

# Experimental and Computational Methods in Linguistic Research

Spring 2025

Instructor: Sanghee Kim

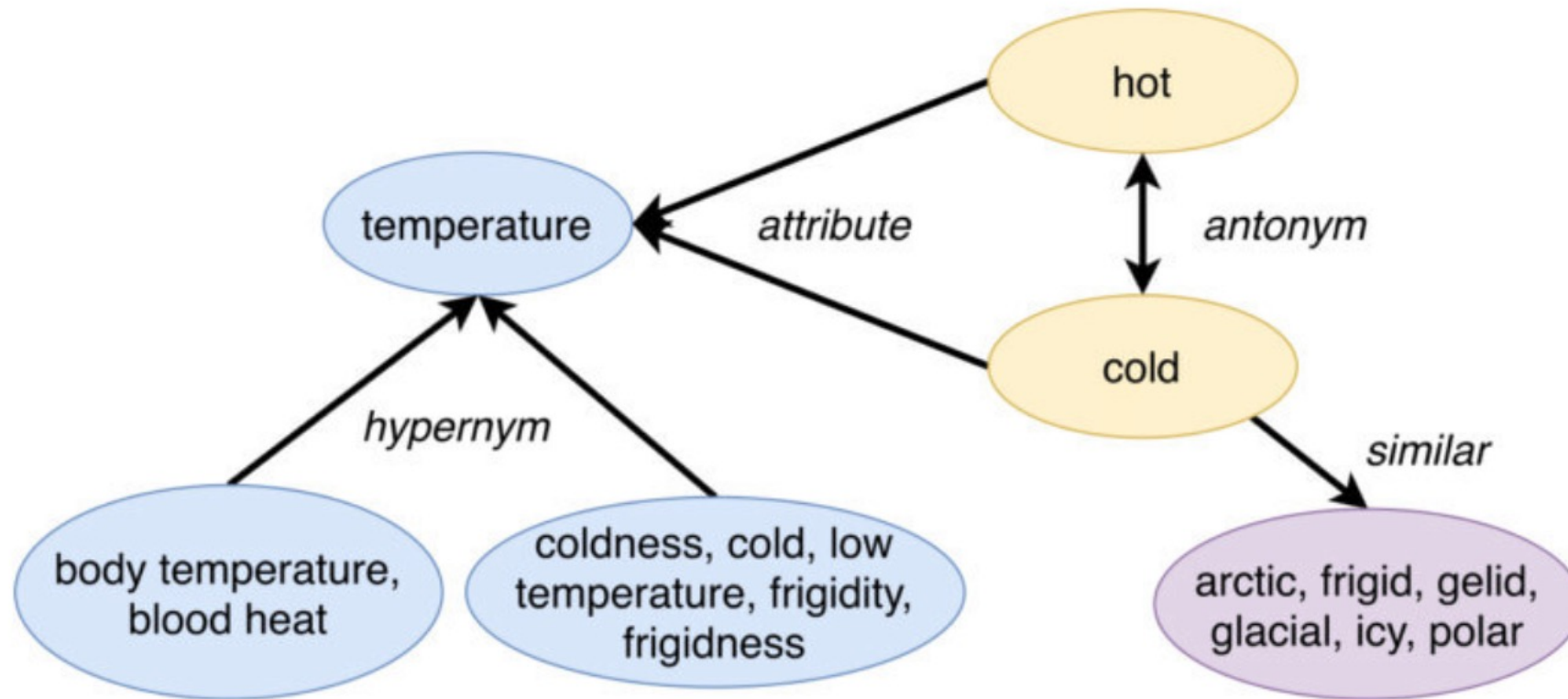
Week 3

Representing word meaning

# Word meaning?

- What is meaning?
- How do we represent word meaning?
- How do we quantify word meaning?

# WordNet (see Lecture slides in Week 2)



# Representing words and vocabulary

- What's an easy way to make machines “understand” words?

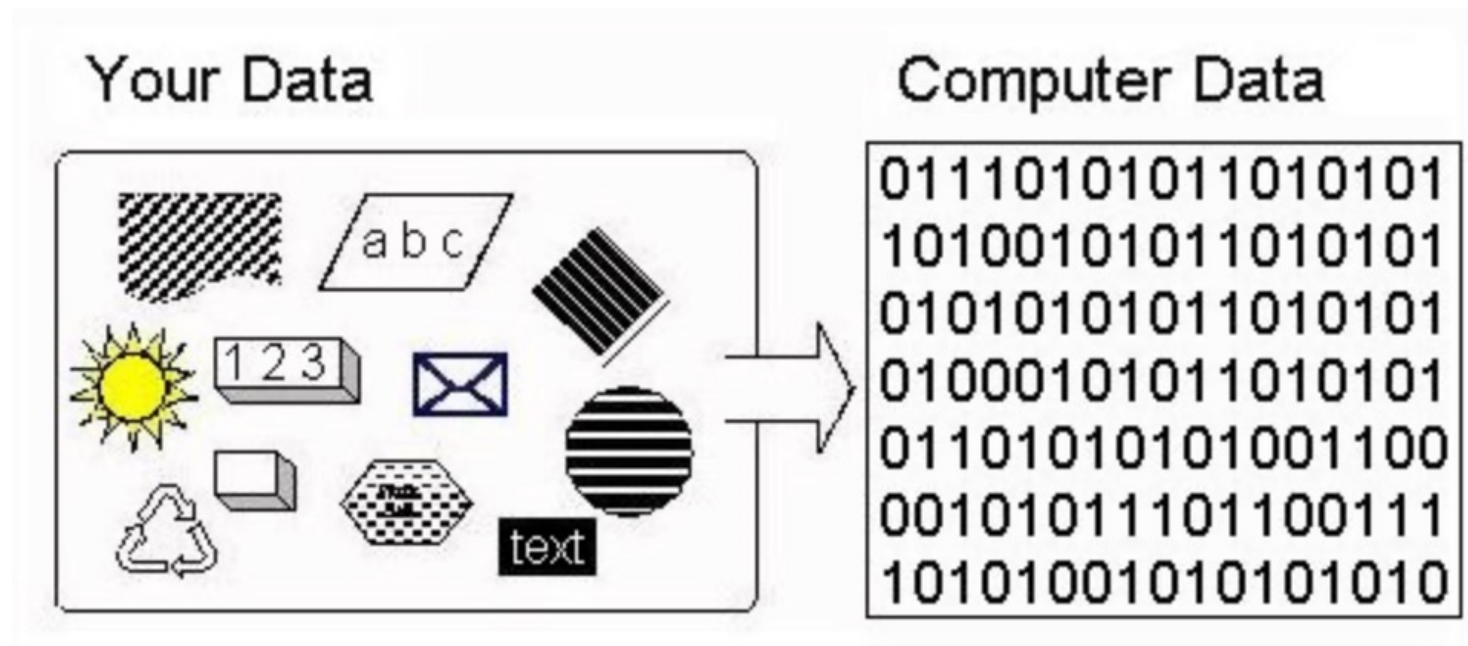
# Representing words and vocabulary

- Zooming out a little – How does communication work in computers?  
What fundamental language do computers use to process and transmit data?



# Representing words and vocabulary

- Zooming out a little – How does communication work in computers? What fundamental language do computers use to process and transmit data?



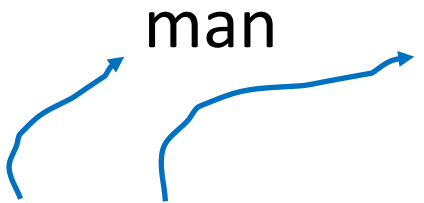
# Representing words and vocabulary

- Perhaps we represent words like 'man', 'woman', 'boy', and 'girl' in a similar way? How can we do this?
- Maybe we can use a vector?



# Representing words and vocabulary

		man	woman
man	=	[1, 0, 0, 0]	
woman	=	[0, 1, 0, 0]	
boy	=	[0, 0, 1, 0]	
girl	=	[0, 0, 0, 1]	



**One-hot encoding**

# Representing words and vocabulary

man = [1, 0, 0, 0, 0, 0, ..., 0]

woman = [0, 1, 0, 0, 0, 0, ..., 0]

boy = [0, 0, 1, 0, 0, 0, ..., 0]

girl = [0, 0, 0, 1, 0, 0, ..., 0]

word  $V$

# One-hot encoding

# One-hot encoding

- Useful and efficient when dealing with data that can be expressed *categorically*
- What are some caveats of one-hot encoding when we apply it to representing word meaning?

# One-hot encoding

.. has issues related to

1. Vocabulary size
2. Frequency information
3. Distribution information
4. Word similarity
5. Contextual information

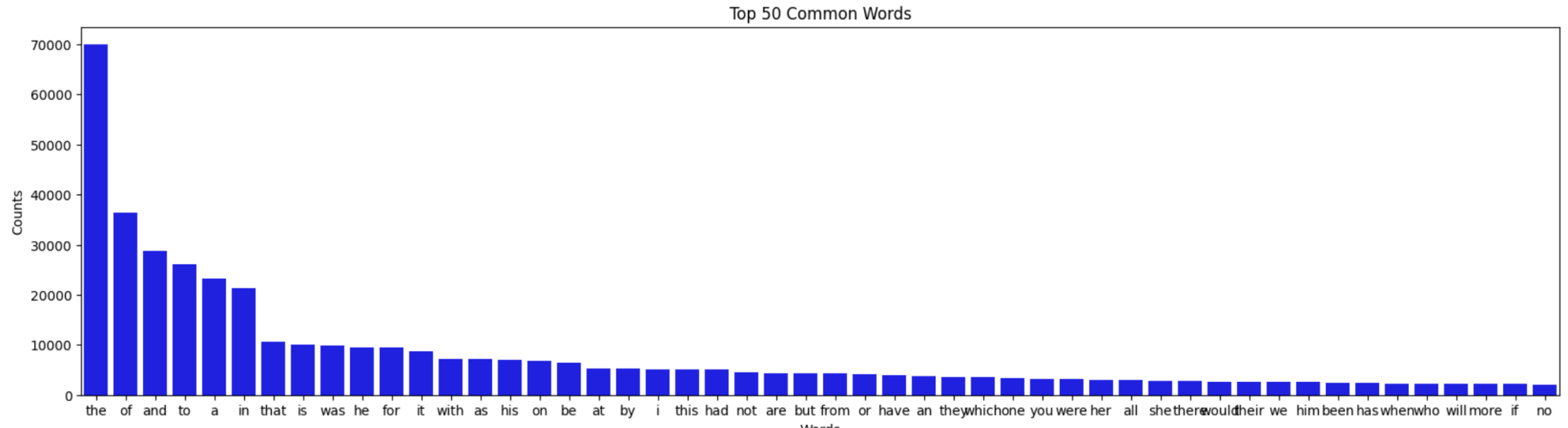
1. **Vocabulary size**
2. Frequency information
3. Distribution information
4. Word similarity
5. Contextual information

# 1. Vocabulary size

- So, if we have 981716 words in the corpus, we will need a vector of size 981716.
- **High dimensional**
- ..this isn't ideal – it can be computationally costly and inefficient.

