# Announcements

- HW 5 due **Wednesday, March 29 at 8pm**
  - Please start early!
- Project Milestone 2 due **Wednesday, April 5 at 8pm**
  - We will post the template this week

# Lecture 17: NLP (Part 1)

CIS 4190/5190

Spring 2023

# Goals of NLP

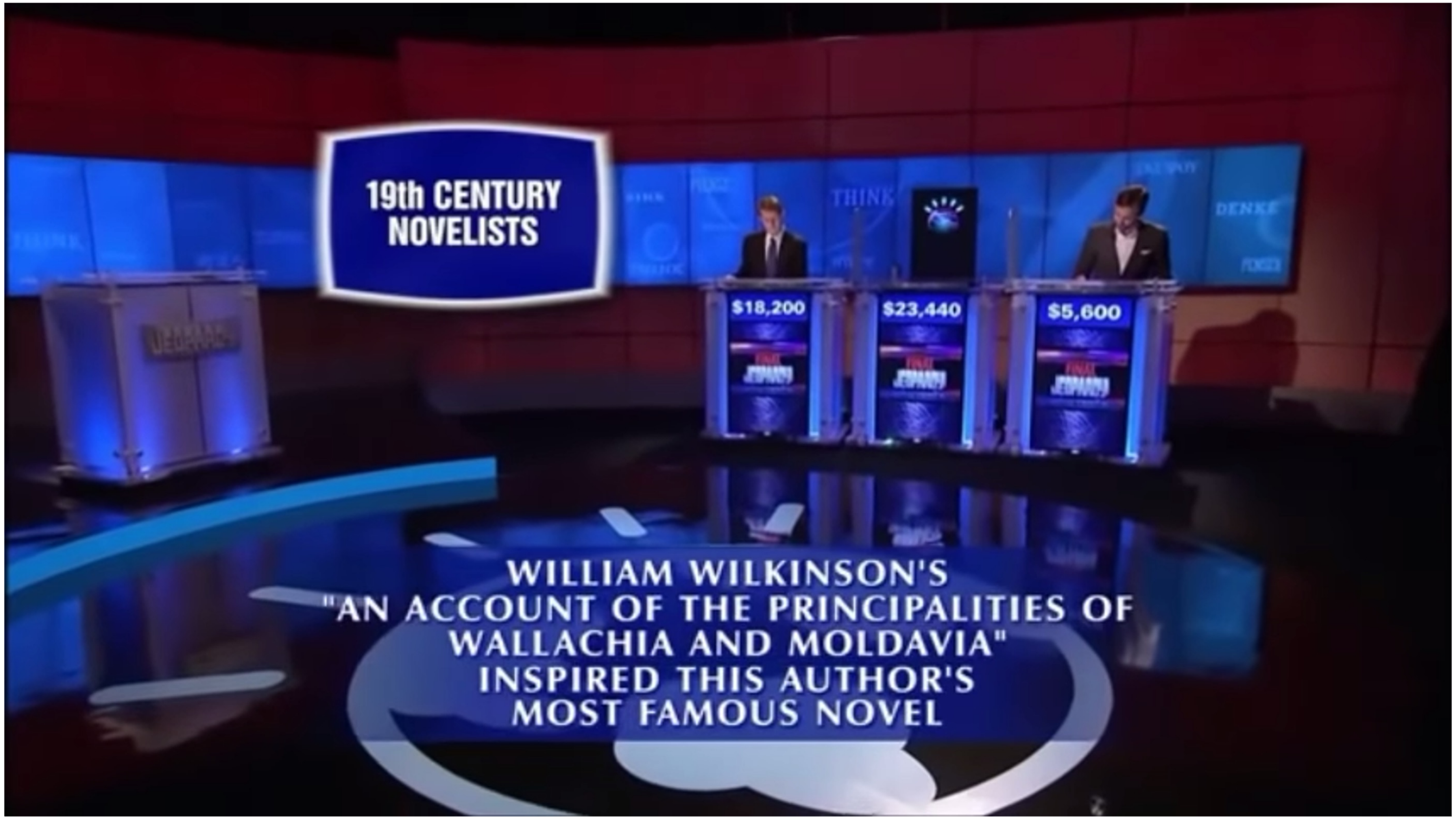
- Recognize spam email, fake news articles, etc.
- Read a textbook and solve an exam question
- Translate from English to French
- Search for webpages relevant to a search query
- Read tweets and understand public sentiment on a topic

- **Generally:** We would like to be able to understand text and extract all the same kinds of information in the same ways as humans might

# Language Understanding is Hard!

- Did Abraham Lincoln have an iPhone?
  - No! (requires common sense)
- Mary fought with Kate because she was a bad person. Who was a bad person? Mary or Kate?
  - Ambiguous (requires long-term context)
- The guitar didn’t fit into the box because it was too small. What was too small? The guitar or the box?
  - The box (requires common sense)

