CS769 Advanced NLP

Word Embeddings and Text Classification

Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Noah, Yulia

https://junjiehu.github.io/cs769-fall25/

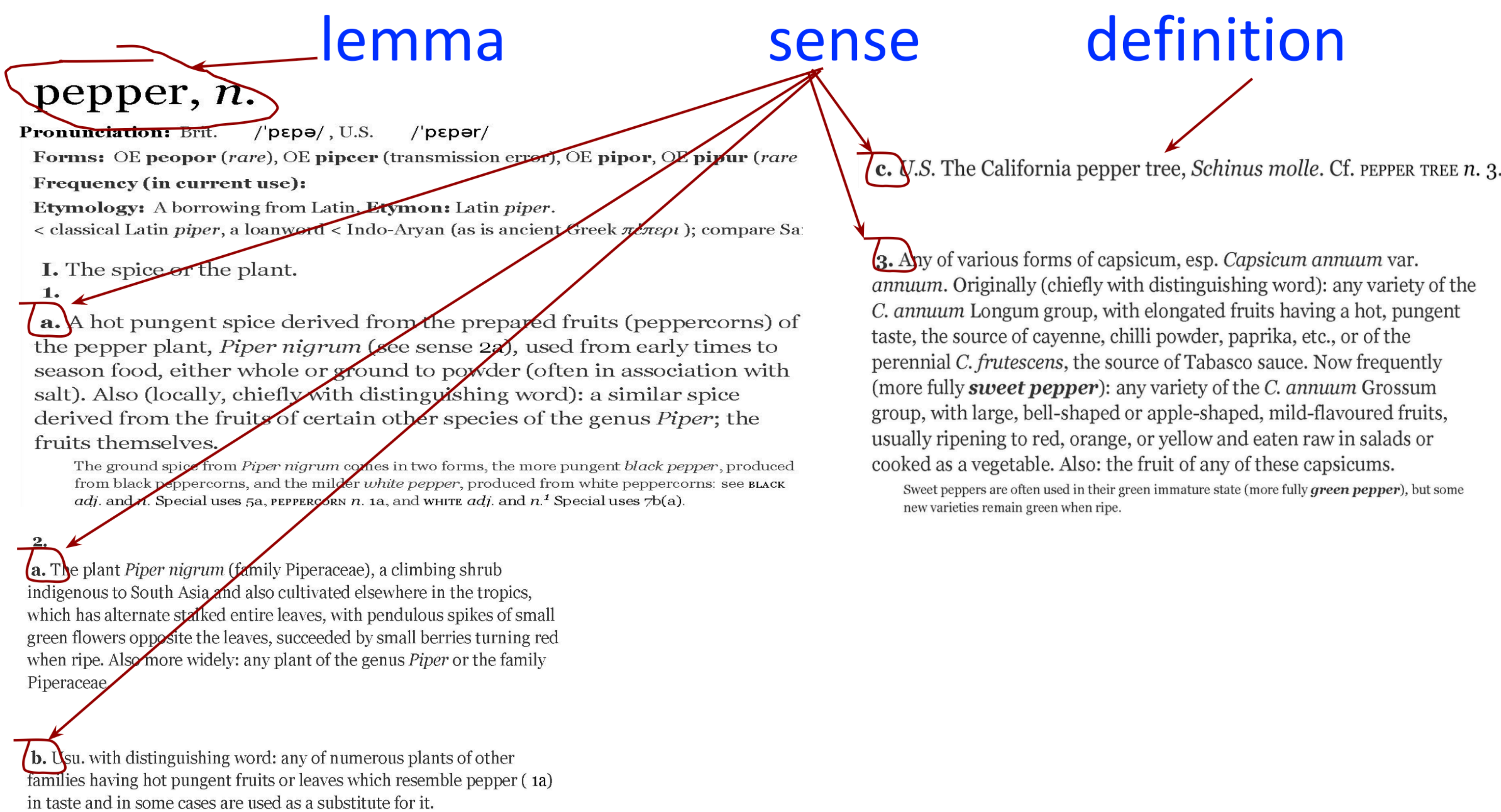
Goals for Today

- Lexical Semantics and Distributional Semantics
- **Word Embeddings** (e.g., Skip-gram, CBOW)
- Evaluation (intrinsic and extrinsic)
- Text Classification

How should we represent the meaning of the word?

Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definition



pepper, *n.*

Pronunciation: Brit. /ˈpɛpə/ , U.S. /ˈpɛpər/

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare*

Frequency (in current use):

Etymology: A borrowing from Latin. **Etymon:** Latin *piper*.

< classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek πέπερι); compare Sa

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper*, produced from black peppercorns, and the milder *white pepper*, produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.* 3.

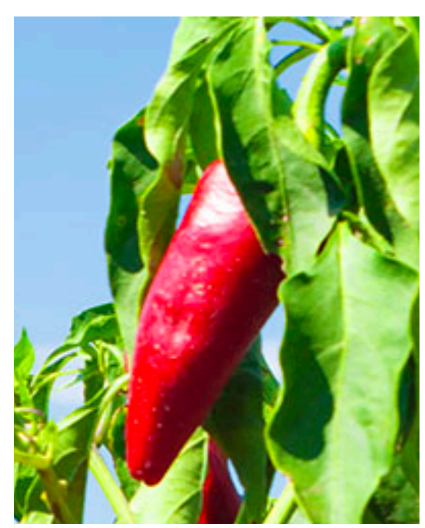
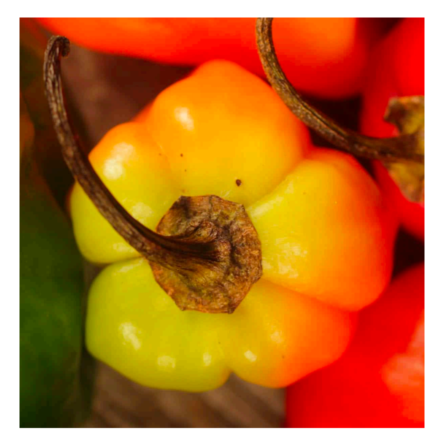
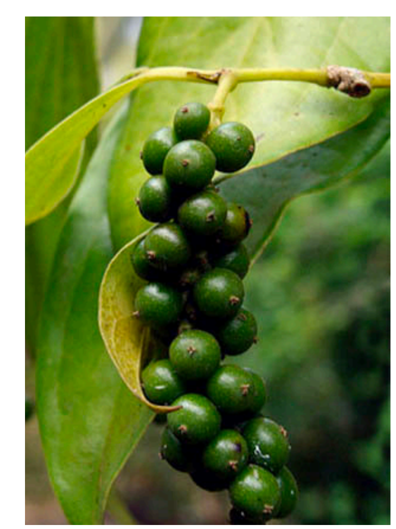
3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the perennial *C. frutescens*, the source of Tabasco sauce. Now frequently (more fully ***sweet pepper***): any variety of the *C. annuum* Grossum group, with large, bell-shaped or apple-shaped, mild-flavoured fruits, usually ripening to red, orange, or yellow and eaten raw in salads or cooked as a vegetable. Also: the fruit of any of these capsicums.

Sweet peppers are often used in their green immature state (more fully ***green pepper***), but some new varieties remain green when ripe.

Oxford English Dictionary: https://www.oed.com/

Lemma pepper

- Sense 1: spice from pepper plant
- Sense 2: the pepper plant itself
- Sense 3: another similar plant (Jamaican pepper)
- Sense 4: plant with peppercorns (California pepper)
- Sense 5: capsicum (i.e., chili, paprika, bell pepper, etc)