1. Vocabulary size
- 2. Frequency information**
3. Distribution information
4. Word similarity
5. Contextual information

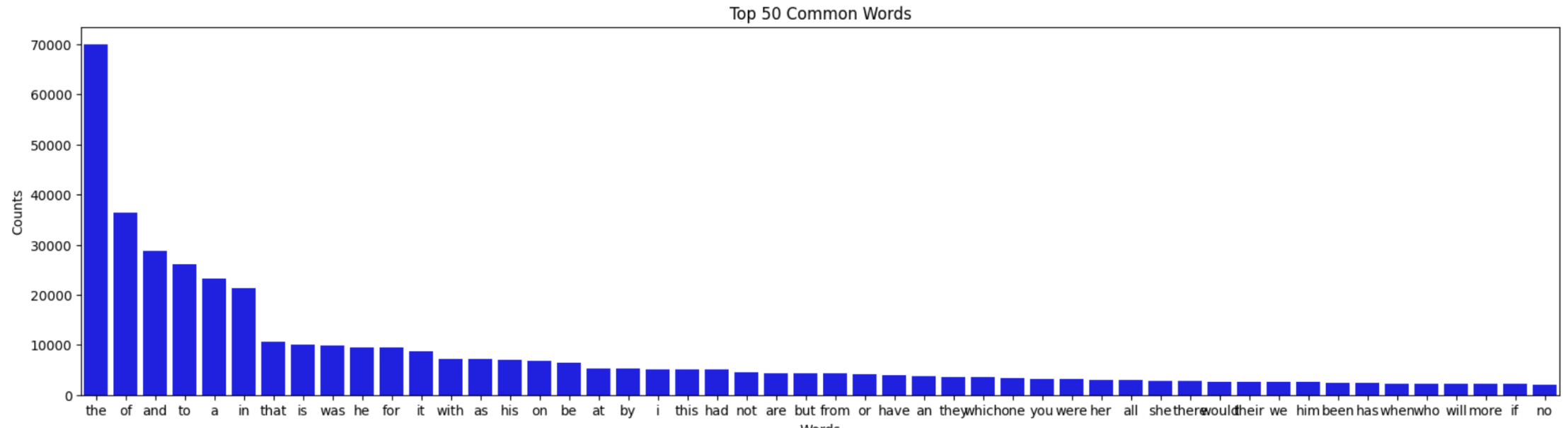
## 2. Frequency information



This information cannot be captured in a one-hot encoding representation.



## 2. Frequency information



If a word  $X$  appears a lot in a text, does that mean  $X$  is an important word?

## 2. Frequency information

- Early approaches focused on analyzing statistical properties of word usage across texts.
- Capturing Relationship between words and documents
  - LSA (Latent Semantic Analysis)
- Weighted word importance within a document relative to their frequency in corpus
  - TF-IDF (Term Frequency-Inverse Document Frequency)
- ...

1. Vocabulary size
2. Frequency information
- 3. Distribution information**
4. Word similarity
5. Contextual information

### 3. Distribution information

- “You shall know a word by the company it keeps” (Firth 1957)
- Intuition: if two words occur in very similar distribution and context, they have similar meanings.

Left context	Word	Right context
friends are pretty annoying. Their		is cute though. (END)
cute when it's a small		Also people who encourage their
dog meat industry. You American		lovers would be heart sickened
loneliness. Usually, a good cute		pic can cheer me up
on with me the cute		pictures. I almost never browse
but if you've got a		that's an asshole I don't
elected in a South American		is testament to that. It
and talk about the American		because I have trouble imagining
kid, they had a family		on what toy got to

Can you predict the target word given surrounding words? How?

Table 4.6: Collocation contexts for *dog*, *cat*, and *election* in AskReddit.

Left context	Word	Right context
friends are pretty annoying. Their	<i>dog</i>	is cute though. (END)
cute when it's a small	<i>dog.</i>	Also people who encourage their
dog meat industry. You American	<i>dog</i>	lovers would be heart sickened
loneliness. Usually, a good cute	<i>cat</i>	pic can cheer me up
on with me the cute	<i>cat</i>	pictures. I almost never browse
but if you've got a	<i>cat</i>	that's an asshole I don't
elected in a South American	<i>election</i>	is testament to that. It
and talk about the American	<i>election</i>	because I have trouble imagining
kid, they had a family	<i>election</i>	on what toy got to

Can surrounding words be predicted by the target word?

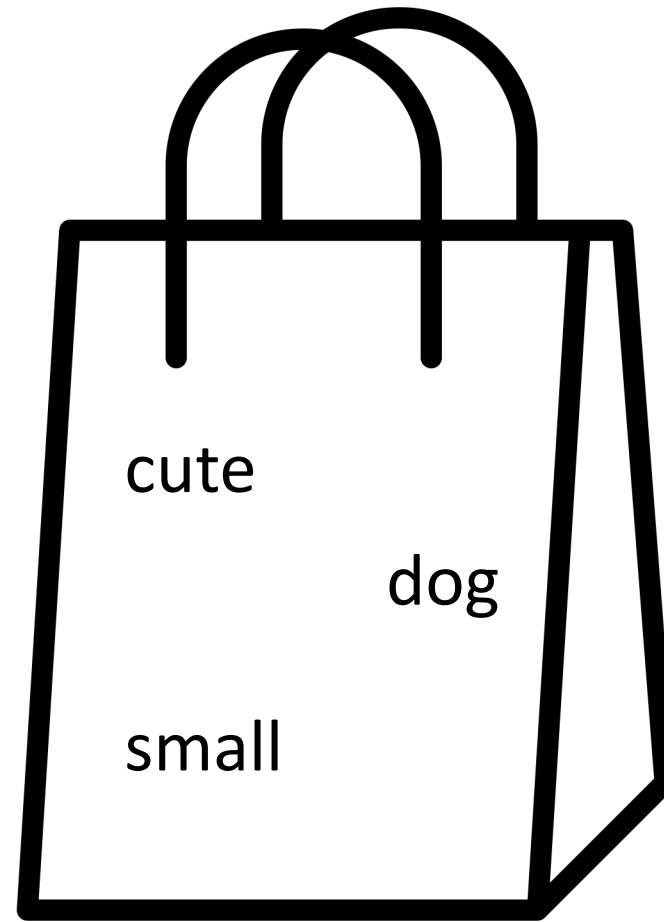
# Co-occurrence information

- Capturing word meaning through **co-occurrence information**
- (small, dog), (cute, cat), (America, n election), (cat, pictures) ...
- There are multiple ways to capture co-occurrence information
  - E.g., Bag of Words (BoW), N-grams

# Co-occurrence information

- **Bag of Words (BoW)**

cute when it's a small *dog.* Also people who encourage their



Word order not captured.



# Co-occurrence information

- **N-gram models**

cute when it's a small *dog*. Also people who encourage their

- How likely is it to see *dog* given prior words?

# N-gram models

- **N-gram models** are great in reflecting co-occurrence information, especially modeling what words appear consecutively.
  - E.g., Bigram model ( $N = 2$ ), Trigram model ( $N = 3$ )

I like my coffee with cream and \_\_\_\_.

I like my coffee with cream and sugar.

I like my coffee with cream and socks.

# Using probabilistic information

- We make predictions!
- Predictions based on lots of different information, e.g., how likely is it to see word X in the given context?

# Bigram model

- Calculating bigram (when N=2) probability

$$\text{Bigram probability: } P(w_n | w_{n-1}) = \frac{\text{count}(w_{n-1}, w_n)}{\text{count}(w_{n-1})}$$

$$\text{Example: } P(\text{sugar} | \text{and}) = \frac{\text{count}(\text{and}, \text{sugar})}{\text{count}(\text{and})}$$

# Bigram model

- Calculating bigram (when N=2) probability

I had coffee with cream and sugar.

$$\text{Example: } P(\text{sugar} | \text{and}) = \frac{\text{count}(\text{and}, \text{sugar})}{\text{count}(\text{and})}$$



# Bigram model

- When  $N=2$ ,

`<s> I like my coffee with cream and sugar </s>`

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

`<s> I like my coffee with cream and sugar </s>`

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>



# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- When  $N=2$ , we will get  $P(w_n | w_{n-1})$

<s> I like my coffee with cream and sugar </s>

# Bigram model

- N-gram (e.g., bigram) assumption:

$P(<s> \text{ I want coffee with cream and sugar } </s>) \approx$

$P(\text{I} | <s>) P(\text{want} | \text{I}) P(\text{coffee} | \text{want}) P(\text{with} | \text{coffee})$   
 $P(\text{cream} | \text{with}) P(\text{and} | \text{cream}) P(\text{sugar} | \text{and}) P(</s> | \text{sugar})$

# N-gram models

- In an N-gram model, the probability of a full sentence is obtained by breaking it down into strings of length  $n$  and then multiplying the probabilities of each  $n$ -gram.

# N-gram models

- How can we express the probability of a word sequence?
- $P(w_1, w_2, w_3, \dots, w_n)$ .
- We won't go into detail, but there is a way to express the conditional probabilities using what is called the **chain rule of probability**:

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \dots P(w_n|w_{1:n-1}) \\ &= \prod_{k=1}^n P(w_k|w_{1:k-1}) \end{aligned}$$

# N-gram models

- It's difficult to get accurate probabilities for long strings
- How do we address this issue?
- Based on a “Markov assumption,” we will assume that we can approximate long context with last few words.
  - Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past.