# IBM Watson Jeopardy! Challenge




https://www.youtube.com/watch?v=Sp4q60BsHoY

# Smart Assistant Advancements


Google Assistant

# Machine Translation

# Question Answering

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

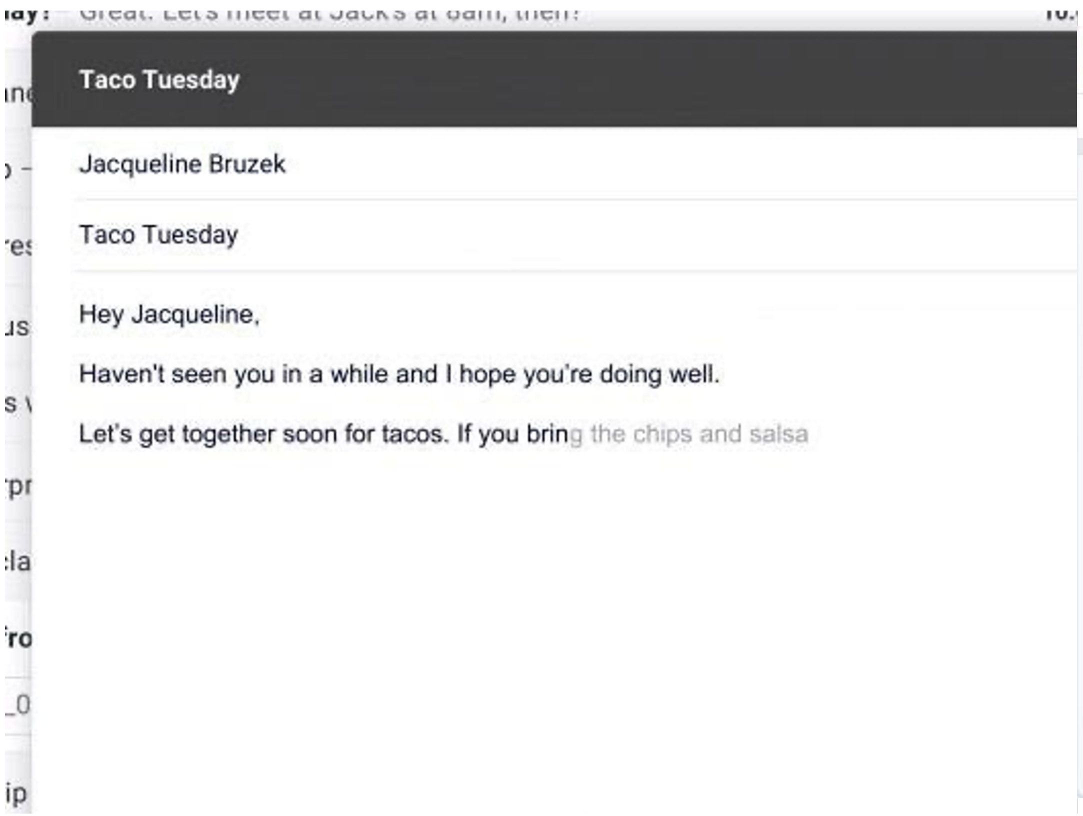
**Answer:** through contact with Persian traders

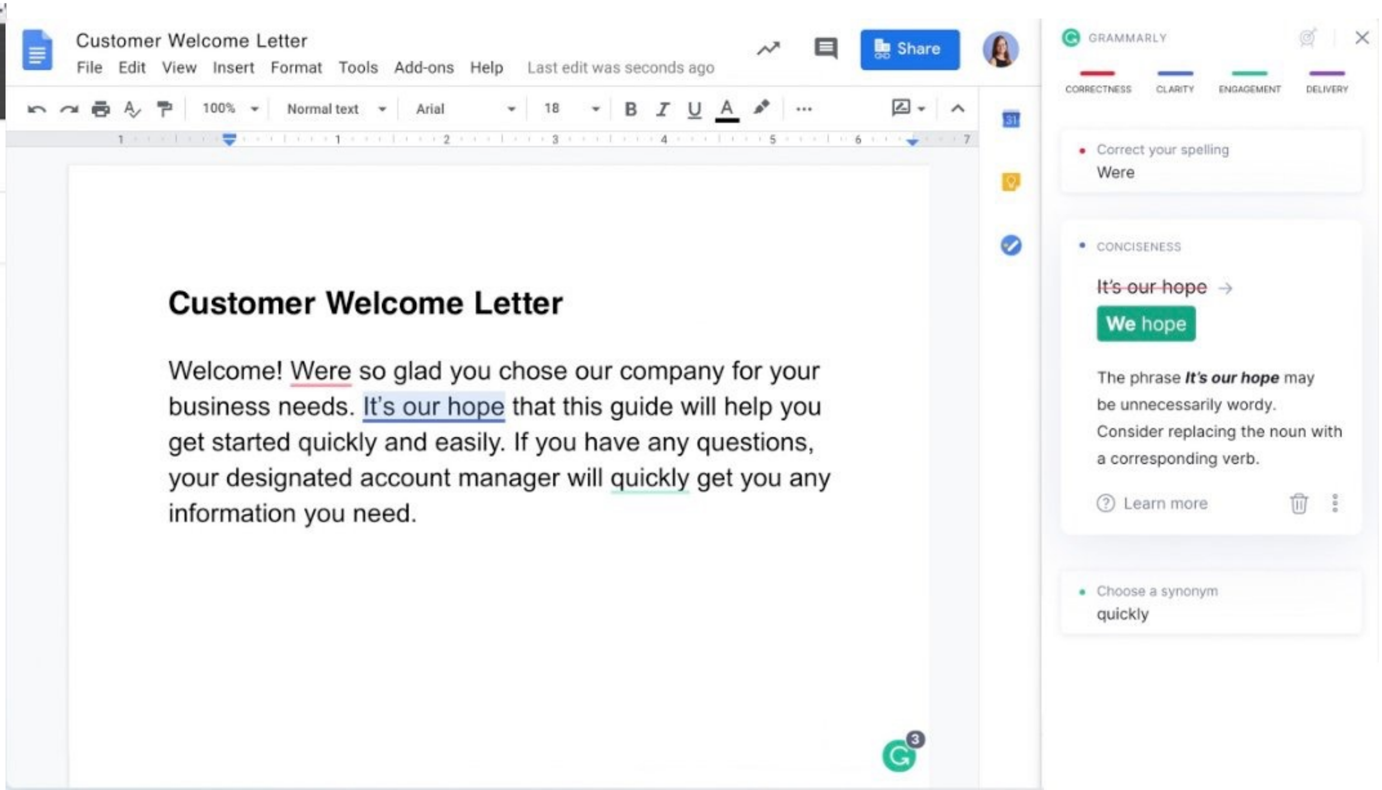
## Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

| Rank | Model | EM | F1 |
|---|---|---|---|
| | Human Performance<br>*Stanford University*<br>(Rajpurkar & Jia et al. '18) | 86.831 | 89.452 |
| 1<br>Apr 06, 2020 | SA-Net on Albert (ensemble)<br>*QIANXIN* | **90.724** | **93.011** |
| 2<br>May 05, 2020 | SA-Net-V2 (ensemble)<br>*QIANXIN* | 90.679 | 92.948 |
| 2<br>Apr 05, 2020 | Retro-Reader (ensemble)<br>*Shanghai Jiao Tong University*<br>http://arxiv.org/abs/2001.09694 | 90.578 | 92.978 |

# Text Completion



# Text Generation

**Prompt**

Write a creative ad for the following product to run on Facebook aimed at parents:

Product: Learning Room is a virtual environment to help students from kindergarten to high school excel in school.

**Sample response**

Are you looking for a way to help your child excel in school? Look no further than Learning Room! Our virtual environment is designed to help students from kindergarten to high school learn and grow. With our help, your child will be able to get ahead in school and achieve their academic goals.

