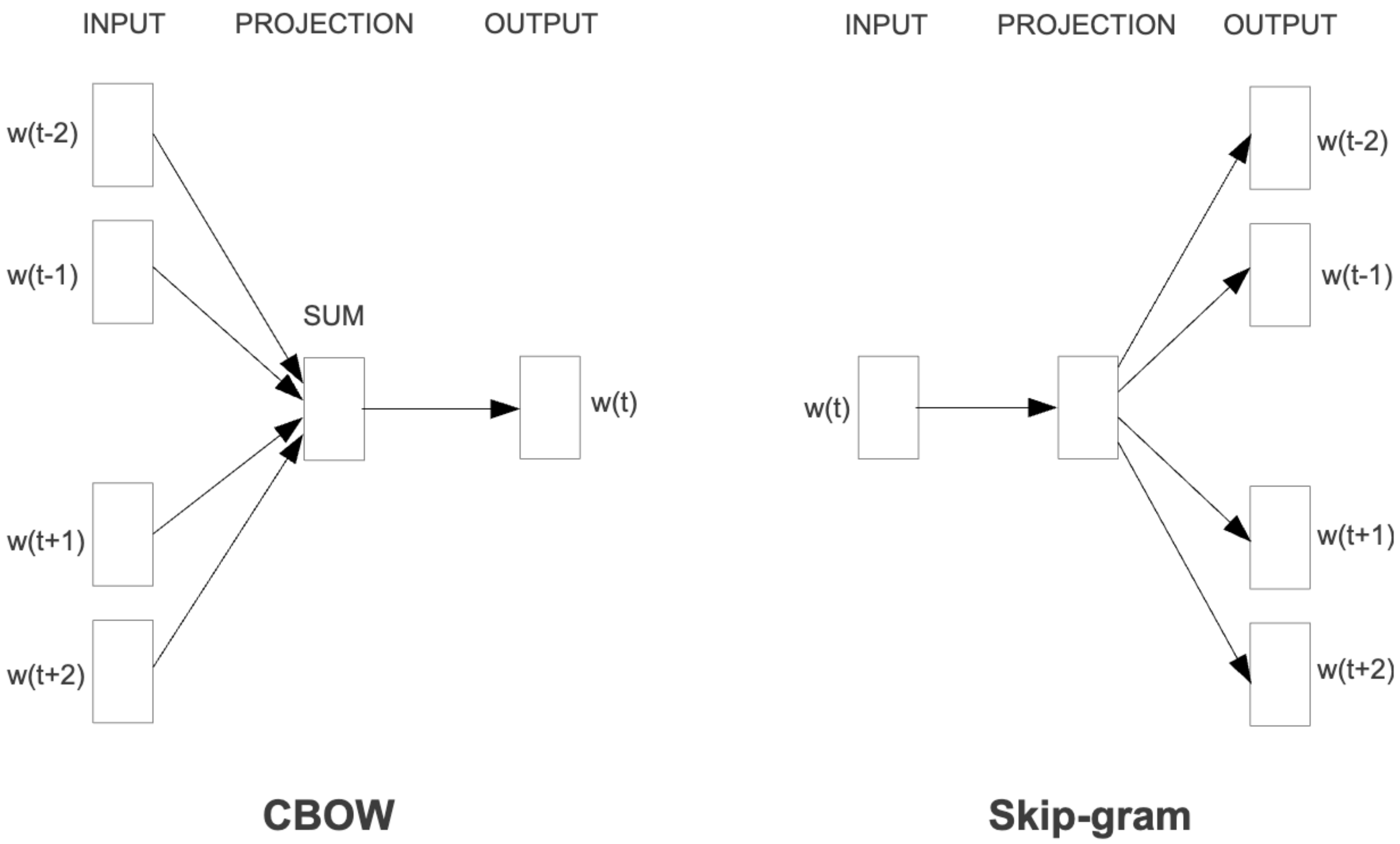
Negative sampling: instead of considering all the words in V, we randomly sample K(5-20) negative examples