Lexical Semantics

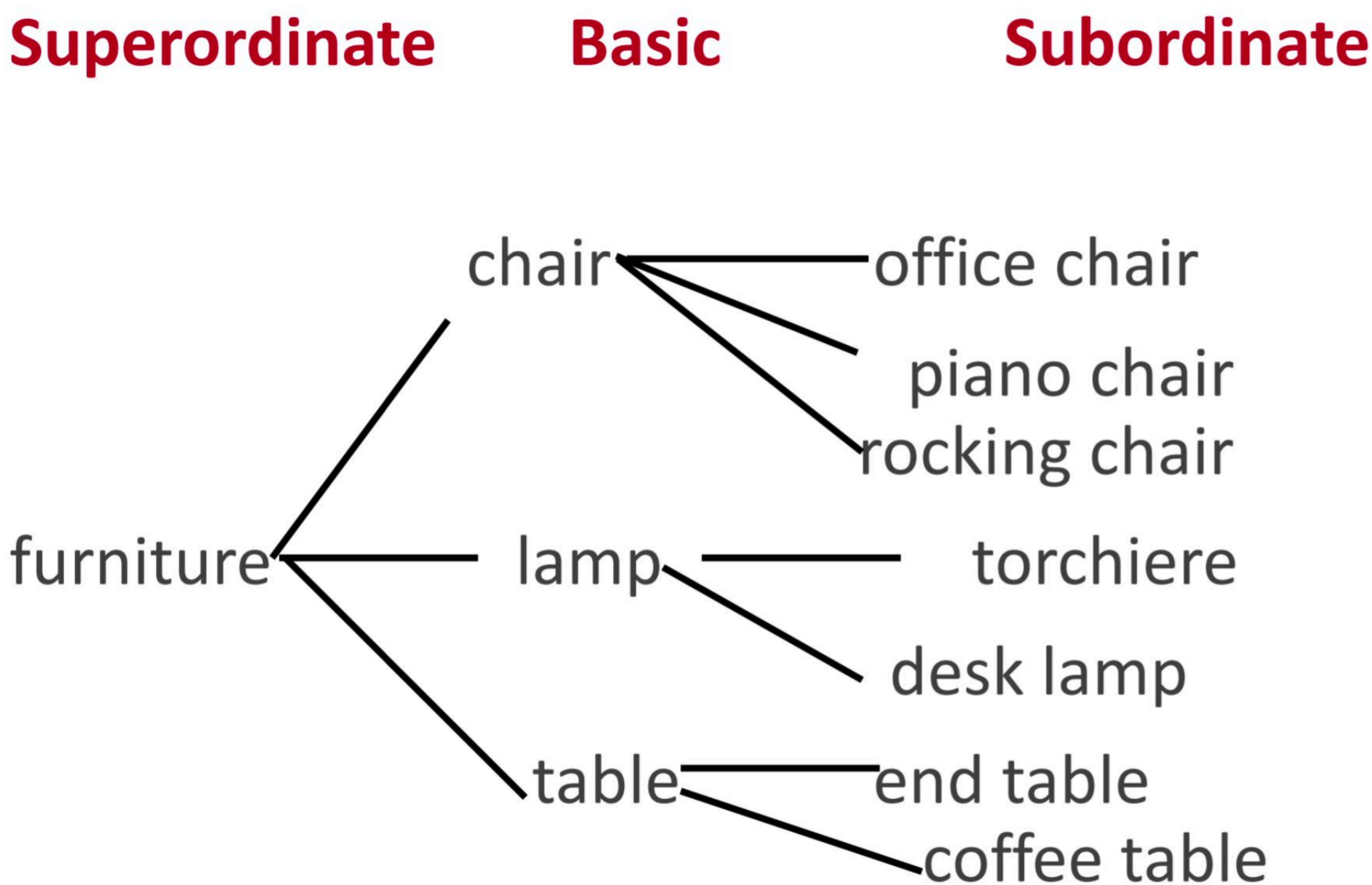
- How should we represent the meaning of the word?
  - Words, lemmas, senses, definition
  - Relationships between words or senses
    1. Synonymity: same meaning, e.g., couch/sofa
    2. Antonymy: opposite senses, e.g., hot/cold
    3. Similarity: similar meanings, e.g., car/bicycle
    4. Relatedness: association, e.g., car/gasoline
    5. Superordinate/Subordinate: e.g., car/vehicle, mango/fruit

Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definition
  - Relationships between words or senses
  - Taxonomy: abstract -> concrete

Taxonomy

- abstract -> concrete



Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definition
  - Relationships between words or senses
  - Taxonomy: abstract -> concrete
  - Semantic frames and roles

Semantic Frame

- A set of words that denote perspectives or participants in an event
  - Tom brought a book from Bill.

buyer event from the perspective of the buyer

  - Bill sold a book to Tom.

seller event from the perspective of the seller

Mismatch

- Theories of language tend to view the data (words, sentences, documents) and abstractions over it as *symbolic* or categorical.
    - Uses symbols to represent linguistic information

- Machine learning algorithms built on optimization rely more on *continuous* data.
    - Uses floating-point numbers (vectors)

Problems with Discrete Representations

- Too coarse: *expert* ↔ *skillful*
- Sparse
- Subjective
- Expensive
- Hard to compute word relationships

One-hot vector:

expert	[0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0]
skillful	[0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0]

Distributional Hypothesis

"The meaning of a word is its use in the language"

[Wittgenstein 1943]

"You shall know a word by the company it keeps"

[Firth 1957]

"If A and B have almost identical environments we say that they are synonyms."

[Harris 1954]

Example

- What does “Ong Choy” mean?
  - Suppose you see these sentences:
    - Ong Choy is delicious **sautéed with garlic**
    - Ong Choy is superb **over rice**
    - Ong Choy **leaves** with salty sauces
  - And you’ve also seen these:
    - … water spinach **sautéed with garlic over rice**
    - Chard stems and **leaves** are delicious
    - Collard greens and other **salty leafy** greens

Ong Choy ≈ “Water Spinach”?

- Ong Choy is a leafy green like spinach, chard, or collard greens

Ong Choy: pronunciation of “蕹菜” in Cantonese

Model of Meaning Focusing on Similarity

- Each word = a vector
  - Similar words are “nearby in space”
  - the standard way to represent meaning in NLP

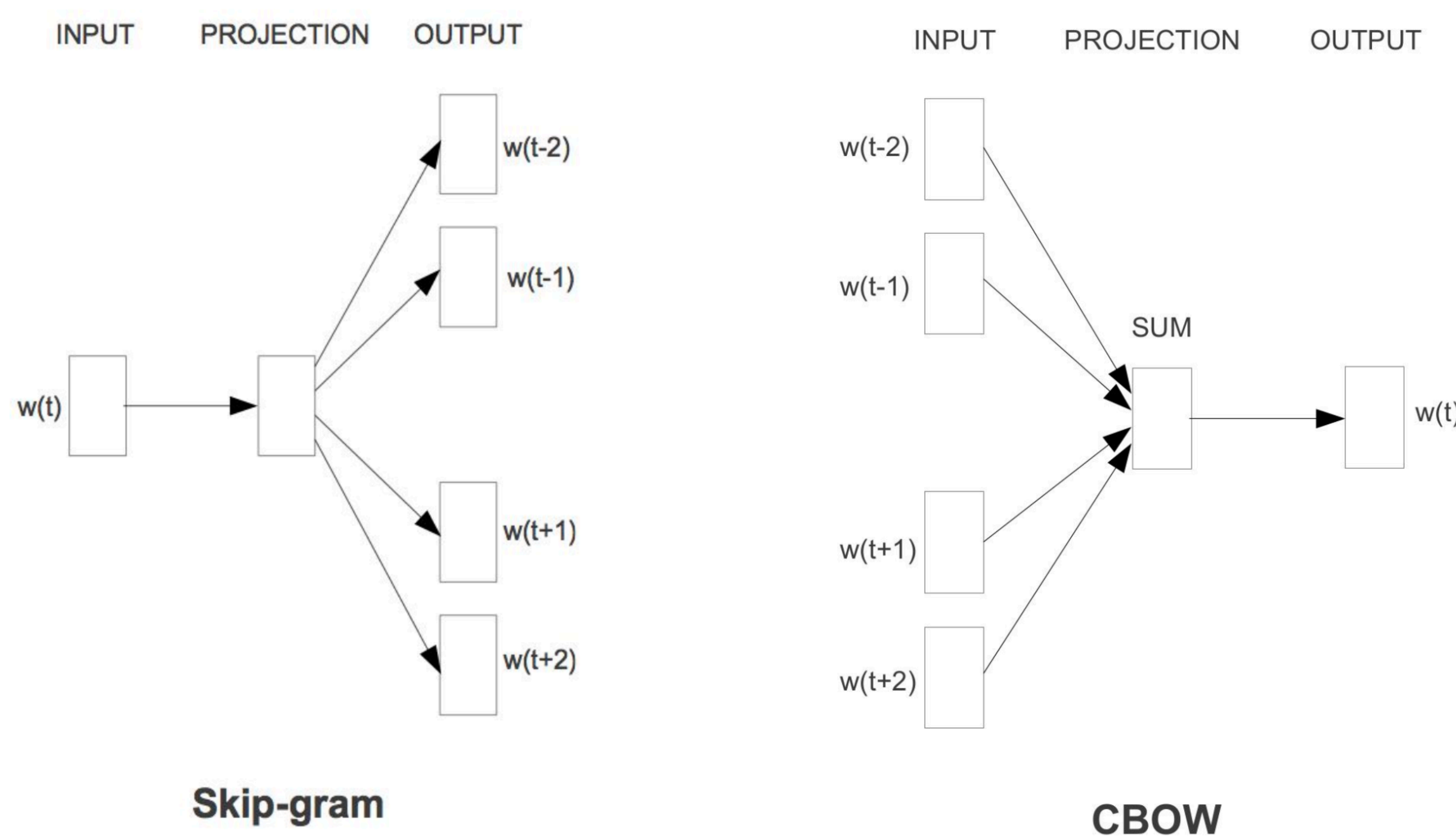
Distributed Word Embeddings

Word Vector Models

- These models are designed to “guess” a word at position i given a word at a position in $\{i-w,\ldots,i-1\} \cup \{i+1,\ldots,i+w\}$
- “Pre-train” word vectors are used in other larger models (e.g., neural LM)