# Basic NLP Pipeline

- **Classical approach**
  - **Step 1:** Manually construct feature mapping from text to $\mathbb{R}^d$
  - **Step 2:** Run supervised learning algorithm in conjunction with feature map

- **Deep learning approach**
  - **Step 1:** Design neural network architecture that can take text as input
  - **Step 2:** Train neural network end-to-end

# Bag of Words Feature Map

- **Idea:** Treat each document as an **unordered** set of words
  - Simple but can be effective choice in practice
- **Lexicon:** Set of "all possible words"
  - Union of words from all documents in the dataset
  - Use a dictionary
- Then, represent document as a vector $x \in \mathbb{R}^d$, where $d$ is number of words in the lexicon
  - $x_j$ is the number of occurrences of word $j$ in the document

# Bag of Words Feature Map

document $x$

number of times "abbey" occurs

| 0 | 0 | 1 | 0 | 0 | 0 | 4 | 0 | ... | 0 |
|---|---|---|---|---|---|---|---|---|---|
| aardvark | abacus | abandon | abase | abate | aberration | abbey | abbot | ... | zoo |

$\phi(x)$

# Shortcomings of Bag of Words

- Cannot distinguish word senses (which come from **context**)
  - "Took money out of the **bank**"
  - "Got stuck on the river **bank**"
  - "The pilot tried to **bank** the plane"

- Significance of some words vs. others
  - Articles ("a", "an", "the") vs. unusual terms ("hagiography")

# Shortcomings of Bag of Words

- Ignores the fact that some words are more similar than others
    - “I have a dog”
    - “I have a cat”
    - “I have a tomato”

- Ignores ordering of words
    - “Mary runs faster than Jack”
    - “Jack runs faster than Mary”

# Improvements to Bag of Words

- **$\boldsymbol{n}$-grams:** Each feature counts the number of times a sequence of $n$ words occurs in the document
  - "I have a cat" → ["I have": 1, "have a": 1, "a cat": 1]
  - **Shortcoming:** Quickly becomes high dimensional!

- **TF-IDF:** Downweight words that occur across many documents
  - "a" counts for a lot less than "hagiography"
  - Can be used for feature selection

# Alternatives?

- Can we automatically learn representations of words?

- We can use deep learning to do so, but classical unsupervised learning approaches can also work well
  - Specialized to NLP

# Word Embeddings

- **Embed words as vectors**
  - Automatically learn feature map $\phi(x) \in \mathbb{R}^d$

- **Bag-of-words:** $\phi(x) = \sum_{\text{word } i \in \text{document } x} \text{OneHot}(i)$
  - $\text{OneHot}(i)$ is the vector with all zeros except it equals one at position corresponding to word $i$
  - $\text{OneHot}(\text{“dog”}) = [0, 0, 0, 1, 0, 0, 0]$
  - $\text{OneHot}(\text{“cat”}) = [1, 0, 0, 0, 0, 0, 0]$

- We want to learn embeddings where the structure captures semantics, e.g., nearby vectors correspond to similar words

# Document-Term Matrix

- Counts the number of times each word occurs in each document

| Wikipedia Article<br>Words | **Cat** | **Dog** | Apple Inc. | Apple (fruit) | Microsoft Inc. | … |
|---|---|---|---|---|---|---|
| **a** | 377 | 370 | 842 | 231 | 286 | … |
| **the** | 929 | 787 | 1690 | 503 | 872 | … |
| **apple** | 0 | 0 | 1091 | 166 | 14 | … |
| **computer** | 0 | 0 | 88 | 0 | 36 | … |
| **fur** | 15 | 2 | 0 | 0 | 0 | … |
| **hair** | 6 | 6 | 0 | 0 | 0 | … |
| **…** | … | … | … | … | … | … |

# Document-Term Matrix

- **Key observation:** Similar words tend to co-occur

| Wikipedia Article / Words | **Cat** | **Dog** | Apple Inc. | Apple (fruit) | Microsoft Inc. | … |
|---|---|---|---|---|---|---|
| **a** | 377 | 370 | 842 | 231 | 286 | … |
| **the** | 929 | 787 | 1690 | 503 | 872 | … |
| **apple** | 0 | 0 | 1091 | 166 | 14 | … |
| **computer** | 0 | 0 | 88 | 0 | 36 | … |
| **fur** | 15 | 2 | 0 | 0 | 0 | … |
| **hair** | 6 | 6 | 0 | 0 | 0 | … |
| **…** | … | … | … | … | … | … |

# Document-Term Matrix

- **Key observation:** Similar words tend to co-occur

| Wikipedia Article / Words | **Cat** | **Dog** | Apple Inc. | Apple (fruit) | Microsoft Inc. | … |
|---|---|---|---|---|---|---|
| **a** | 377 | 370 | 842 | 231 | 286 | … |
| **the** | 929 | 787 | 1690 | 503 | 872 | … |
| **apple** | 0 | 0 | 1091 | 166 | 14 | … |
| **computer** | 0 | 0 | 88 | 0 | 36 | … |
| **fur** | 15 | 2 | 0 | 0 | 0 | … |
| **hair** | 6 | 6 | 0 | 0 | 0 | … |
| **…** | … | … | … | … | … | … |

# Document-Term Matrix

- **Key observation:** Similar words tend to co-occur

| Wikipedia Article / Words | **Cat** | **Dog** | Apple Inc. | Apple (fruit) | Microsoft Inc. | … |
|---|---|---|---|---|---|---|
| **a** | 377 | 370 | 842 | 231 | 286 | … |
| **the** | 929 | 787 | 1690 | 503 | 872 | … |
| **apple** | 0 | 0 | 1091 | 166 | 14 | … |
| **computer** | 0 | 0 | 88 | 0 | 36 | … |
| **fur** | 15 | 2 | 0 | 0 | 0 | … |
| **hair** | 6 | 6 | 0 | 0 | 0 | … |
| **…** | … | … | … | … | … | … |

# Document-Term Matrix

- **Key observation:** Similar words tend to co-occur
- **Potential idea:** Represent word by its row!