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

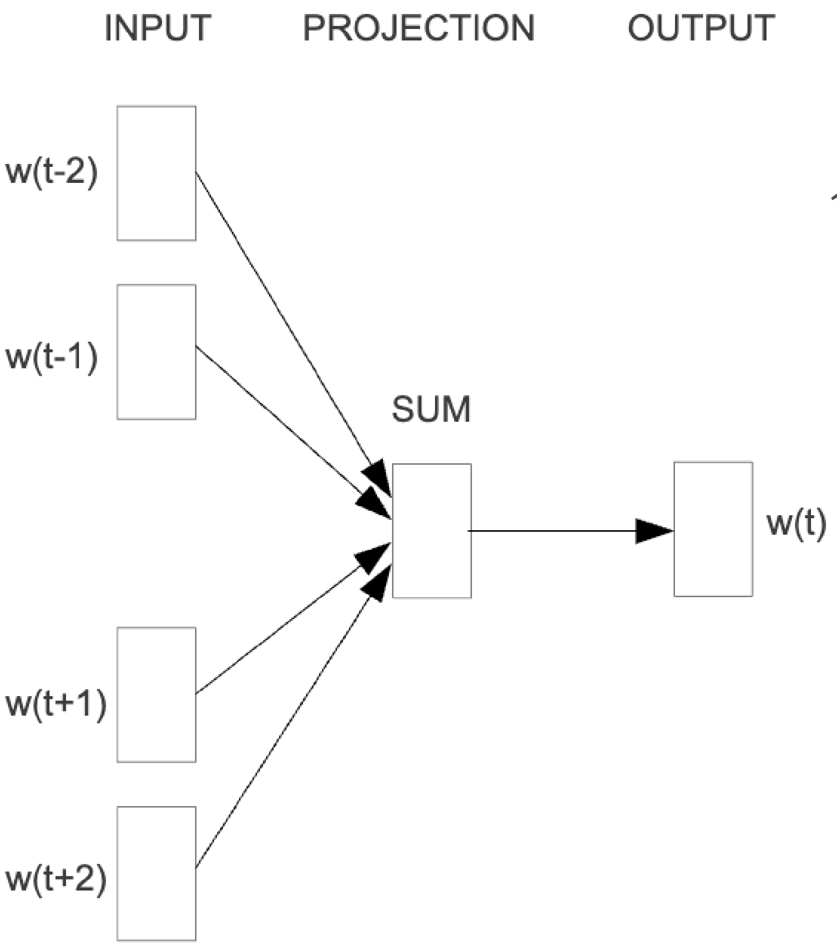
Negative sampling $\quad y = -\log(\sigma(\mathbf{u}_o \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_o \cdot \mathbf{v}_j))$

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

Continuous Bag of Words (CBOW) vs Skip-Grams



Continuous Bag of Words (CBOW)



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

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

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

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/

Word Analogy Test

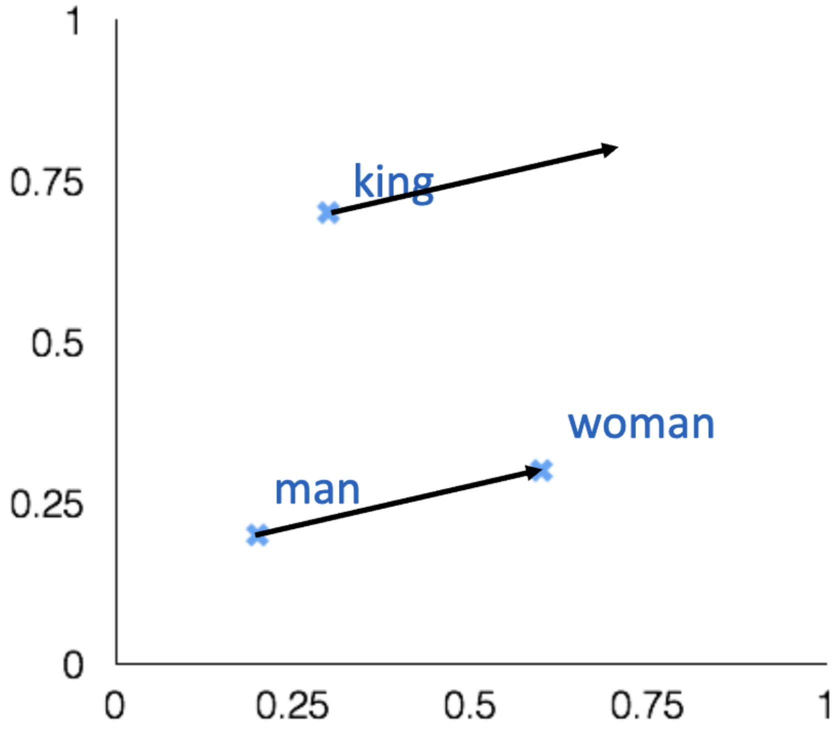
Word analogy

man: woman ≈ king: ?