# Agenda

- Assignment demonstration by Eric
- Wrap-up of n-gram
- Word similarity and embeddings
- Demonstration using vector representation (genism)
  - Measuring similarity scores
  - Getting semantically (un)related words
  - Bonus: Analogy

# N-gram models

- In an N-gram model, the probability of a full sentence is obtained by breaking it down into strings of length  $n$  and then multiplying the probabilities of each  $n$ -gram.



# Bigram model: A toy example

- Mini-corpus of three sentences:

<s> I want coffee with cream and sugar </s>

<s> I want tea </s>

<s> They want coffee </s>

“They want {coffee/tea}.” Which is more likely?

# Bigram model: A toy example

- A mini-corpus of three sentence:

<s> I want coffee with cream and sugar </s>

<s> I want tea </s>

<s> They want coffee </s>

$P(<s>\text{They want coffee}</s>)$

$= P(\text{they} | <s>) P(\text{want} | \text{they}) P(\text{coffee} | \text{want}) P(</s> | \text{coffee})$

$= \frac{\text{count}(<s>, \text{they})}{\text{count}(<s>)} \times \frac{\text{count}(\text{they}, \text{want})}{\text{count}(\text{they})} \times \frac{\text{count}(\text{want}, \text{coffee})}{\text{count}(\text{want})} \times \frac{\text{count}(\text{coffee}, </s>)}{\text{count}(\text{coffee})}$

$= \frac{1}{3} \times \frac{1}{1} \times \frac{2}{3} \times \frac{1}{2}$

$= 0.11111\dots$

# Bigram model: A toy example

- A mini-corpus of three sentence:

<s> I want coffee with cream and sugar </s>

<s> I want tea </s>

<s> They want coffee </s>

$P(<s>\text{They want tea}</s>)$

$= P(\text{they} | <s>) P(\text{want} | \text{they}) P(\text{tea} | \text{want}) P(</s> | \text{tea})$

$= \frac{\text{count}(<s>, \text{they})}{\text{count}(<s>)} \times \frac{\text{count}(\text{they}, \text{want})}{\text{count}(\text{they})} \times \frac{\text{count}(\text{want}, \text{tea})}{\text{count}(\text{want})} \times \frac{\text{count}(\text{tea}, </s>)}{\text{count}(\text{tea})}$

$= \frac{1}{3} \times \frac{1}{1} \times \frac{1}{3} \times \frac{1}{1}$

$= 0.11111\dots$

# Bigram model: A toy example

- A mini-corpus of three sentence:

<s> I want coffee with cream and sugar </s>

<s> I want tea </s>

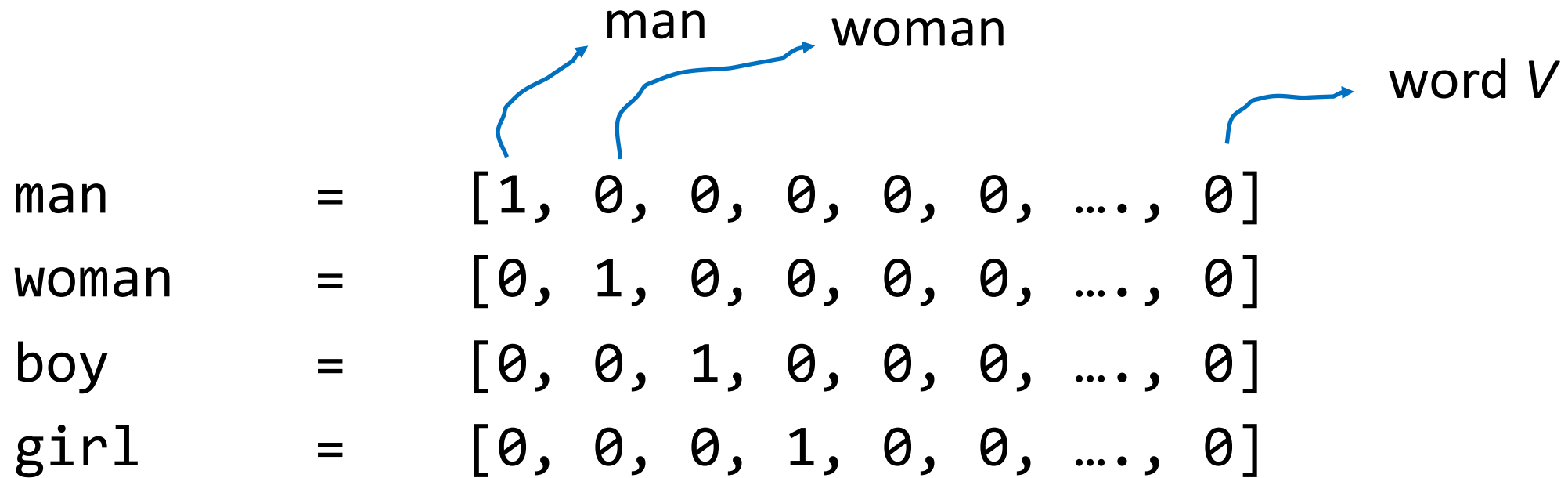
<s> They want coffee </s>

# N-gram models

- They capture co-occurrence information and some probabilistic aspects of language!

1. Vocabulary size
2. Frequency information
3. Distribution information
- 4. Word similarity**
5. Contextual information

## 4. Word similarity



How do we represent that 'man' and 'woman' are similar in age;  
'woman' and 'girl' are similar in gender?

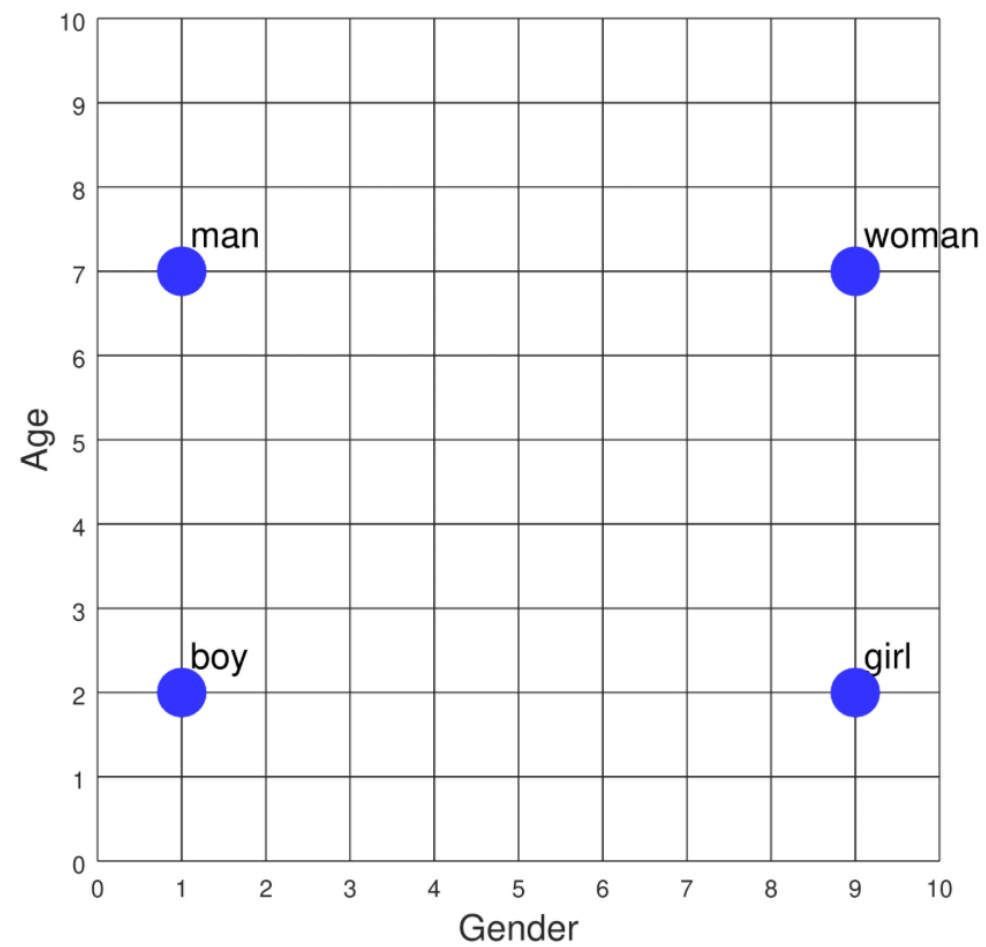
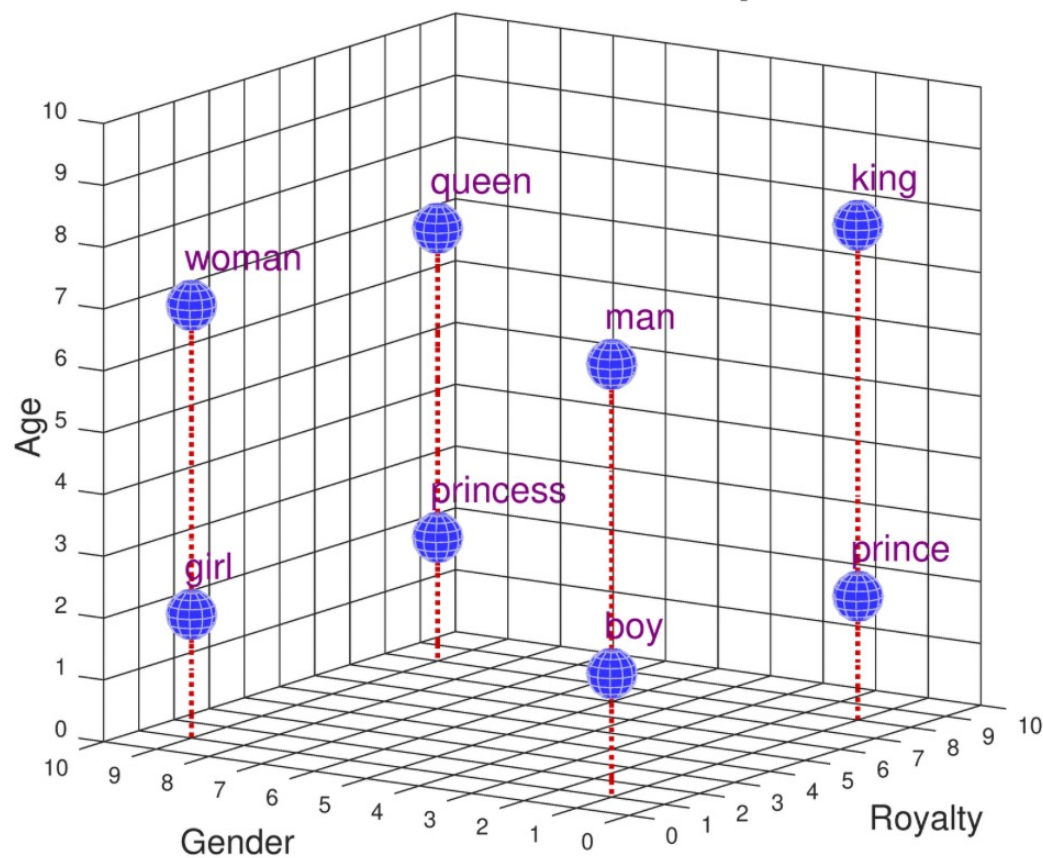