| Wikipedia Article / Words | **Cat** | **Dog** | Apple Inc. | Apple (fruit) | Microsoft Inc. | … |
|---|---|---|---|---|---|---|
| **a** | 377 | 370 | 842 | 231 | 286 | … |
| **the** | 929 | 787 | 1690 | 503 | 872 | … |
| **apple** | 0 | 0 | 1091 | 166 | 14 | … |
| **computer** | 0 | 0 | 88 | 0 | 36 | … |
| **fur** | 15 | 2 | 0 | 0 | 0 | … |
| **hair** | 6 | 6 | 0 | 0 | 0 | … |
| **…** | … | … | … | … | … | … |

# Term-Term Matrix

- **Shortcoming:** Document-term matrix depends heavily on structure of documents in the training data

- **Alternative:** Term-term matrix counts co-occurrences of pairs of words across all documents

# Term-Term Matrix

- Count how many times a word appears within the neighborhood "context" of another word (e.g., 4 words to the left/right)

| Words / Words | **pet** | **play** | tire | engine | run | … |
|---|---|---|---|---|---|---|
| **dog** | 872 | 649 | 1 | 7 | 378 | … |
| **cat** | 789 | 831 | 5 | 0 | 285 | … |
| **tomato** | 12 | 4 | 290 | 927 | 562 | … |
| **…** | … | … | … | … | … | … |

# Term-Term Matrix

- Count how many times a word appears within the neighborhood "context" of another word (e.g., 4 words to the left/right)
  - **Idea:** Represent each word by its row

| Words \ Words | **pet** | **play** | tire | engine | run | … |
|---|---|---|---|---|---|---|
| **dog** | 872 | 649 | 1 | 7 | 378 | … |
| **cat** | 789 | 831 | 5 | 0 | 285 | … |
| **tomato** | 12 | 4 | 290 | 927 | 562 | … |
| **…** | … | … | … | … | … | … |

# Term-Term Matrix

- **Intuition:** Each words is represented by words in its neighborhood

- “The **distributional hypothesis** in linguistics is derived from the semantic theory of language usage, i.e. words that are used and occur in the same contexts tend to purport similar meanings.”
  - *“A word is characterized by the company it keeps”* – John Firth

# Term-Term Matrix

- For example, the words that frequently co-occur with “dog” in a sentence might be words like “play”, “pet”, “sleep”, “fur”, “feed”, etc.
  - Would these words tend to co-occur with “cat”?
  - How about with “tomato”?
  - “I have a pet cat”
  - “I have a pet dog”
  - “I have a pet tomato”

- Similar words have similar embeddings

# Shortcomings of Classical Approaches

- **Word embedding vector dimensions:**
  - Document-term = # of documents
  - Term-Term = # of words

- These are huge vectors!
  - Can we get a more compact representation?

- **Idea:** Train a neural network classifier to predict whether one word will co-occur in the context of another word
  - The **classifier weights** can be interpreted as word embeddings!

# Word2Vec

- **Idea:** Train a neural network classifier to predict whether one word will co-occur in the context of another word

- Then, the **classifier weights** can be interpreted as word embeddings!

# Word2Vec Training Data

- “The quick brown fox jumped over the lazy dog.”

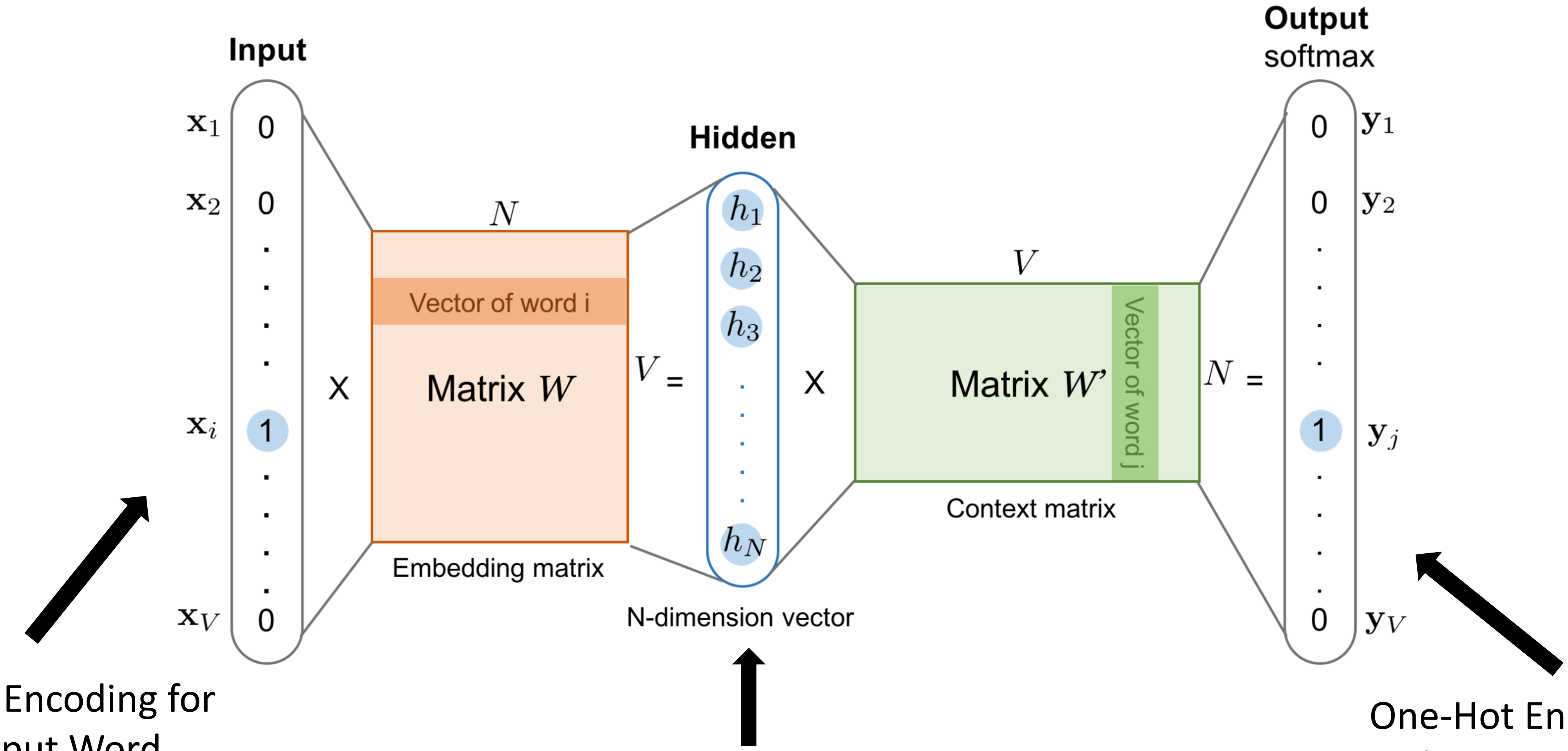
| **Word** | **Context** |
|---|---|
| the | [quick] |
| quick | [the, brown] |
| brown | [quick, fox] |
| … | … |