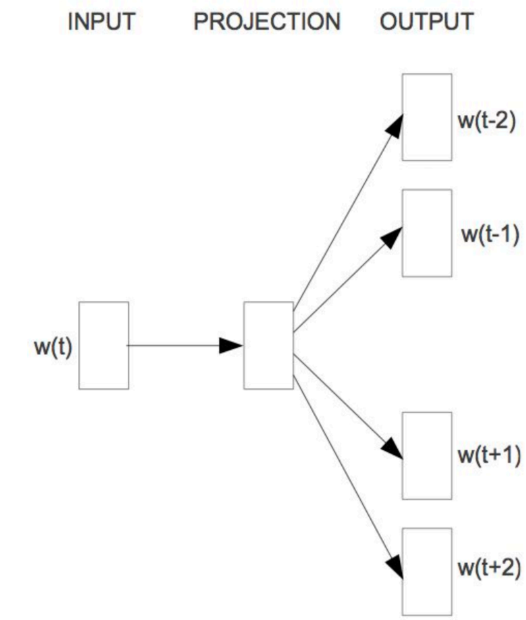
Word2vec

- Continuous bag of words (CBOW): $p(v \mid c)$
  - Similar to feedforward neural LM w/o the feedforward layers in Lecture 3.
- Skip-gram: $p(c \mid v)$



Skip-gram Prediction

- Predict vs Count



Skip-gram

the cat sat on the mat

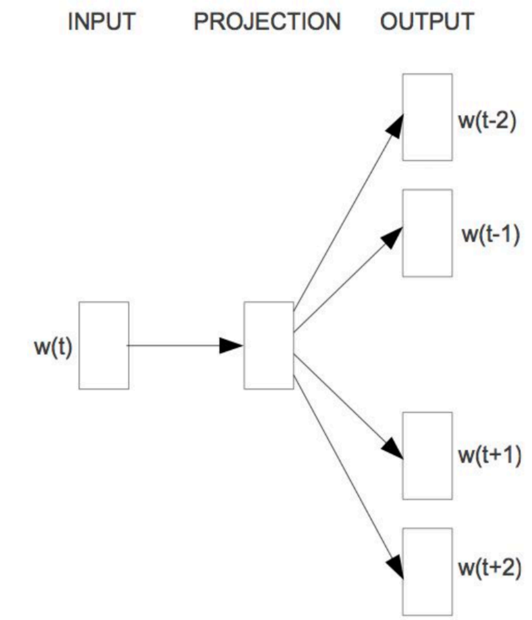
w_t = the → **CLASSIFIER** →

$w_{t-2} = <\text{start}_{-2}>$
$w_{t-1} = <\text{start}_{-1}>$
w_{t+1} = cat
w_{t+2} = sat

context size = 2

Skip-gram Prediction

- Predict vs Count



Skip-gram

the cat sat on the mat

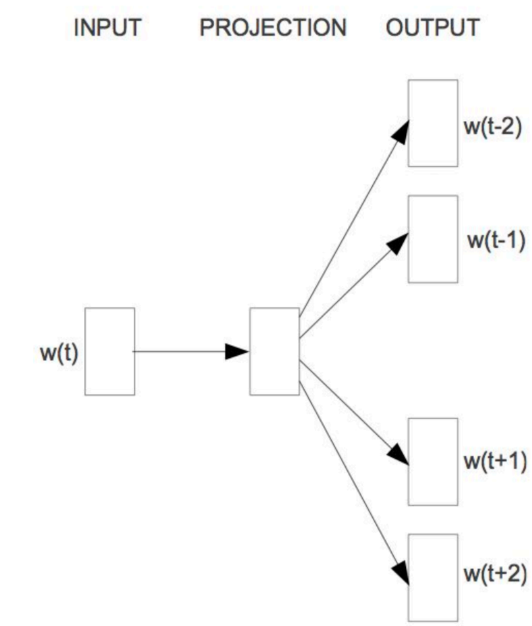
w_t = cat → **CLASSIFIER** →

w_{t-2} = <start$_{-1}$>
w_{t-1} = the
w_{t+1} = sat
w_{t+2} = on

context size = 2

Skip-gram Prediction

- Predict vs Count



Skip-gram

the cat sat on the mat

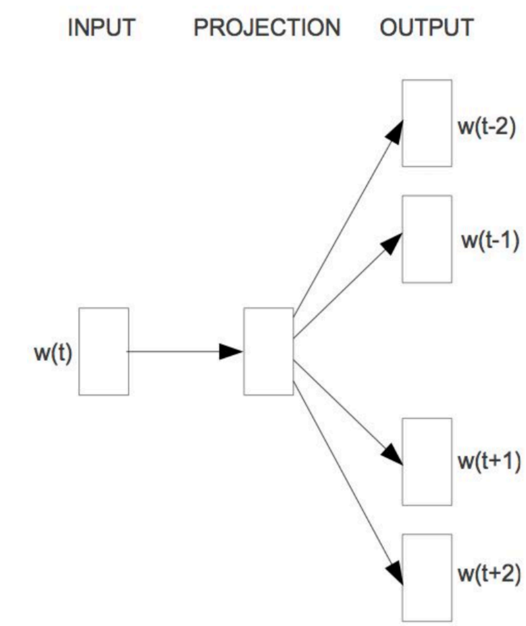
w_t = sat → **CLASSIFIER** →

w_{t-2} = the
w_{t-1} = cat
w_{t+1} = on
w_{t+2} = the

context size = 2

Skip-gram Prediction

- Predict vs Count



Skip-gram

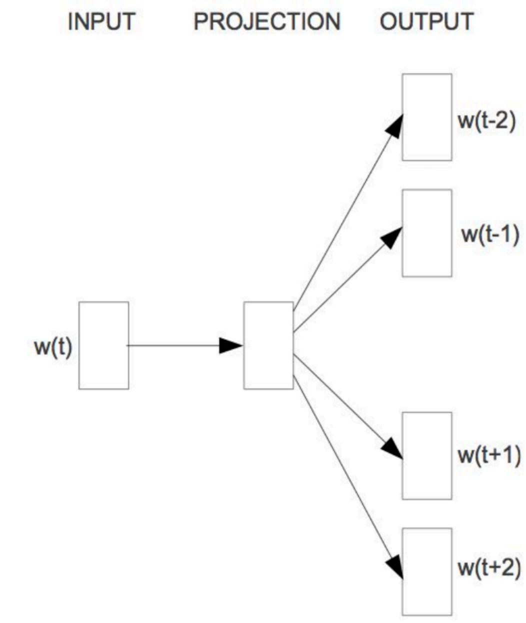
the cat sat <u>on</u> the mat

w_t = on → **CLASSIFIER** → w_{t-2} = cat, w_{t-1} = sat, w_{t+1} = the, w_{t+2} = mat