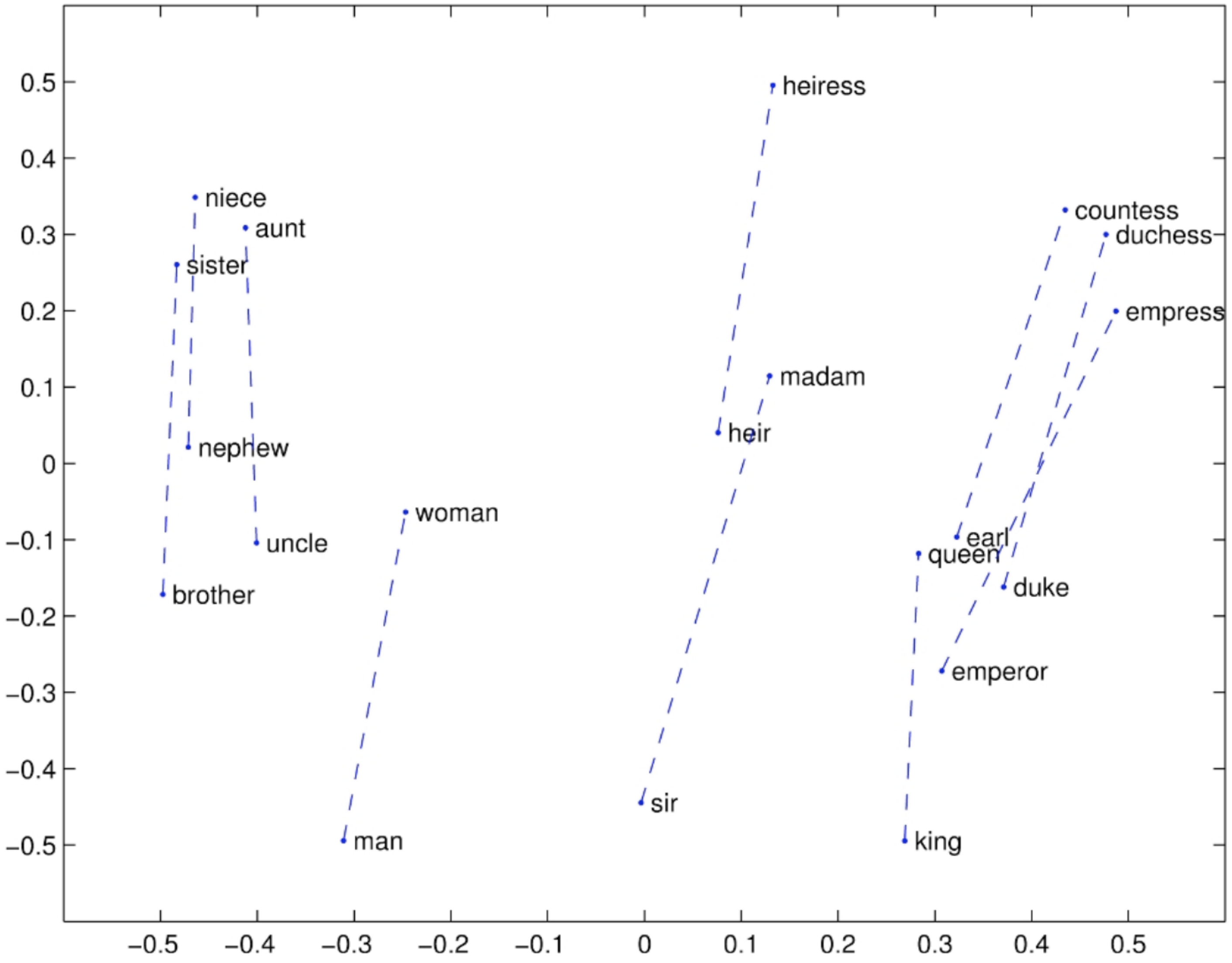
Paris: France ≈ London: ?

bad: worst ≈ cool: ?