# Word2Vec Training Data

- "The quick brown fox jumped over the lazy dog."

| **Word** | **Context** |
|---|---|
| the | quick |
| quick | the |
| quick | brown |
| brown | quick |
| brown | fox |
| … | … |

# Word2Vec Model



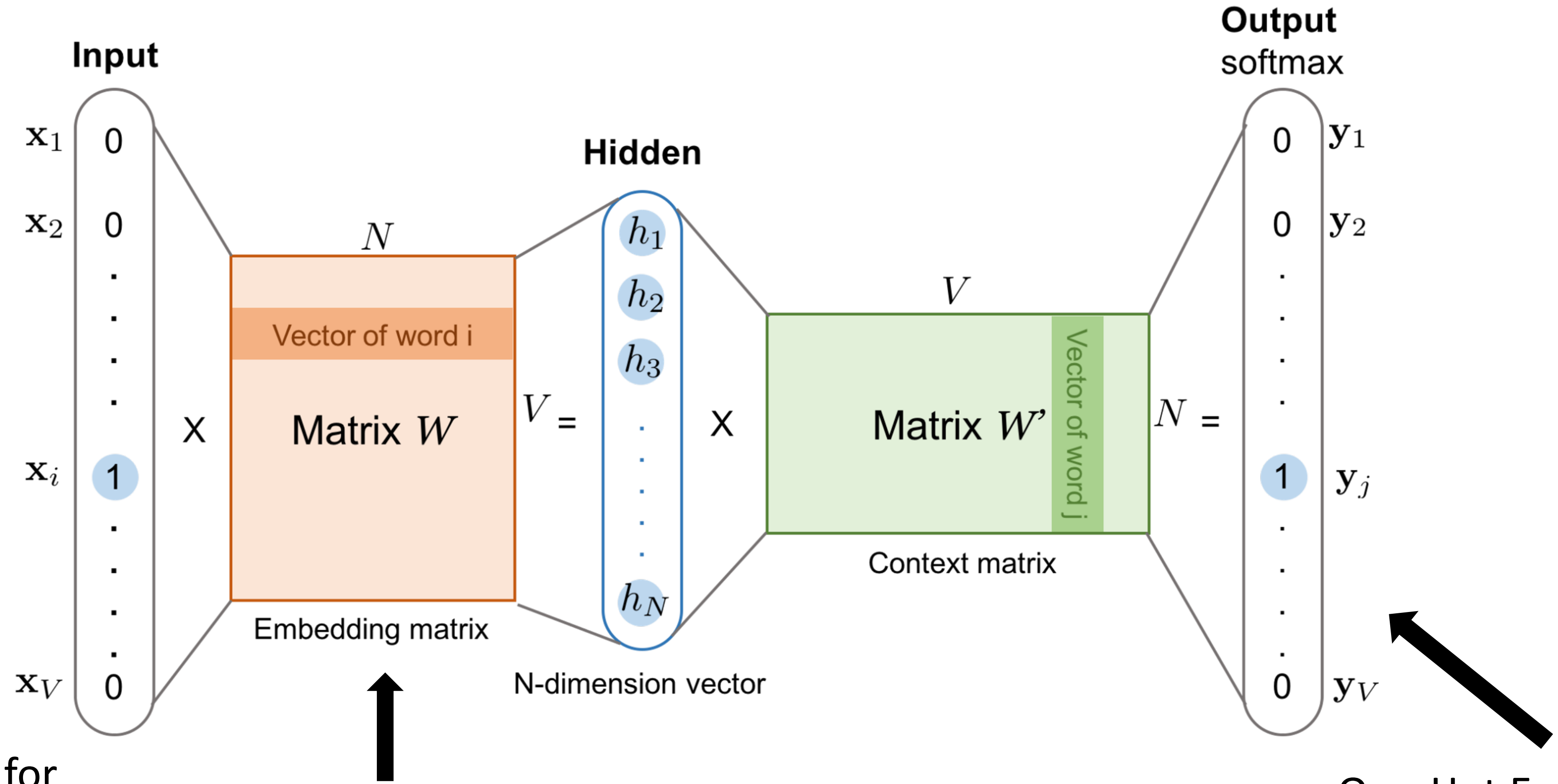

One-Hot Encoding for the Input Word

One-Hot Encoding for the Output Word

$N$ hidden units, for $N \ll V$

Source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

# Word2Vec Model

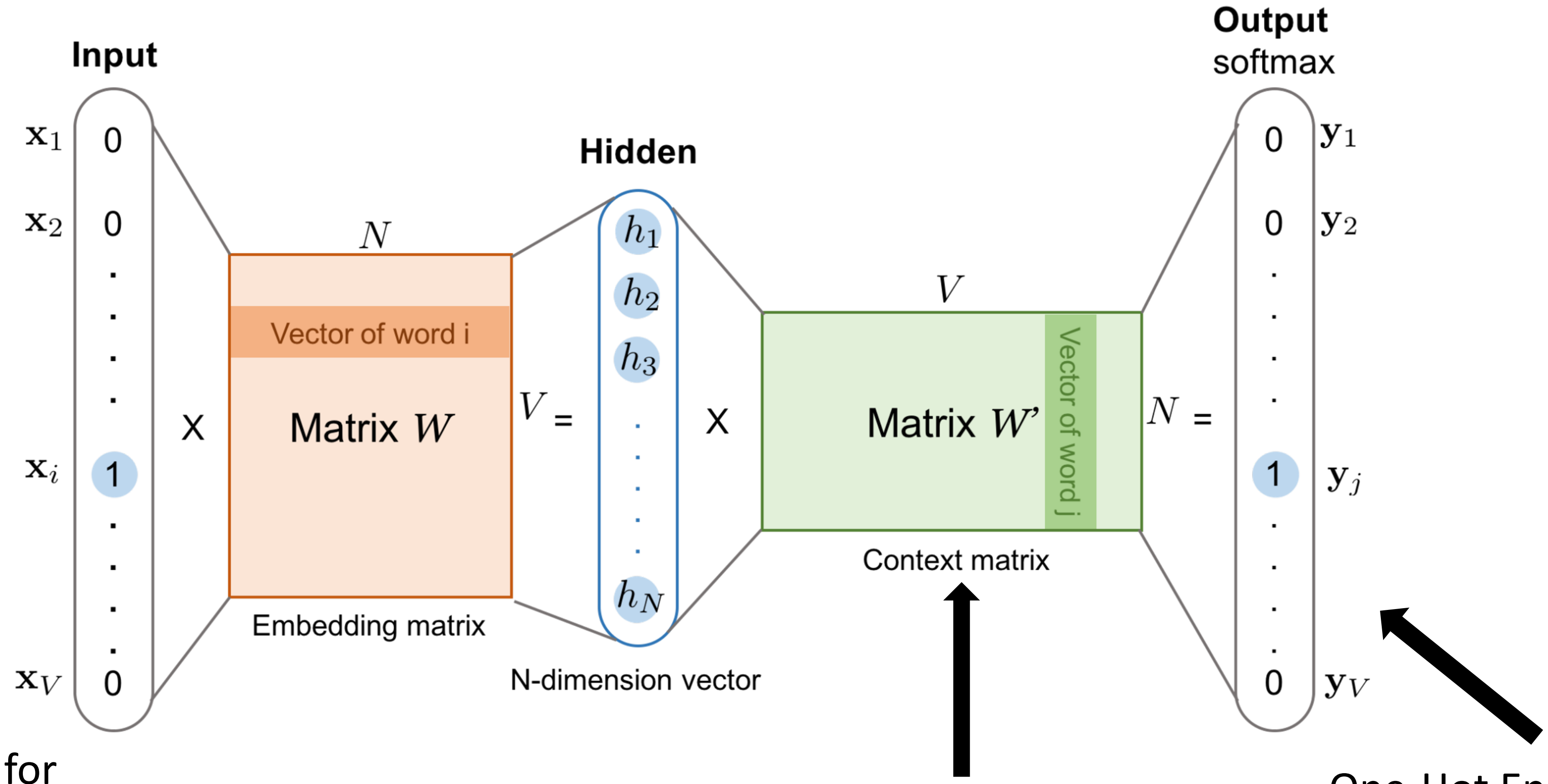