Image: <https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html>



## 3D Semantic Feature Space



Word Coordinates			
	Gender	Age	Royalty
man	[ 1,	7,	1 ]
woman	[ 9,	7,	1 ]
boy	[ 1,	2,	1 ]
girl	[ 9,	2,	1 ]
king	[ 1,	8,	8 ]
queen	[ 9,	7,	8 ]
prince	[ 1,	2,	8 ]
princess	[ 9,	2,	8 ]

Image: <https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html>

- But there's limitation in creating an exhaustive list of all semantic features for the entire vocabulary.

# Interim takeaways

1. Word meaning can be expressed as **vectors**.
  2. Word meaning can be expressed using **probabilities**.
- Addressing the aforementioned issue, we will combine these two.

# Language models

1. How are words represented?
2. What's the architecture?
3. What's the goal?

# Language models

1. How are words represented?

**In vectors**

2. What's the architecture?

**Neural network-based models**

3. What's the goal?

**Predicting/generating word(s)**

# Language models

- **Language models:** (basically) probabilistic distributions of words given a lot of data about the language – especially most current neural network-based models

## A Neural Probabilistic Language Model

**Yoshua Bengio**

**Réjean Ducharme**

**Pascal Vincent**

**Christian Jauvin**

*Département d'Informatique et Recherche Opérationnelle*

*Centre de Recherche Mathématiques*

*Université de Montréal, Montréal, Québec, Canada*

BENGIOY@IRO.UMONTREAL.CA

DUCHARME@IRO.UMONTREAL.CA

VINCENTP@IRO.UMONTREAL.CA

JAUVINC@IRO.UMONTREAL.CA

**Editors:** Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

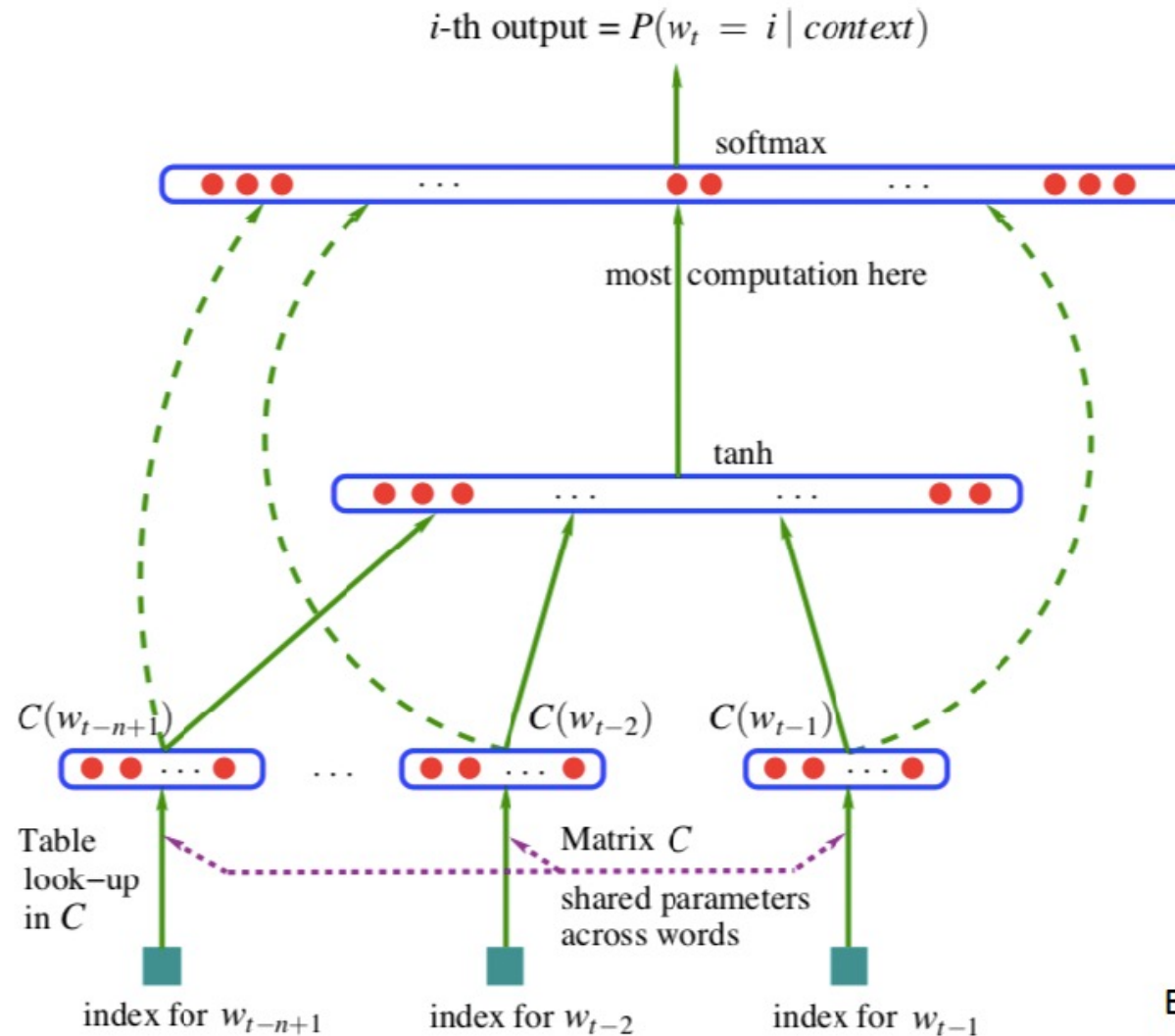
### Abstract

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n-gram models, and that the proposed approach allows to take advantage of longer contexts.

**Keywords:** Statistical language modeling, artificial neural networks, distributed representation, curse of dimensionality

Bengio et al.'s (2003):  
foundation of current  
language models – neural  
(network) probabilistic  
language model.

# Neural network language models





# Neural network language models

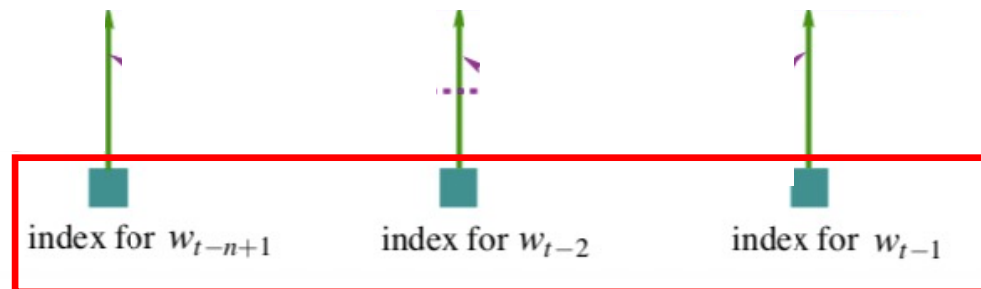
$$i\text{-th output} = P(w_t = i \mid \text{context})$$