context size = 2

Skip-gram Prediction

- Predict vs Count



Skip-gram

the cat sat on the mat

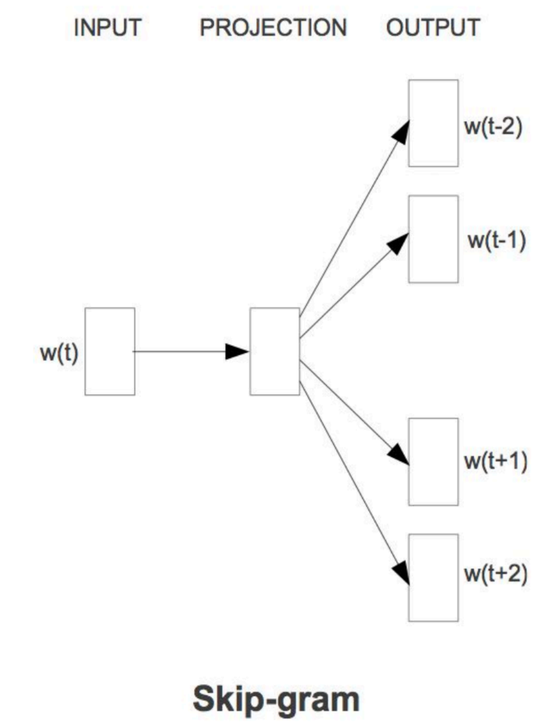
w_t = the → CLASSIFIER →

w_{t-2} = sat
w_{t-1} = on
w_{t+1} = mat
w_{t+2} = <end$_{+1}$>

context size = 2

Skip-gram Prediction

- Predict vs Count



the cat sat on the <u>mat</u>

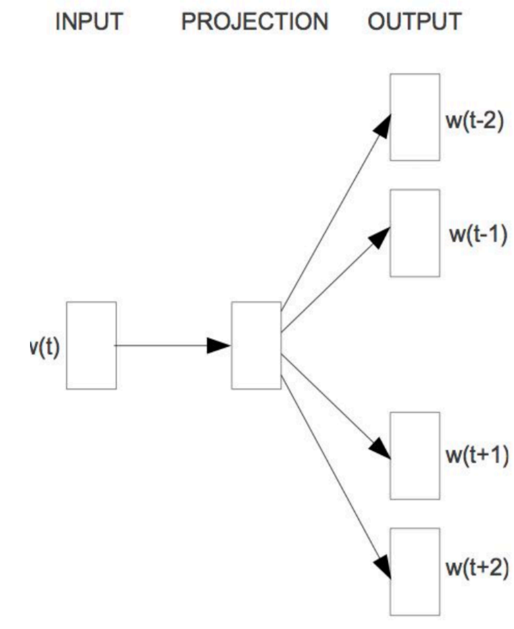
w_t = mat → **CLASSIFIER** →

w_{t-2} = on
w_{t-1} = the
$w_{t+1} = \langle \text{end}_{+1} \rangle$
$w_{t+2} = \langle \text{end}_{+2} \rangle$

context size = 2

Skip-gram Prediction

- The same word can appear in different context.



Skip-gram

w_t = the → **CLASSIFIER** →

w_{t-2} = sat
w_{t-1} = on
w_{t+1} = mat
w_{t+2} = <end_{+1}>

w_t = the → **CLASSIFIER** →

w_{t-2} = <start_{-2}>
w_{t-1} = <start_{-1}>
w_{t+1} = cat
w_{t+2} = sat

context size = 2

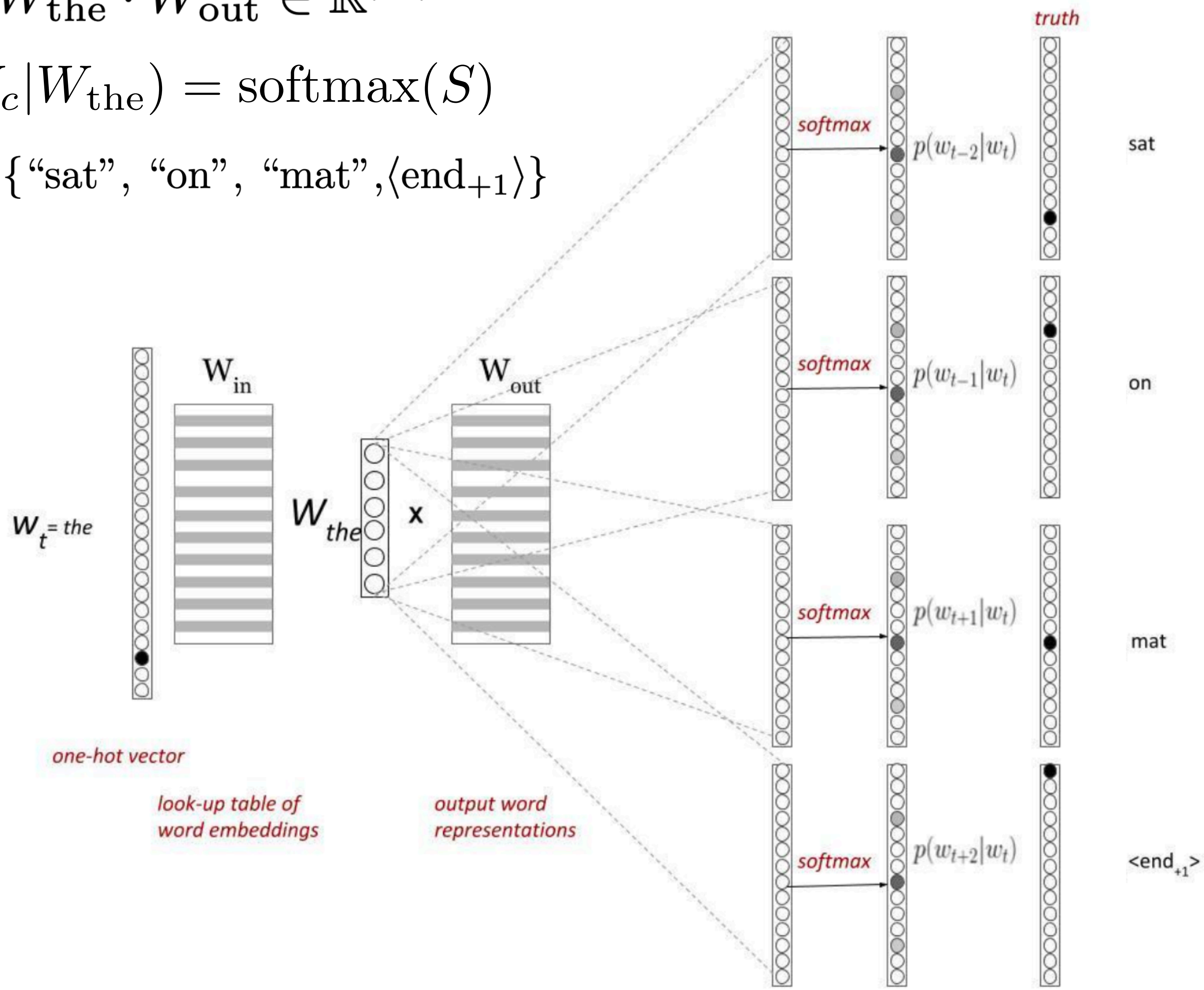
Skip-gram Prediction

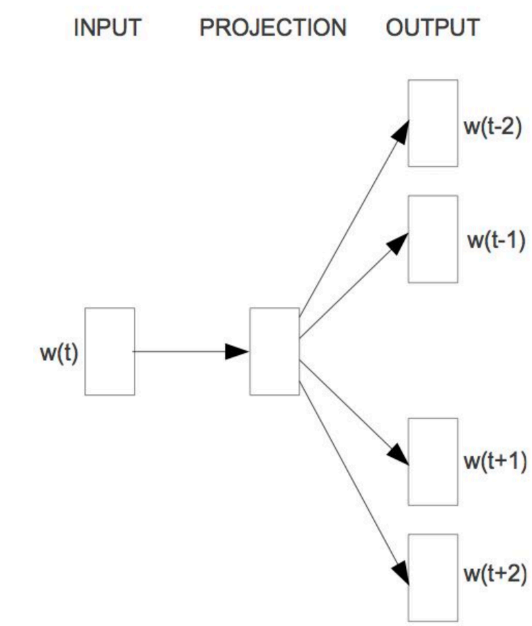
$$W_{\text{the}} = \text{LookUp}(W_{\text{in}}, \text{“the”})) \in \mathbb{R}^d, W_{\text{in}} \in \mathbb{R}^{|V| \times d}$$

$$S = W_{\text{the}} \cdot W_{\text{out}} \in \mathbb{R}^{|V|}$$

$$P(W_c | W_{\text{the}}) = \text{softmax}(S)$$

$$W_c = \{\text{“sat”}, \text{“on”}, \text{“mat”}, \langle \text{end}_{+1} \rangle\}$$





Skip-gram

Skip-gram Objective

- For each word in the corpus

$$J(\Theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} p(w_{t+j} | w_t; \Theta)$$

$$J(\Theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t; \Theta)$$

Maximize the probability of any context window given the current center word

Skip-gram Objective

- For each word in the corpus

$$J(\Theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t;\Theta)$$

$$p(w_{t+j}|w_t) = p(o|c) = \frac{\exp\boxed{(u_o^\top v_c)}}{\sum_{i=1}^{V}\exp(u_i^\top v_c)}$$

dot product (similarity) between outside and center word vectors

Notation simplification:
o = index of outside (context) word
c = index of center word (w_t)
V = vocab size, V can be large 50K - 30M

Skip-gram w/ negative sampling

- V=50K-30M, too large!

$$p(w_{t+j}|w_t) = p(o|c) = \frac{\exp(u_o^\top v_c)}{\boxed{\sum_{i=1}^{V} \exp(u_i^\top v_c)}}$$

- Negative sampling:
  - Treat the center word and a neighboring context word as positive examples.
  - Randomly sample other words in the lexicon to get negative samples.