One-Hot Encoding for the Input Word

One-Hot Encoding for the Output Word

- $N$ columns, $V$ (vocabulary size) rows
- Each row corresponds to a word
- Row $i$ = embedding for word $i$, called "target embedding"

# Word2Vec Model

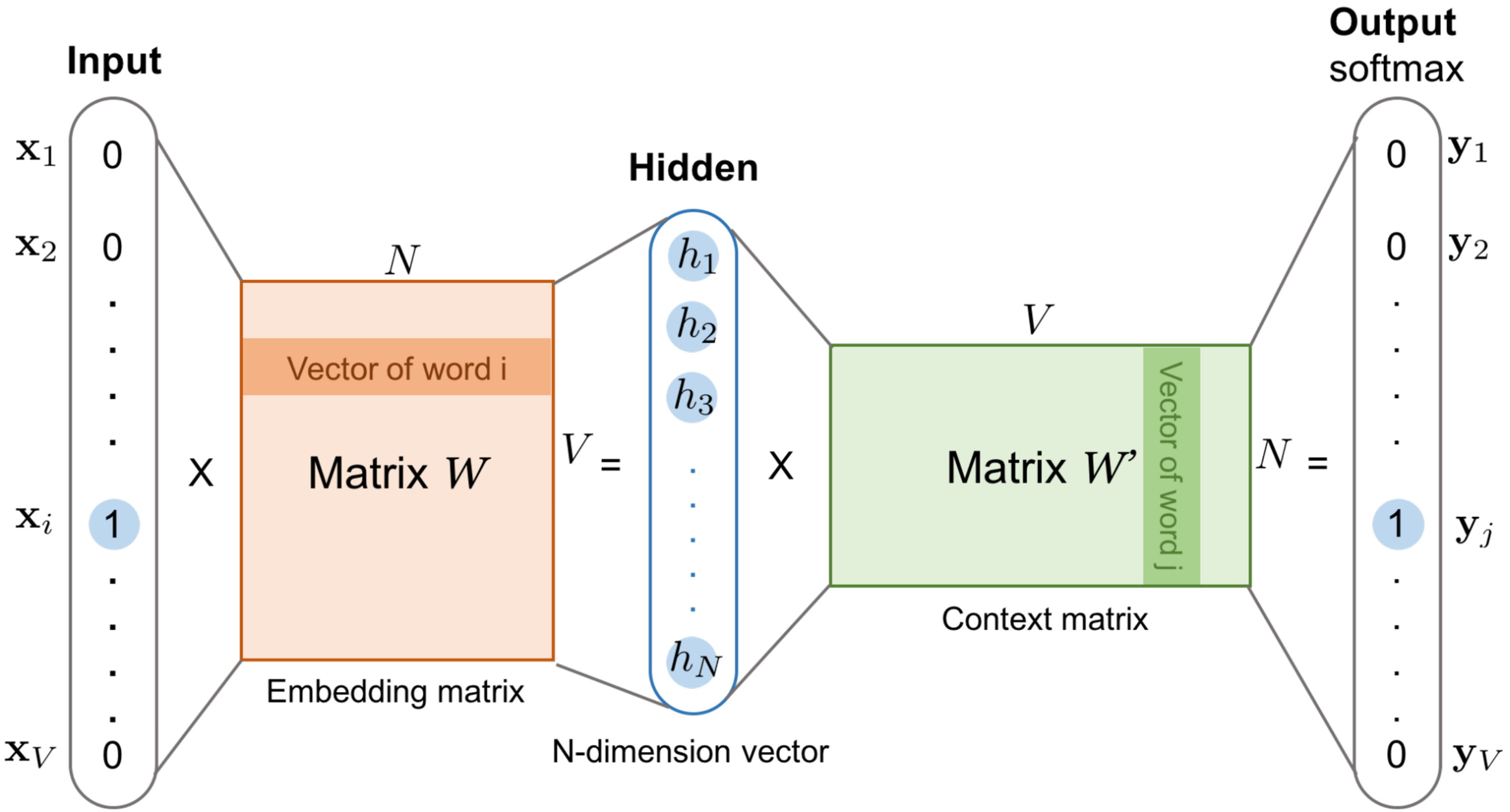



One-Hot Encoding for the Input Word

One-Hot Encoding for the Output Word

- $V$ (vocabulary size) columns, $N$ rows
- Each column corresponds to a word
- Column $i$ = embedding for word $i$, called "context embedding"