# Neural network language models

$$i\text{-th output} = P(w_t = i \mid \text{context})$$

**Words**

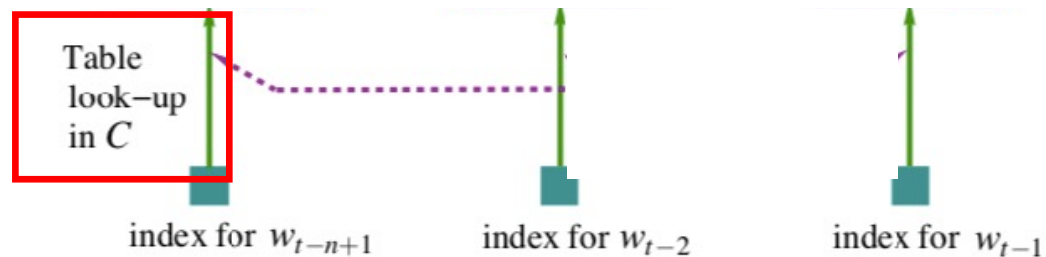


Bengio et al 2003

# Neural network language models

$$i\text{-th output} = P(w_t = i \mid \text{context})$$

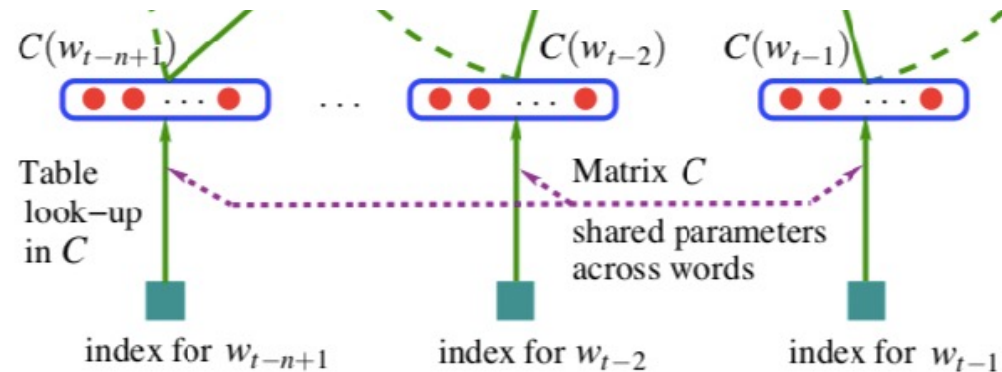
**Vocabulary**



Bengio et al 2003

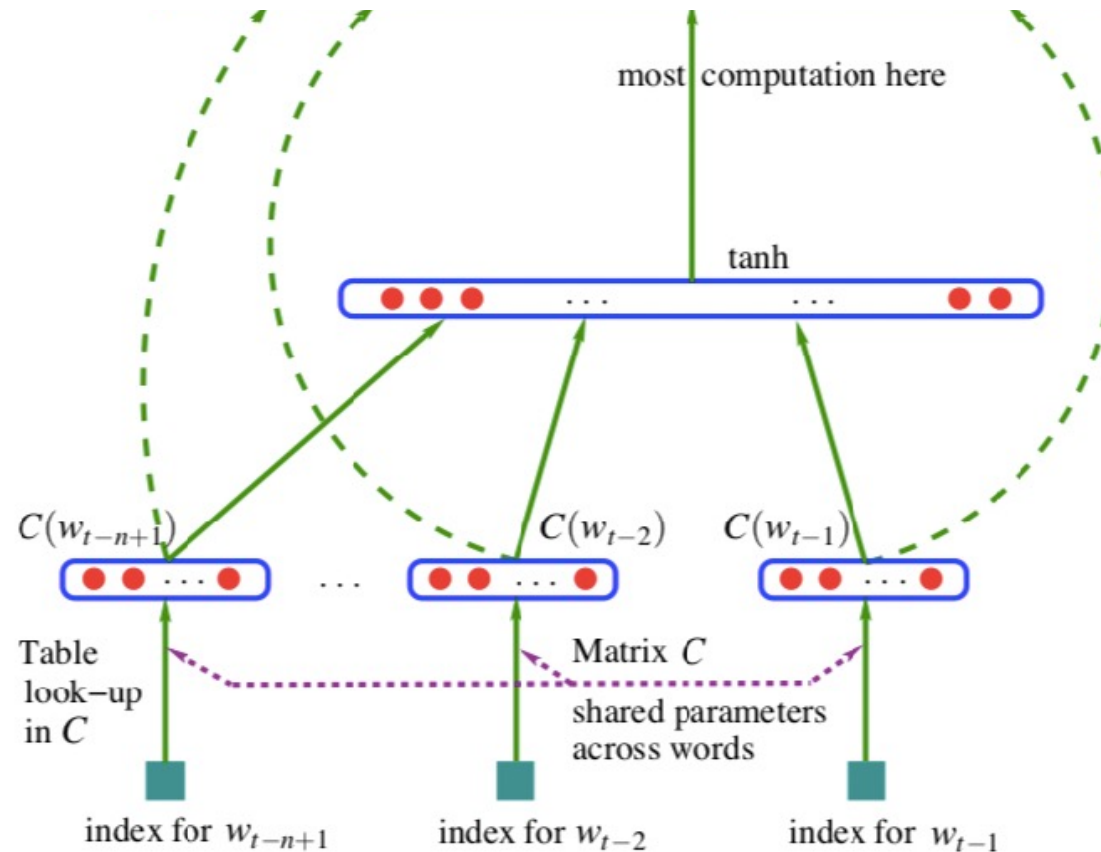
# Neural network language models

$$i\text{-th output} = P(w_t = i \mid \text{context})$$



Bengio et al 2003