(banking, regulation) (banking, aardvark)

Skip-gram w/ negative sampling

- Convert the task to binary classification rather than multiclass:

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)} \longrightarrow P(o \mid c) = \frac{1}{1 + \exp(-u_o^T v_c)} = \sigma(u_o^T v_c)$$

- New objective (single context word, k negative samples):

$$\log P(o_+ \mid c) + \sum_{i=1}^{k} \log(1 - P(o_i \mid c))$$

Choosing negative samples

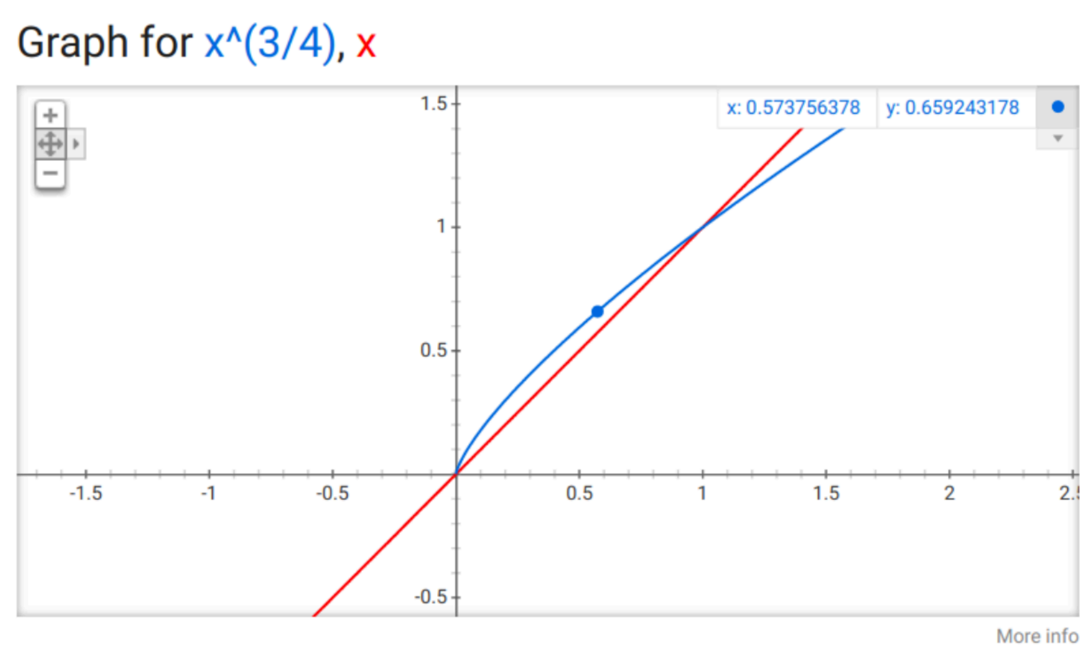
- Pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- $\alpha = 0.75$ works well empirically
- Gives rare words slightly higher probability
  - e.g., P(a) = 0.99, P(b) = 0.01

$$P_\alpha(a) = \frac{0.99^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.97$$

$$P_\alpha(b) = \frac{0.01^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.03$$



Available dense embeddings

- Word2vec (Mikolov et a. 2013)
  - https://code.google.com/archive/p/word2vec/
- GloVe (Pennington et al. 2014)
  - http://nlp.stanford.edu/projects/glove/
- Fasttext (Bojanowsi et al. 2017)
  - http://www.fasttext.cc/

Evaluation

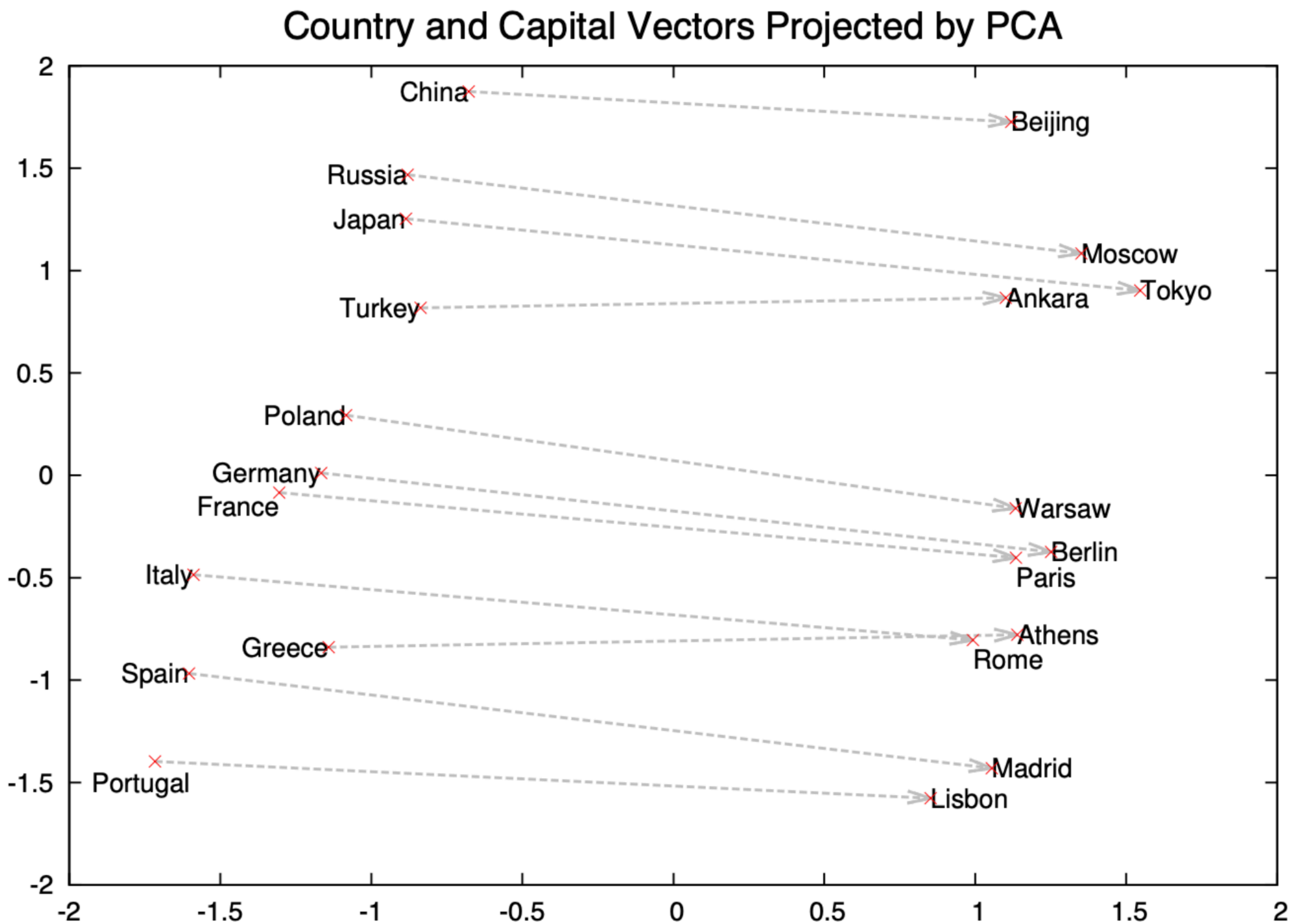
— how well do word vectors capture embedding similarity?