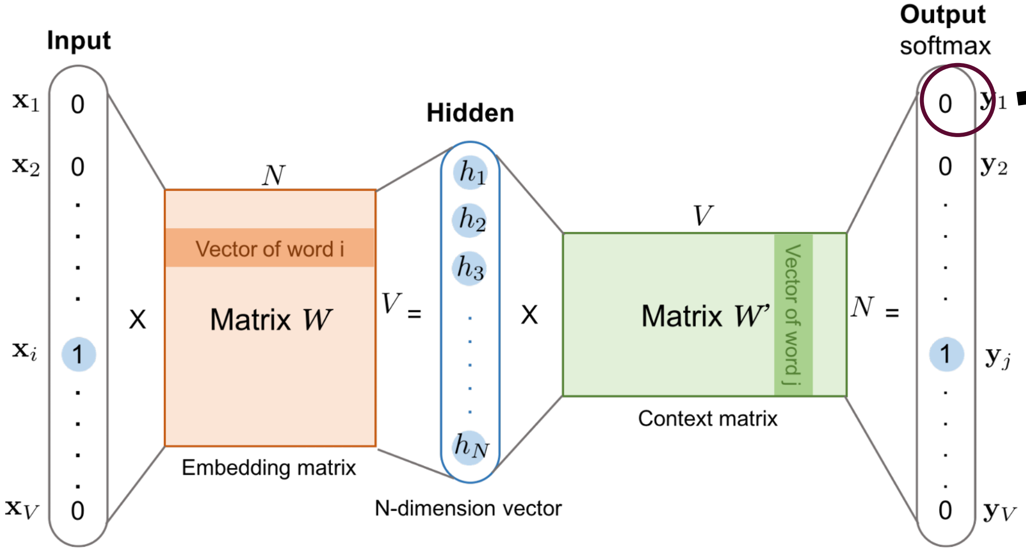
# Word2Vec Model




We can concatenate the target and context embeddings to form our final word embedding

# Word2Vec Training

- Standard softmax loss, then train the neural network

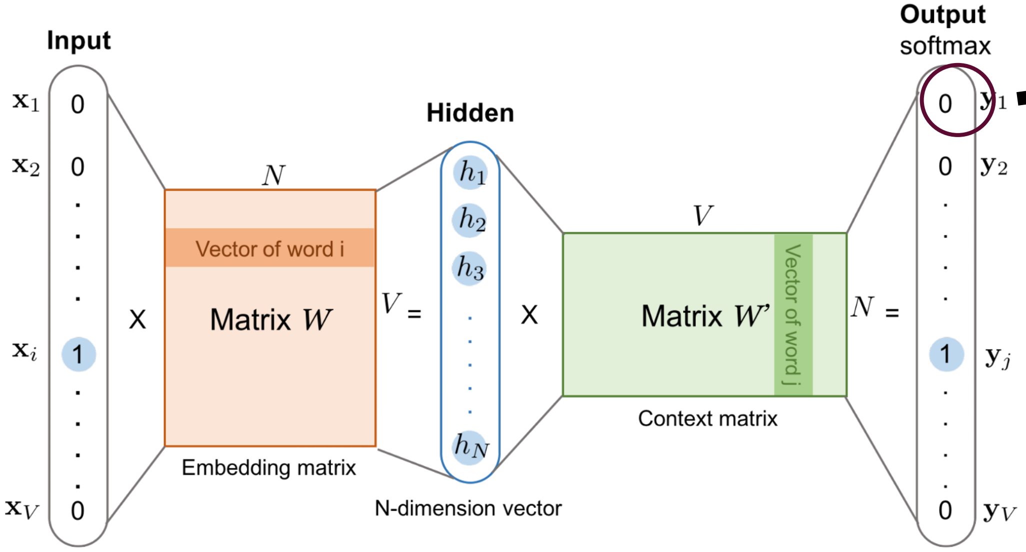



$$p(w_o|w_{in}) = \frac{\exp(v'_{w_o} . v_{w_{in}})}{\sum_{k=1}^{V} \exp(v'_{w_k} . v_{w_{in}})}$$

- Computing this denominator will be expensive.
- Remember that the vocabulary size V is of the order of millions of words!

# Word2Vec Training

- Standard softmax loss, then train the neural network




$$p(w_o|w_{in}) = \frac{\exp(v'_{w_o} . v_{w_{in}})}{\sum_{k=1}^{K} \exp(v'_{w_k} . v_{w_{in}})}$$