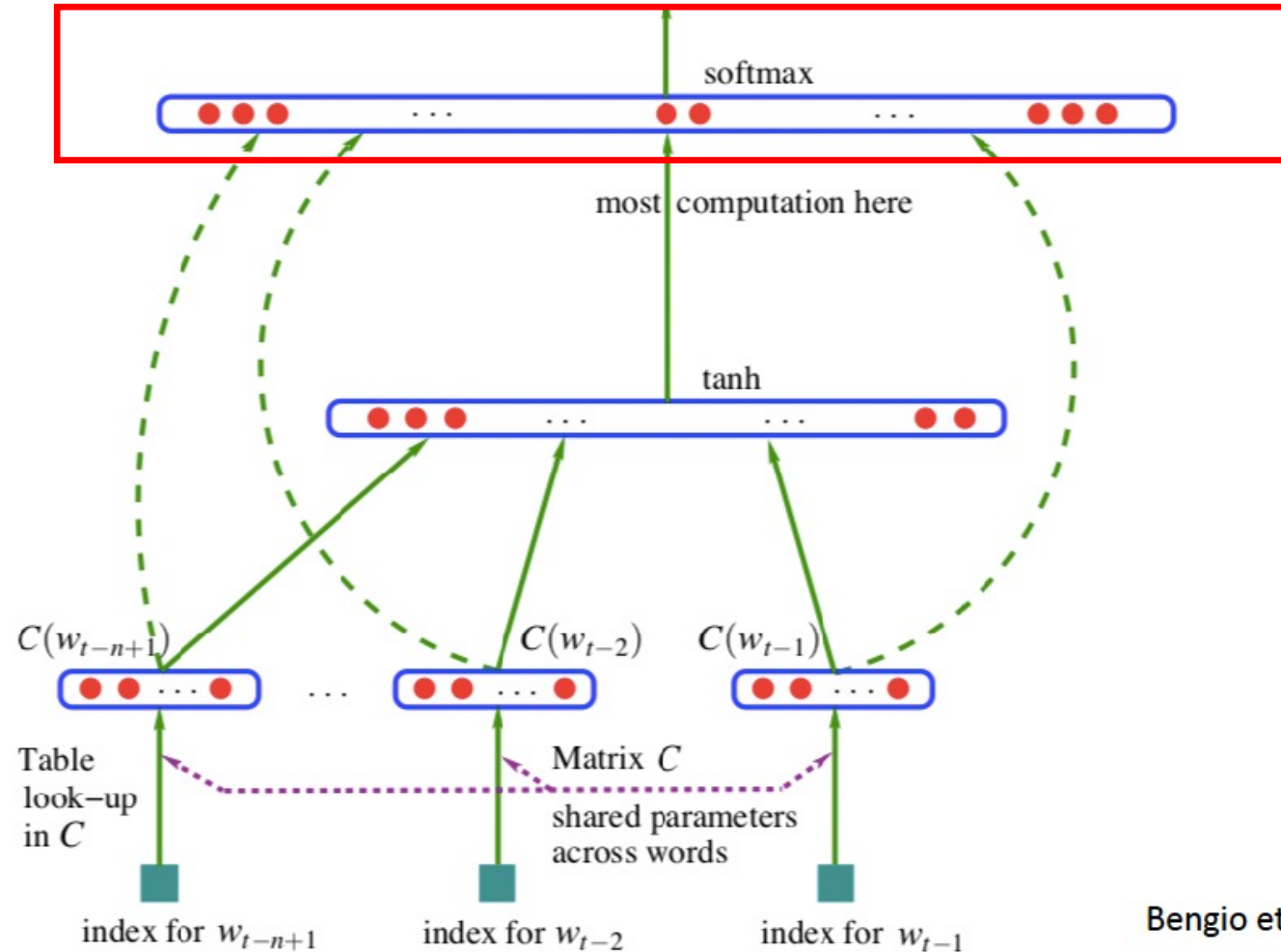
# Neural network language models



Bengio et al 2003

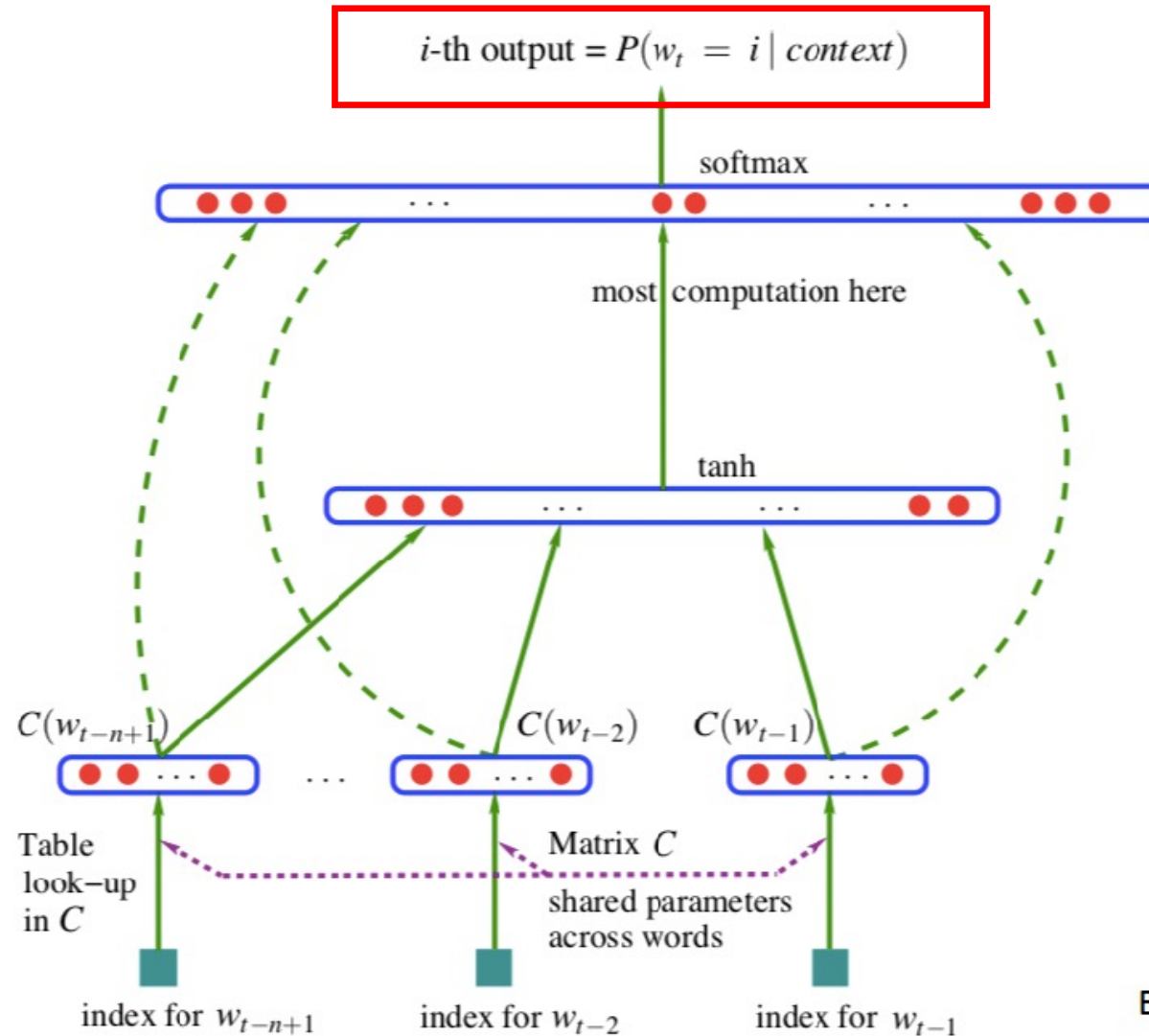
# Neural network language models

**Probability**

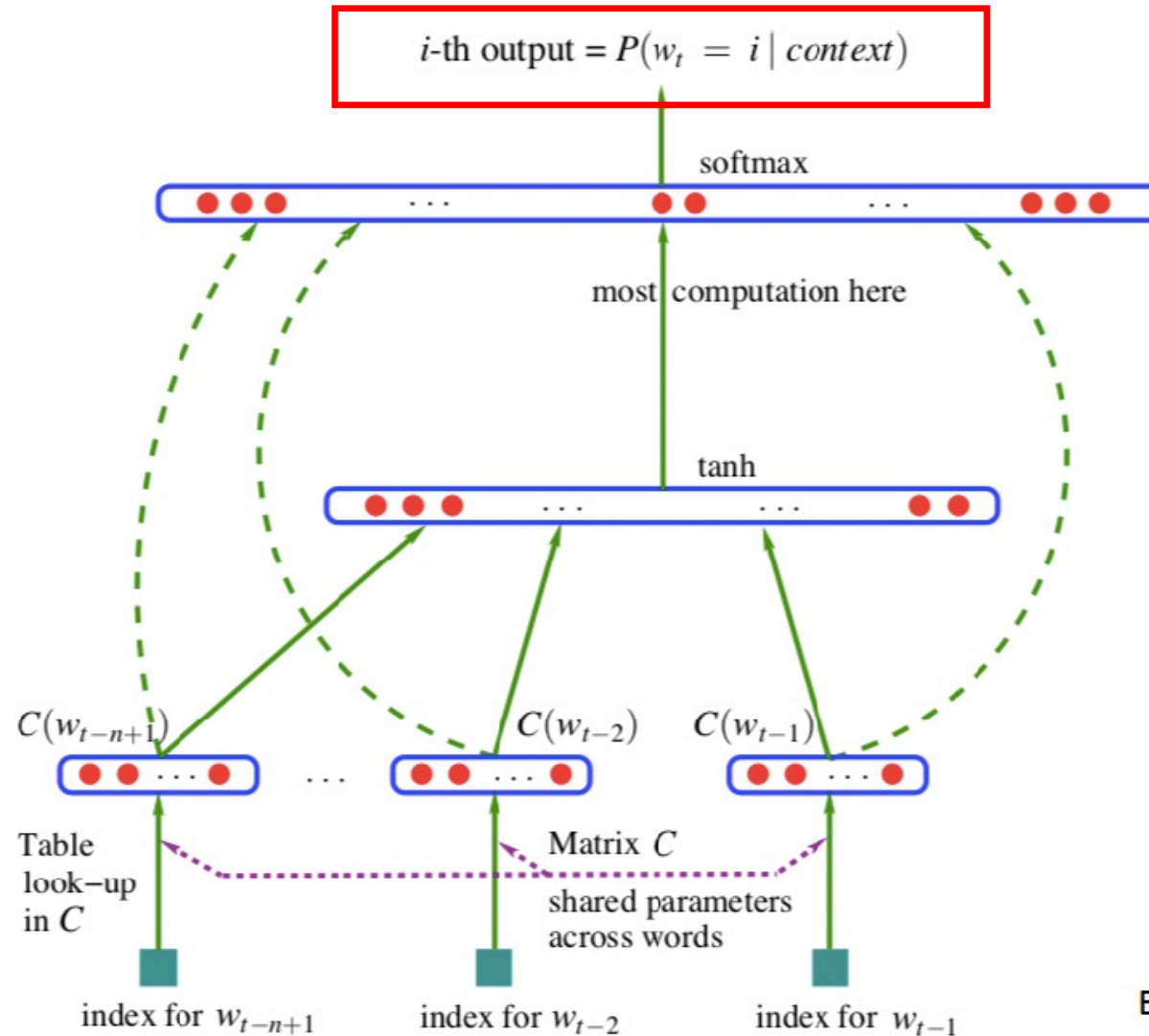


Bengio et al 2003

# Neural network language models



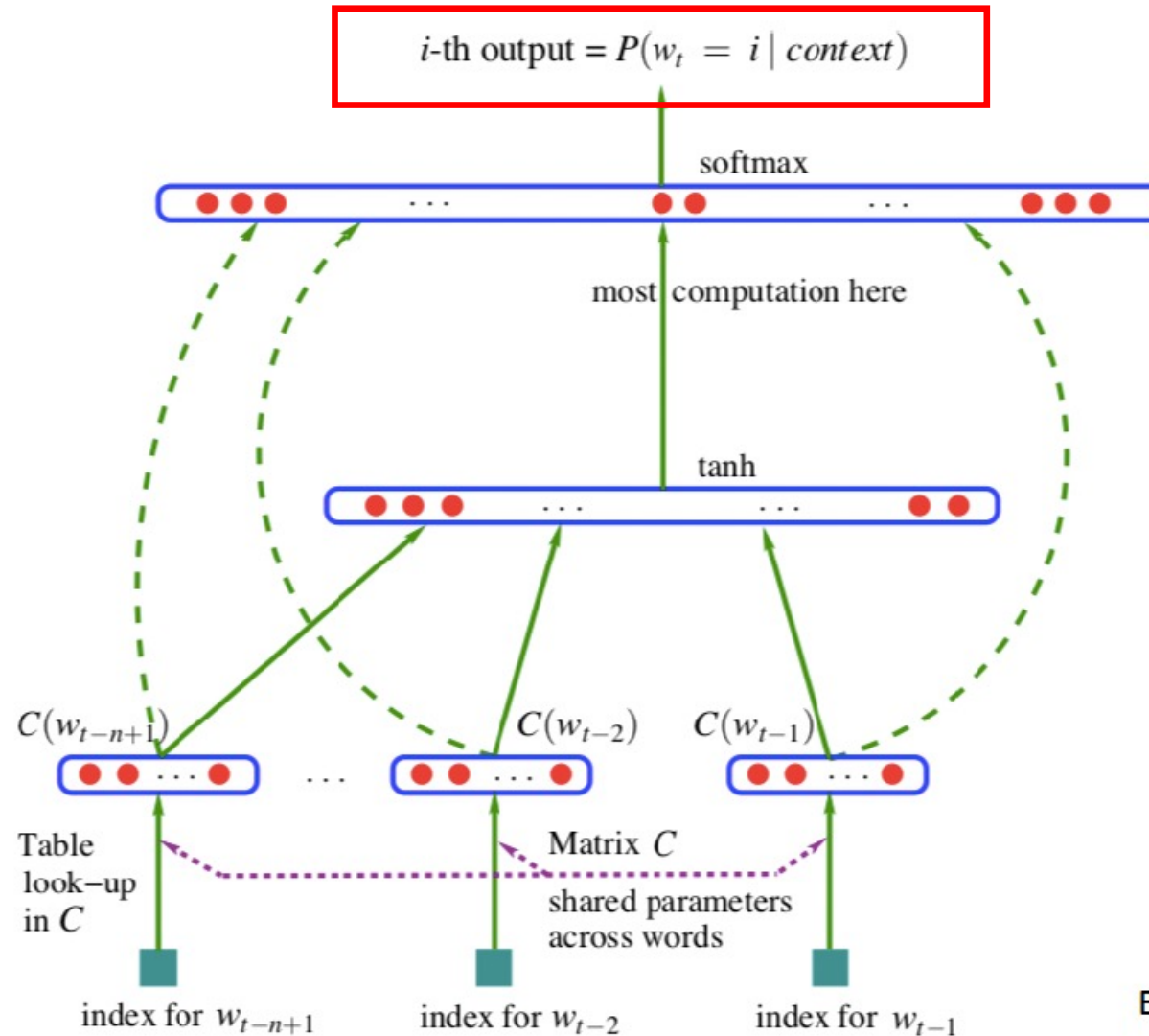
# Neural network language models



Compare the “correct” (based on what we already know given the corpus data we have) and the “guessed” word