Evaluating word vectors

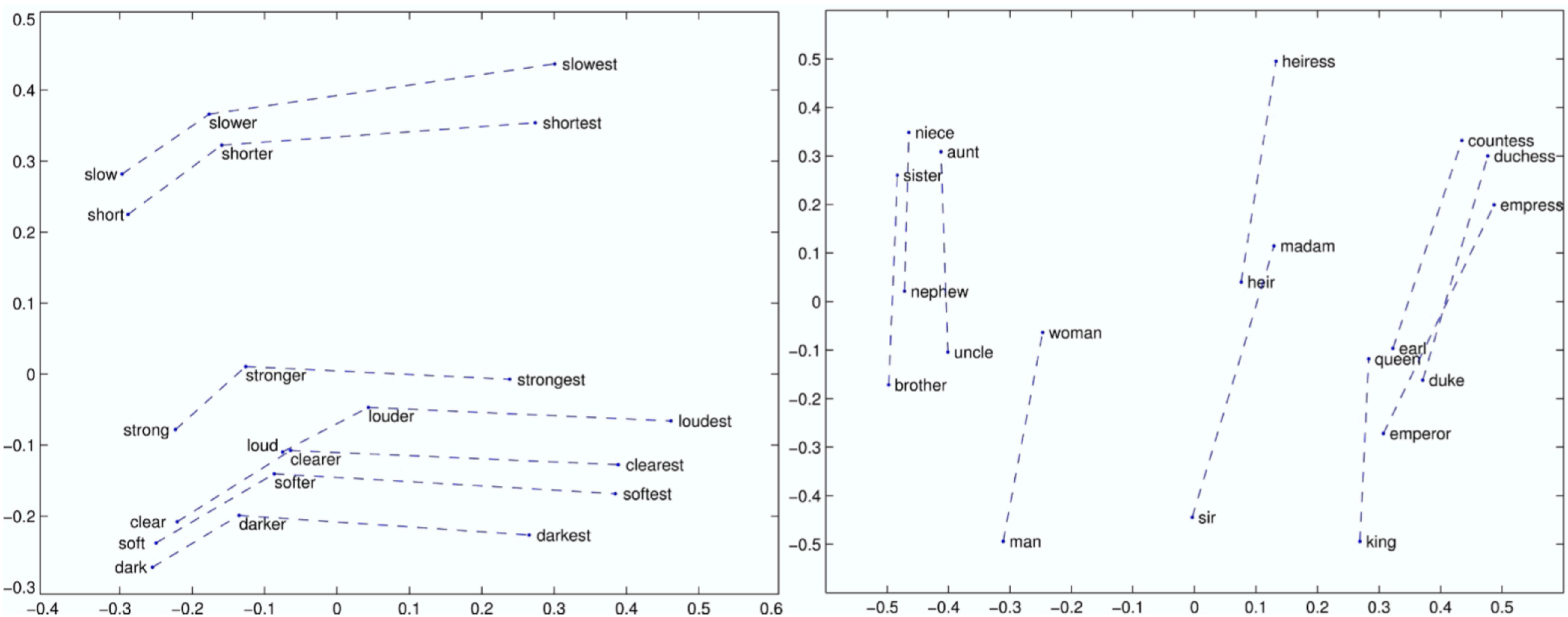
- **Intrinsic evaluation**: test whether the representations align with our intuitions about word meaning.
  - How well does cosine similarity of word embeddings correlate with human judgements?
  - Completing analogies: a:b <-> c: ?

- **Extrinsic evaluation**: test whether the representations are useful for downtream tasks, such as tagging, parsing, QA, …
  - Provide embeddings as input to the same classifier, how well does a model w/ pre-trained embeddings perform?

A:B <-> C:?


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

A:B <-> C:?



slowest
slower
shortest
shorter
slow
short
stronger
strongest
strong
louder
loudest
loud
clearer
clearest
softer
softest
clear
soft
dark
darker
darkest
heiress
niece
aunt
sister
countess
duchess
empress
madam
heir
nephew
woman
uncle
brother
earl
queen
duke
emperor
sir
man
king

Text Classification

Text Classification

- Classify sentences according to various traits
- Topic, sentiment, subjectivity/objectivity, etc.

I hate this movie

positive
neutral
negative

Example: Movie Review

- Predict sentiment from IMDB movie review: {positive, neural, negative}

10/10

positive

So glad to have you back Guiliana!
Love_Life_Laughter 2 September 2018

I watched the ugly breakup of the E! channel circle around Joan Rivers, after her tragic and unexpected demise, with such sadness. I am so glad to have Guiliana back. She has such a unique personna, equally comfortable talking about fashion and cancer recovery, that somehow provides the perfect balance between appropriate gravitas and celebratory girliness. Bravo E!

2/10

negative

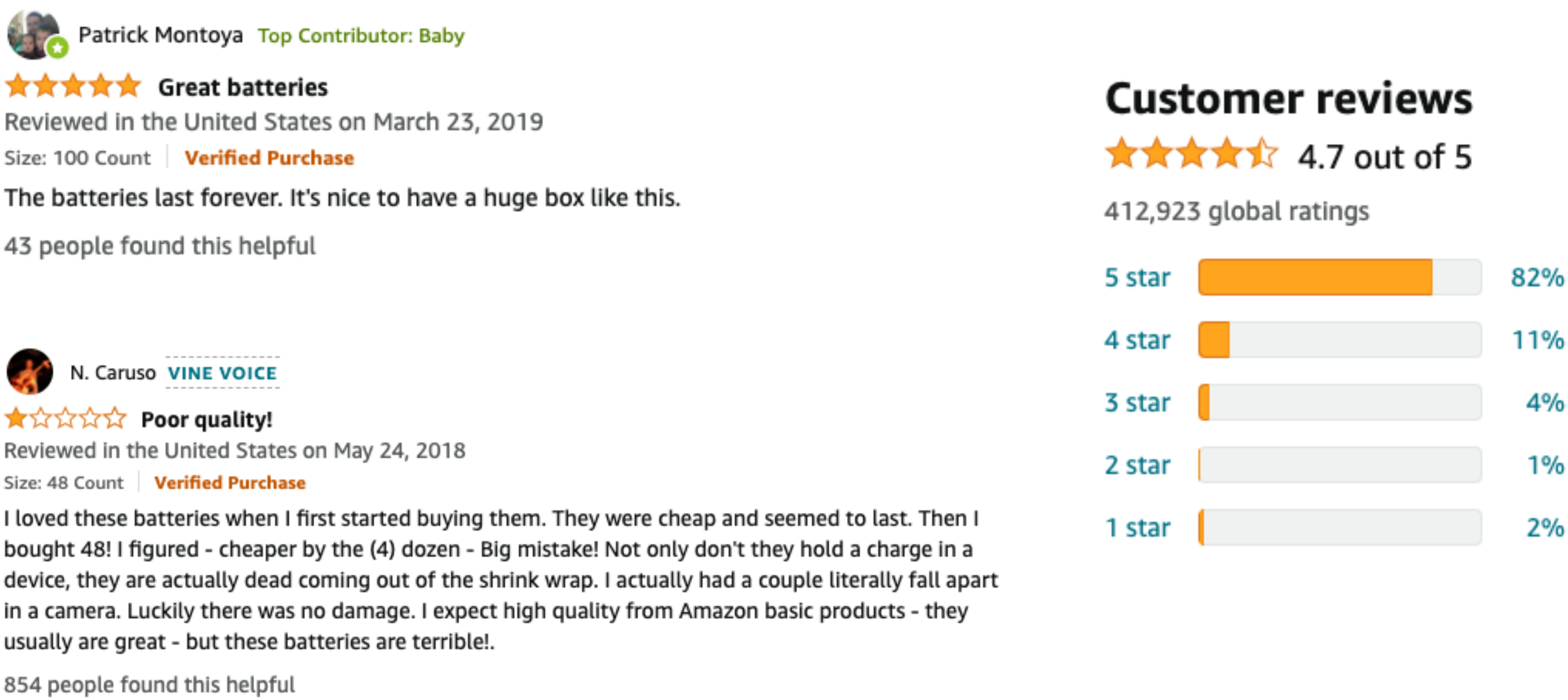
Not funny or entertaining at all
gtamaniak-16300 5 November 2019

Some times nitpicking films and other media might be interesting to watch if the are some clever jokes made by the editor. Here, while some plot holes of certain films are revealed which I wouldn't even think of noticing and that's a good thing, there are some observations that are born from the poor imagination of the narrator and don't have anything to do with the certain media reviewed. Absolutely unfunny moments that don't even make me chuckle. At least the narrator's voice is OK and not like one of those annoying British accents from whatculture.

Code: https://colab.research.google.com/github/bentrevett/pytorch-sentiment-analysis/blob/master/1%20-%20Simple%20Sentiment%20Analysis.ipynb
Dataset: http://ai.stanford.edu/~amaas/data/sentiment/

Example: Customer Rating

- Predict Amazon customer rating: {1, 2, 3, 4, 5}



Dataset: https://s3.amazonaws.com/amazon-reviews-pds/readme.html

Generative and Discriminative Models

Generative vs. Discriminative Models

- **Generative model:** a model that calculates the probability of the input data itself

$$P(X) \qquad\qquad P(X, y)$$

stand-alone   *joint*

- **Discriminative model:** a model that calculates the probability of the output given the input data

$$P(y \mid X)$$

conditional

Application to Text Classification

- **Generative text classification:** Learn a model of the joint P(*X*, *y*), and find

$$\hat{y} = \arg\max_y P(X, y)$$

- **Discriminative text classification:** Learn a model of the conditional P(*y* | *X*), and find

$$\hat{y} = \arg\max_y P(y|X)$$

Discriminative Text Classification

Why Discriminative Classifiers?