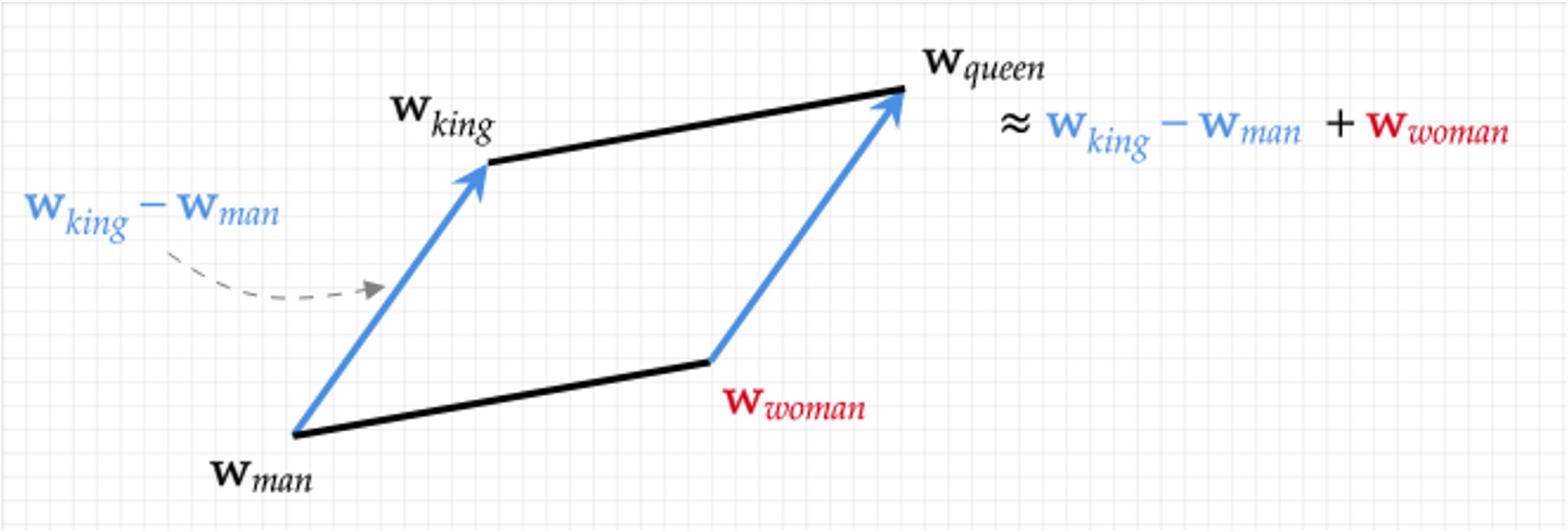
- **Simple Trick:** Sample some random $K-1 \ll V$ negative example words for each sample. e.g. $K = 2, 5, 20$ etc.
- Also means we need to update many fewer weights during each iteration of gradient descent.

# Properties of Word2Vec

- Words that co-occur have vector representations that are close together (in Euclidean distance)
  - “sofa” and “couch” (synonyms) will be close together
  - But also things like “hot” and “cold” (antonyms)
  - People say “It’s ____ outside today” for both

# Properties of Word2Vec

- Vector operations (vector addition and vector subtraction) on word vectors often capture the semantic relationships of their words.

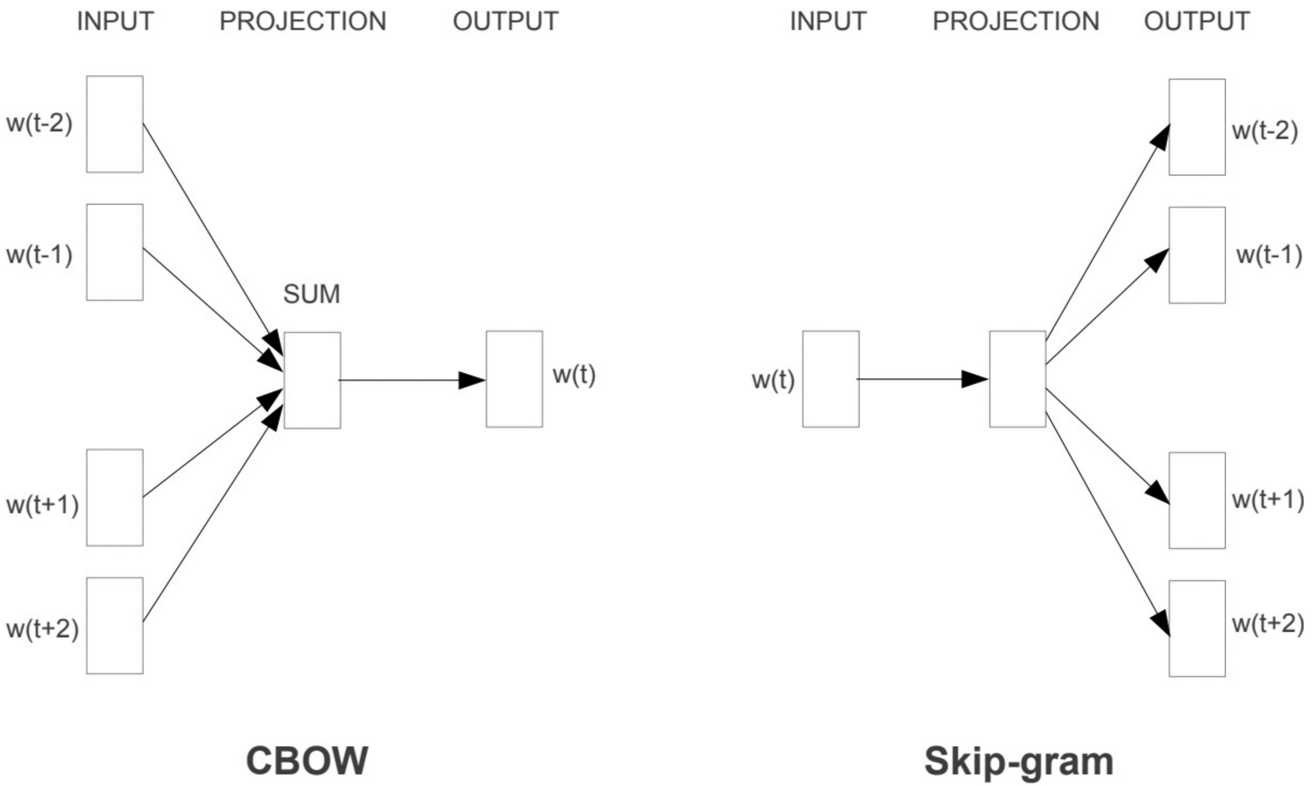



Source: https://www.ed.ac.uk/informatics/news-events/stories/2019/king-man-woman-queen-the-hidden-algebraic-struct

# Use in Practice

- GLoVe is an alternative word vector embedding similar to word2vec
- Available freely, and often used off-the-shelf:
  - English word2vec weights trained on Google News data
  - GloVe vectors trained on the Common Crawl dataset and a Twitter dataset
- If you have a lot of training data or a different/niche domain (e.g., medical), you may want to train your own word vectors!

# Other Variations




Predict word from bag-of-words context

Predict context from word

# From Words to Documents

- Sentence2Vec, Paragraph2Vec scale these Word2Vec ideas to learn direct embeddings for sentences / paragraphs

- However, much more common to treat as a sequence of words, and represent each word by its word2vec-style representation

- Sequence models have produced huge advances in NLP

# Words in Context

- While word2vec is trained based on context, after training, it is applied independently to each word
  - E.g., train linear regression of sum of word vectors, or n-grams
- **Why is this problematic?**
  - “He ate a tasty apple”
  - “He wrote his essay on his Apple computer”
- Both use the same embedding!