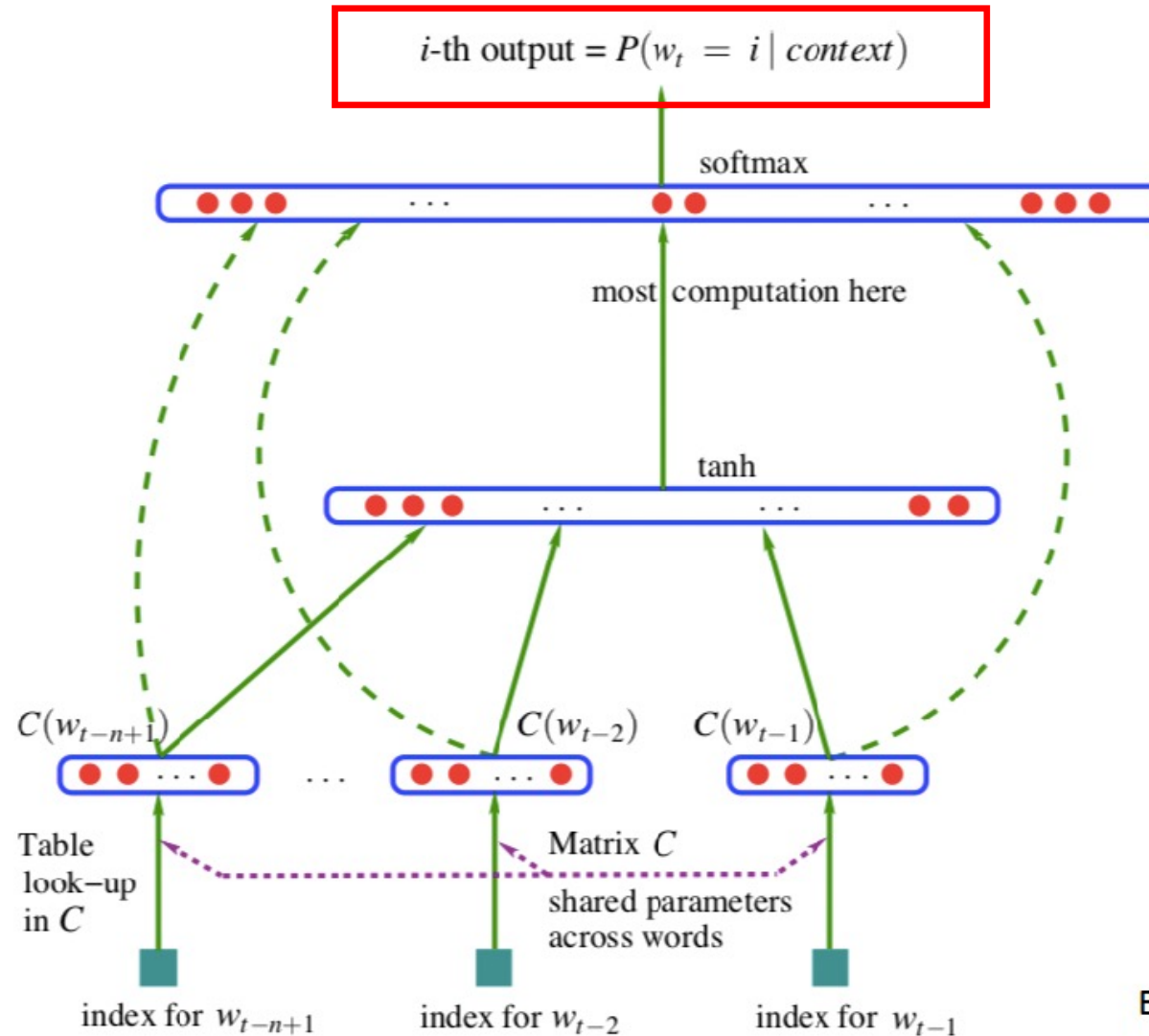


# Neural network language models



Update the model by revising some *knobs* in the model architecture.

# Neural network language models



Repeat the update process until satisfied.

# Word2Vec (Mikolov et al. 2013)

---

## 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

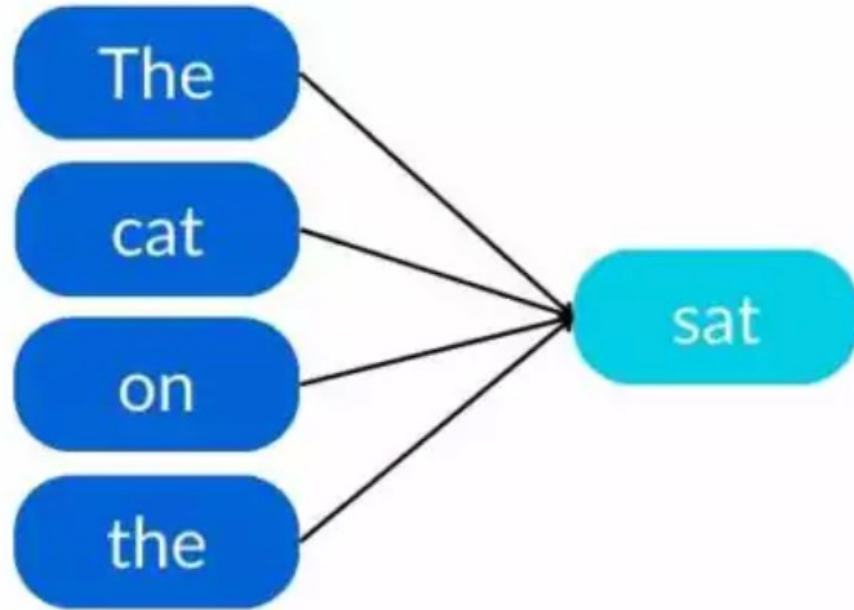
### Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Example Sentence: The cat sat on the mat.

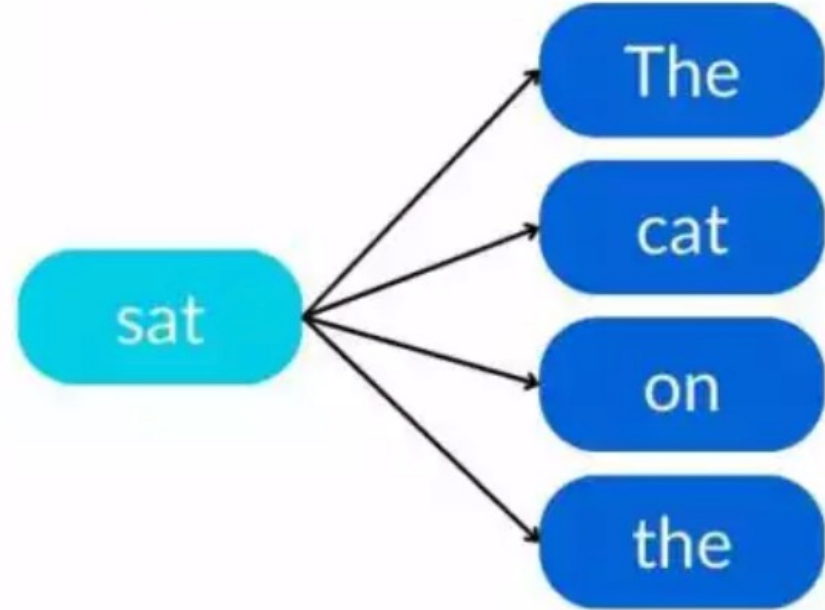
### Continuous Bag-of-Words (CBOW)

Goal: Given context words,  
predict the target word.



### Skip-gram Model

Goal: Given a word,  
predict the surrounding context words.



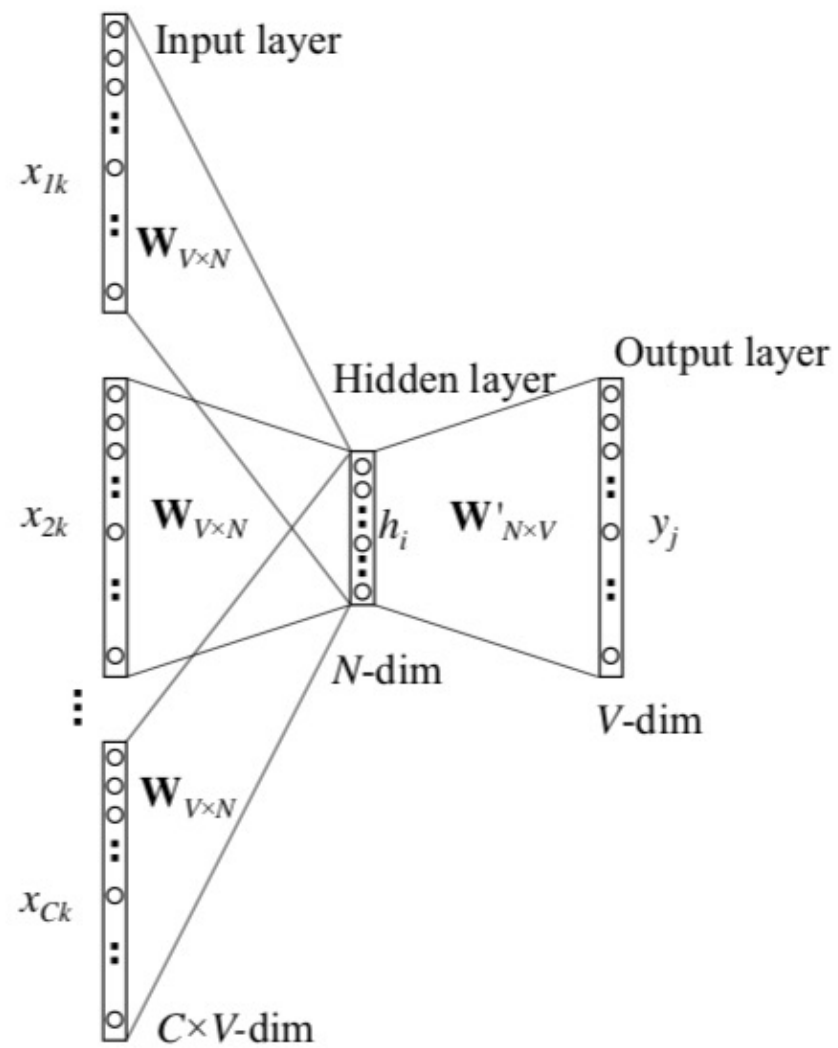


Figure 2: Continuous bag-of-words model

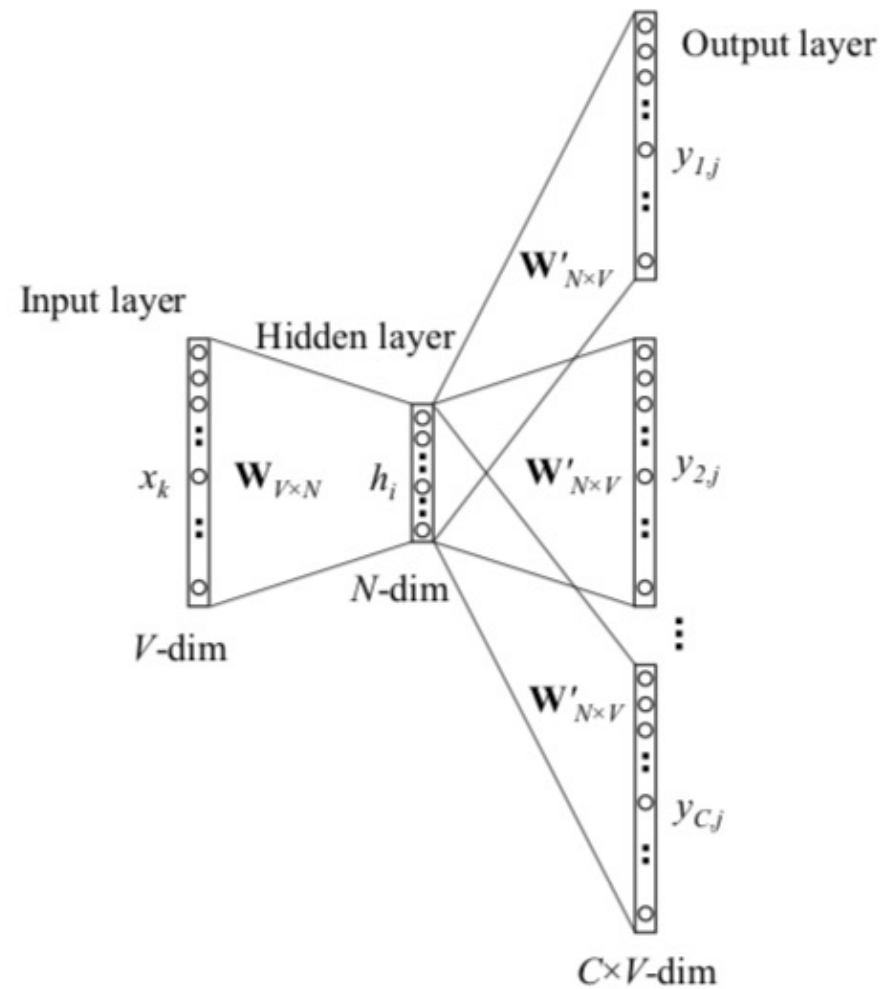


Figure 3: The skip-gram model.