- Generative models are somewhat roundabout
  → spend lots of capacity modeling the input
- Discriminative models directly model the probability of the output → what we care about
- However, discriminative models **don't have an easy count-based decomposition**!

BOW Generative:

$$P(X,y) = P(y)\prod_{i=1}^{|X|} P(x_i|y) = \frac{c(y)}{\sum_{\tilde{y}} c(\tilde{y})}\prod_{i=1}^{|X|}\frac{c(x_i,y)}{\sum_{\tilde{x}} c(\tilde{x},y)}$$

BOW Discriminative:

$$P(y|X) = \frac{P(y,X)}{P(X)} \quad ??$$

Sentence space is infinite!

Discriminative Model Training

- Instead, define model that calculates probability directly based on parameters θ

$$P(y|X;\theta)$$

- Define a **loss function** that is lower if the model is better, such as **negative log likelihood** over training data

$$\mathcal{L}_{\text{train}}(\theta) = -\sum_{\langle X,y\rangle \in \mathcal{D}_{\text{train}}} \log P(y|X;\theta)$$

- And **optimize the parameters directly** to minimize loss

$$\hat{\theta} = \underset{\tilde{\theta}}{\operatorname{argmin}}\ \mathcal{L}_{\text{train}}(\tilde{\theta})$$

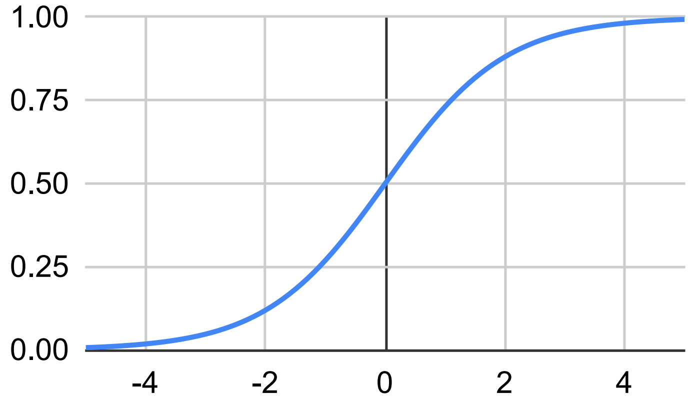
Logistic Regression

- For **binary classification** of positive/negative, first calculate score

$$s_{y|X} = \theta_{y|X} \cdot \underbrace{f(X)}_{\text{Learn a feature extractor}}$$

- Convert into a **probability**, e.g. using *sigmoid* function

$$P(y|X;\theta) = \text{sigmoid}(s_{y|X}) = \frac{1}{1+e^{-s_{y|X}}}$$



- **Learning**: maximize log likelihood of training data

$$\mathcal{L}_{\text{train}}(\theta) = \sum_{\langle X,y\rangle \sim \mathcal{D}_{\text{train}}} \log P(y|X;\theta), \quad \theta^* = \arg\max_{\theta} \mathcal{L}(\theta)$$

Multi-class Classification: Softmax

- Sigmoid can be used for binary decisions
- For multi-class decisions, calculate score for each class and use **softmax**

$$P(y|X;\theta) = \frac{e^{s_{y|X}}}{\sum_{\tilde{y}} e^{s_{\tilde{y}|X}}}$$

$$s = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix} \longrightarrow p = \begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ 0.444 \\ 0.090 \\ \ldots \end{pmatrix}$$

Gradient Descent

- Calculate the **gradient of the loss function** with respect to the parameters

$$\frac{\partial \mathcal{L}_{\text{train}}(\theta)}{\partial \theta}$$

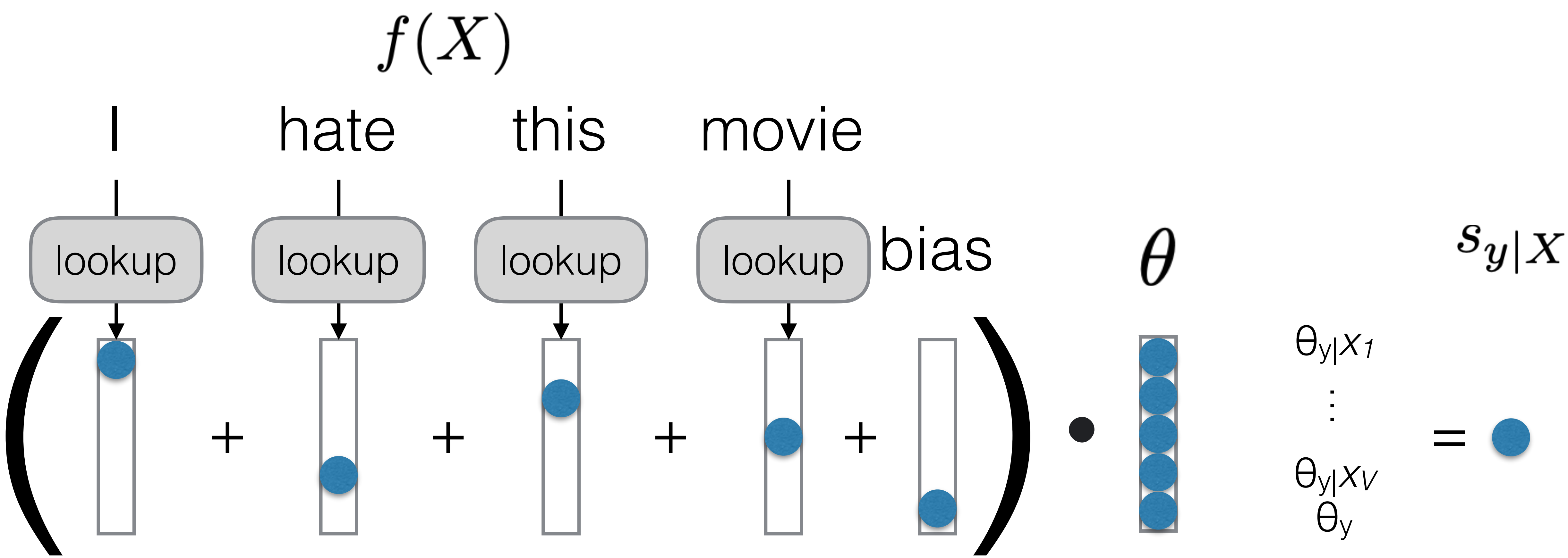
  - How? Use the chain rule - more in later lectures.
- **Update** to move in a direction that decreases the loss

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}_{\text{train}}(\theta)}{\partial \theta}$$

  - α is a **learning rate** dictating speed of movement
  - This is the *first-order* gradient descent
  - Others, e.g. Newton's method, consider *second-order* (curvature) information and converge more quickly