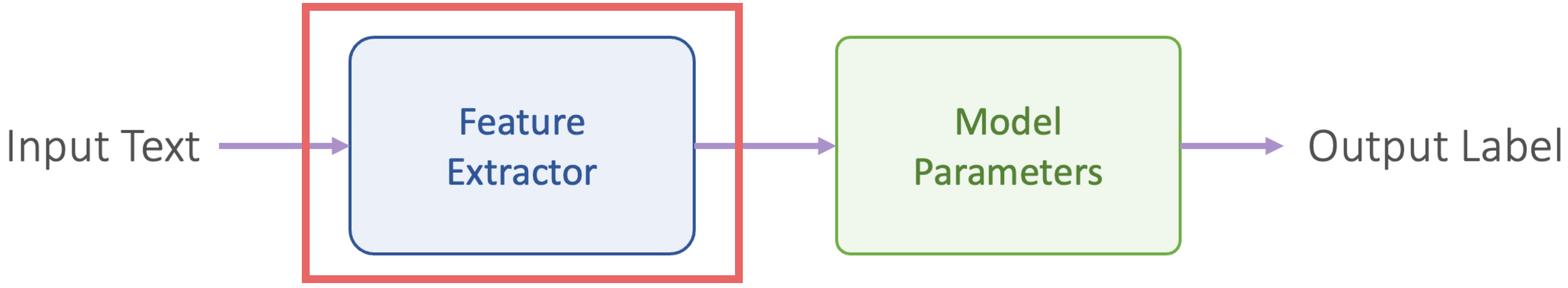
$$\arg\max_{w}\left(\cos(\mathbf{u}_w, \mathbf{u}_{woman} - \mathbf{u}_{man} + \mathbf{u}_{king})\right)$$



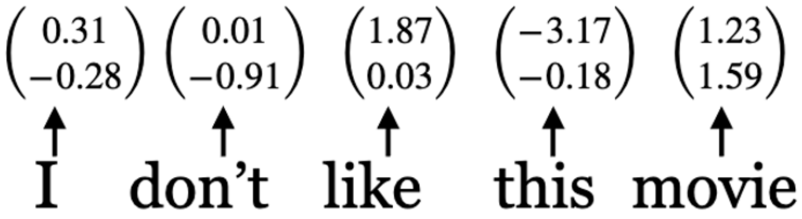
Visualization of Word Vectors

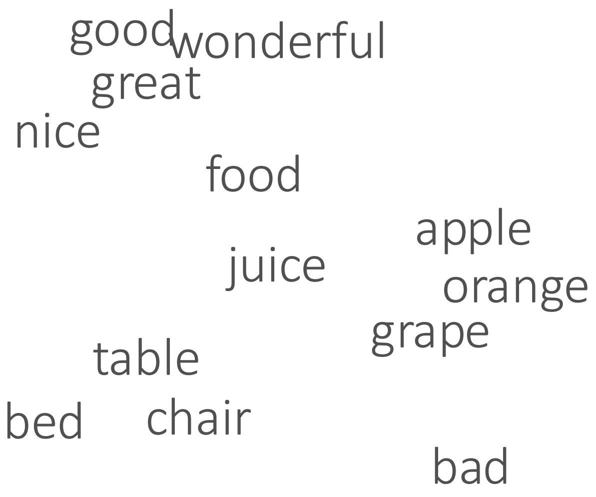


Word Embeddings



Input Text → A Sequence of Word Vectors





Lecture Plan

- Natural Language Processing Basics
- Word Embeddings
  - Word2Vec
- Tokenization
  - Byte-Pair Encoding

Tokenization

- Currently, we use word (and punctuation) as the basic unit to tokenize a text
  - I like this movie so much. → I + like + this + movie + so + much + .

What is the size of word embeddings (how many words)?

Size of Vocabulary

- The larger, the better?
- Storage? Computation?
- Do we need to consider all the words?
  - zcvahu
  - #$^&*
  - Low frequency words

Unknown Token

- We create an unknown token for all the words that have never been seen or low frequency words
  - <UNK>
- <UNK> has its own embedding
  - I like this movie &*# so much → I + like + this + movie + <UNK> + so + much + .
  - I like this movie sooooo much. → I + like + this + movie + <UNK> + much + .
- We can reduce the size of vocabulary
- We can handle unseen words

Is There A Better Way?

- We can guess the meaning of some unknown words
  - soooooooo
  - taaaasty
  - Transformerify
- Some words share the same prefix or suffix
  - happy, happier, happiest
  - drive, driving, driven
  - unlikely, unhappy, unhealthy
  - beautiful, trustful, grateful

Subword Tokenization

- We use subword (and punctuation) as the basic unit to tokenize a text
- Subword: parts of words
  - happy, happier, happiest: happ-, -y, -ier, -iest
  - drive, driving, driven: driv-, -e, -ing, -en
  - beautiful, trustful, grateful: -ful

Next Lecture

- Natural Language Processing Basics
- Tokenization
  - Byte-Pair Encoding
- Common Models
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Long Short-Term Memory (LSTM)