# Embeddings

- **Embeddings**: vectors for representing words
- After multiple rounds of training, we will get embeddings that look something like this:

[0.35,-0.53,1.54,2.75,-3.03,0.24,0.35,0.93, ....]

# Representing words in a vector space



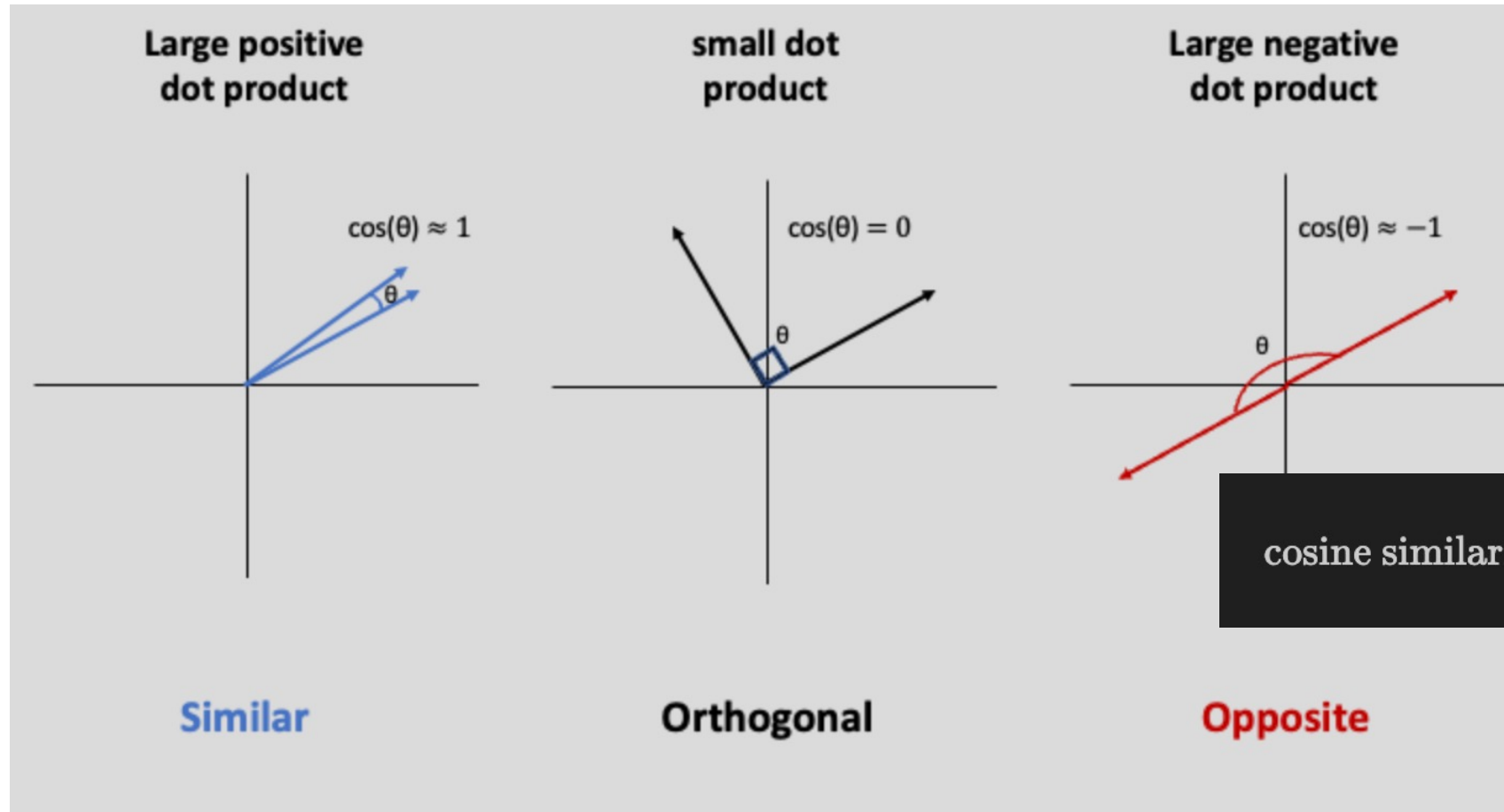
**Figure 6.1** A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from [Li et al. \(2015\)](#) with colors added for explanation. Jurafsky & Martin (2024) Speech and Language Processing, Ch.6

# GloVe (Pennington et al., 2014)

- Global Vectors for Word Representation ([website](#))
- Similar to Word2Vec but focuses more on co-occurrence information than calculating probabilities



# Measuring similarity



- Cosine similarity  
[-1, 1]

$$\text{cosine similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| ||\mathbf{b}||}$$

# Measuring similarity

$$\text{cosine similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| ||\mathbf{b}||}$$

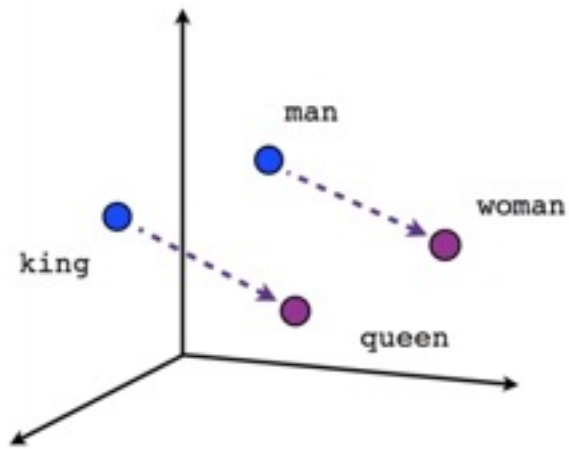
- Cosine similarity [-1, 1]

# Analogies

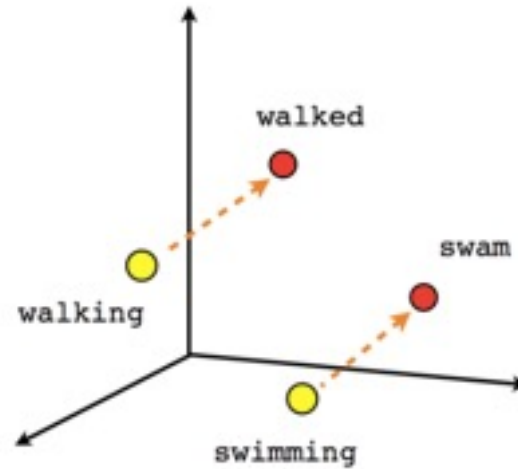
- *big: biggest*
- *small: \_\_\_\_\_*

$\text{vector}(\textit{biggest}) - \text{vector}(\textit{big}) + \text{vector}(\textit{small})$

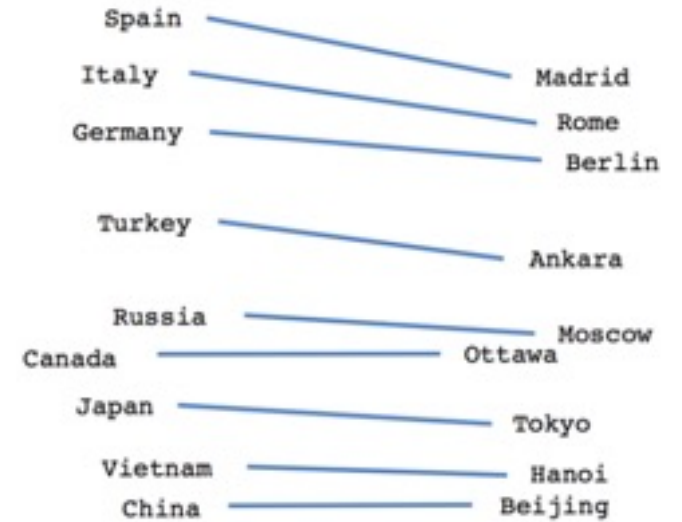
# Analogies



Male-Female



Verb tense



Country-Capital

# Demonstration

- Similarity score
- Analogy

*Note:* genism offers word2vec or glove pre-trained embeddings; but you can also train your own model.

# Assignment 2

- If you were not able to make an account on PCIBex, please come talk to me after class!

# Assignment 3

1. Calculate similarity scores for the word pairs you created for Assignment 2. Do it only for *sem* and *unr* conditions. Output: `similarity-scores.txt`.
2. By using similarity scores, get words that are semantically (un)related to the target words. Output: `unRelatedWords.txt`.
3. Update your PClbex experiment by (i) using the new materials you created in Task 2, and (ii) editing some specific experimental setup. For (i), replace your old *sem* and *unr* conditioned words with new words, using the word list in `unRelatedWords.txt`. For (ii), refer to the posted guideline.

# Assignment 3

## **What to submit**

1. Google Colab URL to your Python code for Task 1 and Task 2. I should be able to run this code without errors, see and download the two output .txt files.
2. URL for your PCIBex experiment.
  - If you're still experiencing issues with setting up your account on PCIBex, please come see me!