BOW Discriminative Model

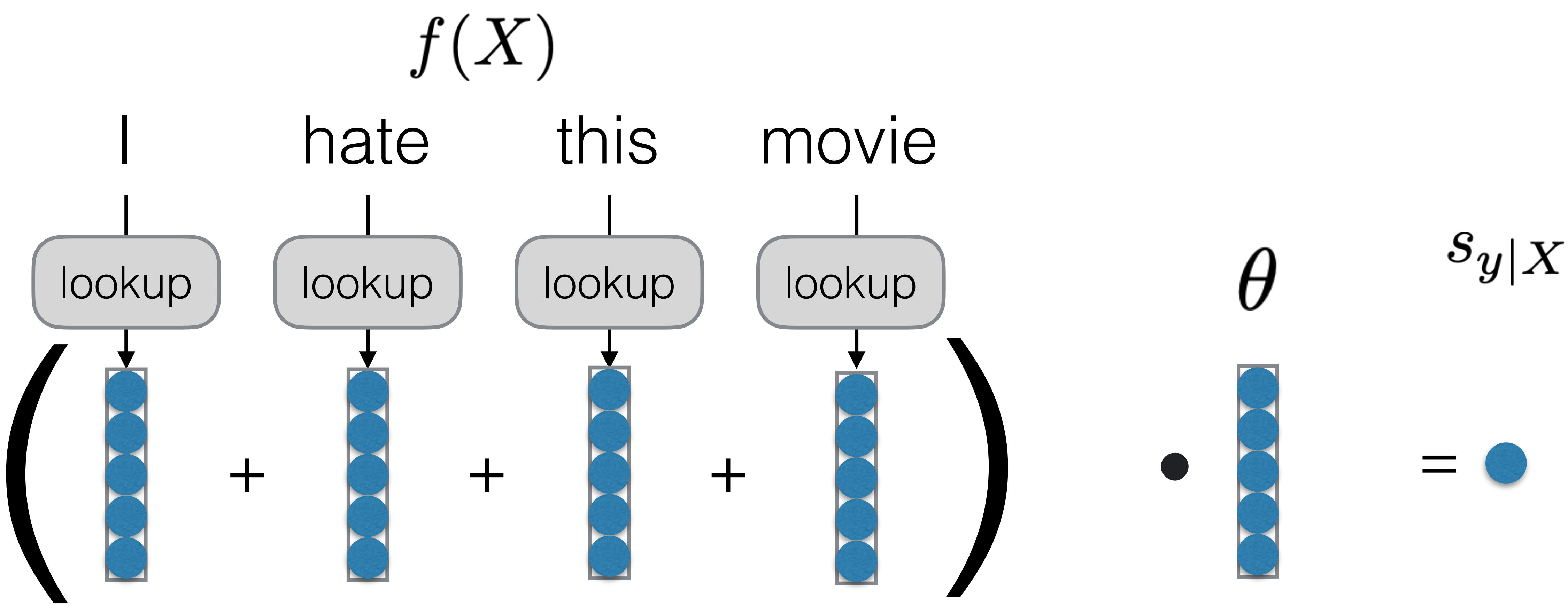
- Use BOW representations for $f(X)$



$$s_{y|X} = \theta_y + \sum_{i=1}^{|X|} \theta_{y|x_i}$$

CBOW Discriminative Model

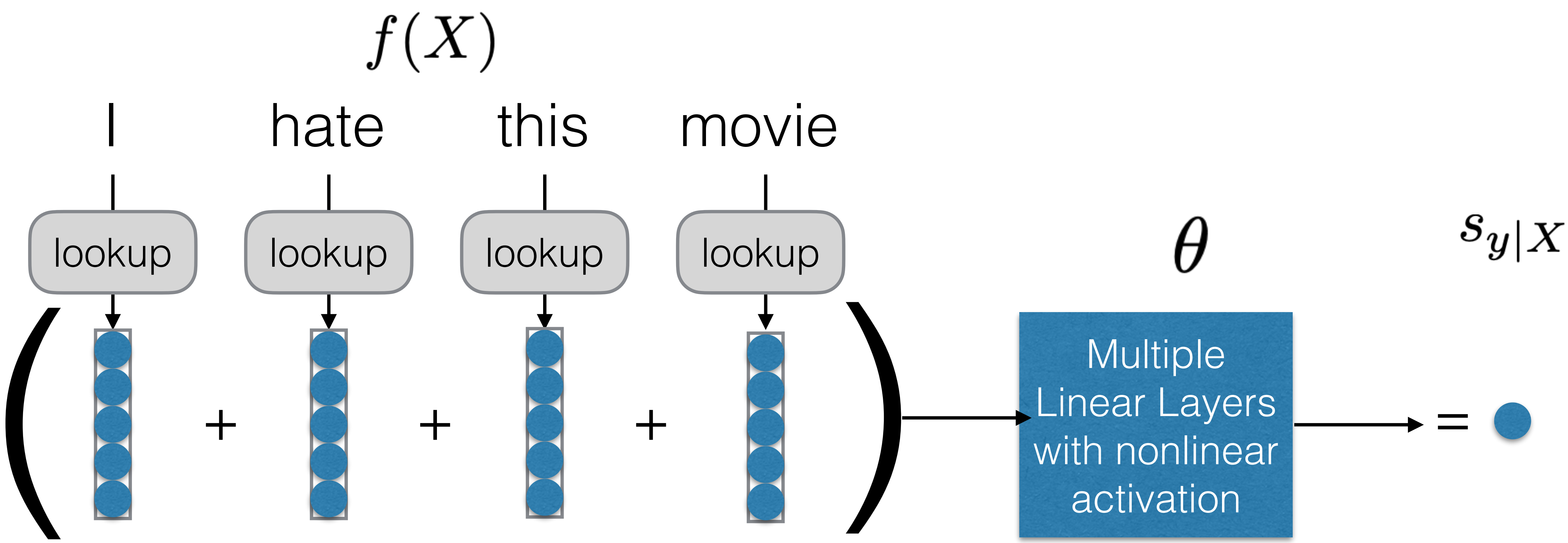
- Use CBOW representations for encoding a sentence $f(X)$



$$s_{y|X} = \theta_y + \sum_{i=1}^{|X|} \theta_{y|x_i}$$

CBOW Discriminative Model

- Use CBOW representations for encoding a sentence $f(X)$



$$s_{y|X} = \theta_y + \sum_{i=1}^{|X|} \theta_{y|x_i}$$

Covert Scores to Probabilities

- Using Softmax function

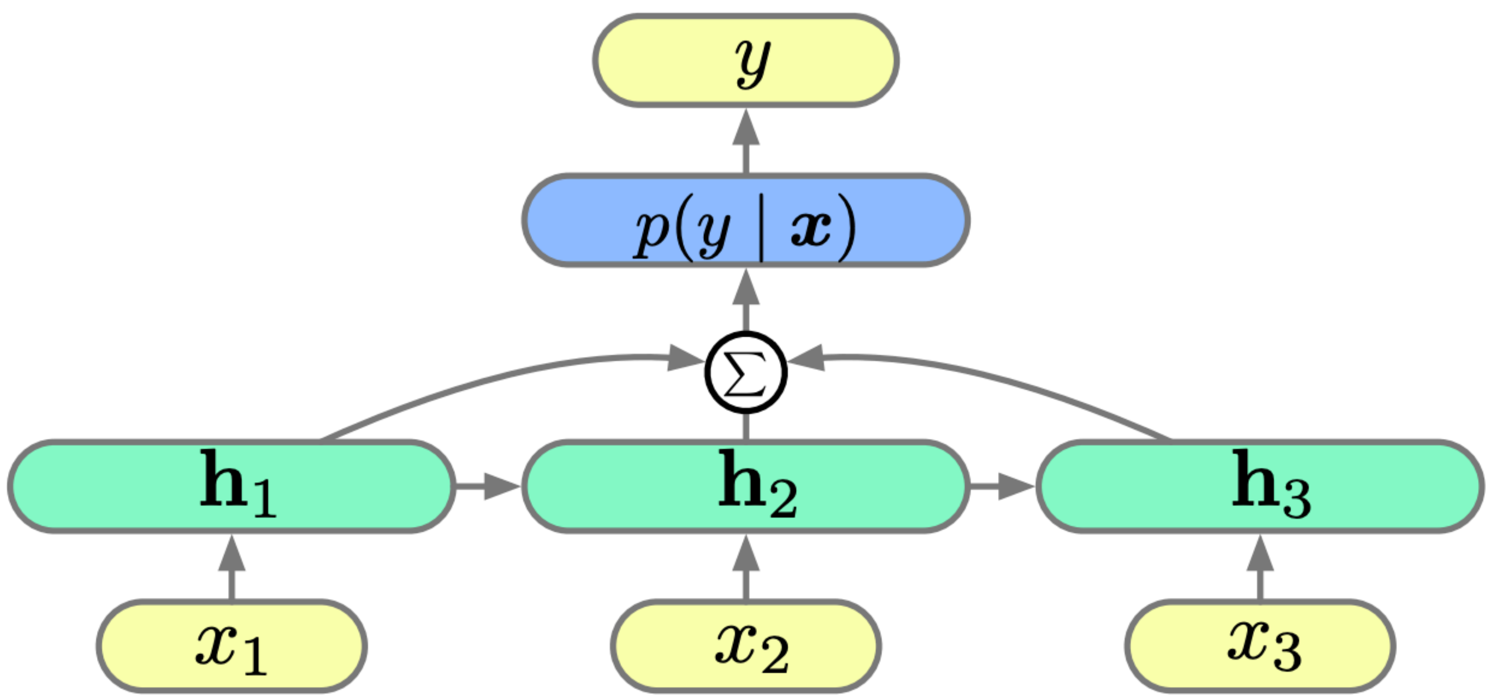
$$P(y|X;\theta) = \mathrm{SoftMax}(s_{y|X}) = \frac{e^{s_{y|X}}}{\sum_{y' \in Y} e^{s_{y'|X}}}$$

- Optimize by gradient descent with an regularization

$$\mathcal{L}_{\mathrm{train}}(\theta) = \sum_{\langle X,y \rangle \sim \mathcal{D}_{\mathrm{train}}} \log P(y|X;\theta) + \lambda \|\theta\|^2$$

Neural Network Discriminative Model

- Use neural network (e.g., LSTM) to learn features $f(X)$



$$s_{y|X} = \mathrm{NN}(X)$$

Any Questions?