# Intro to Language Model and Traditional Language Modeling

Pavlos Protopapas

# Outline

Natural Language Processing

Text Preprocessing

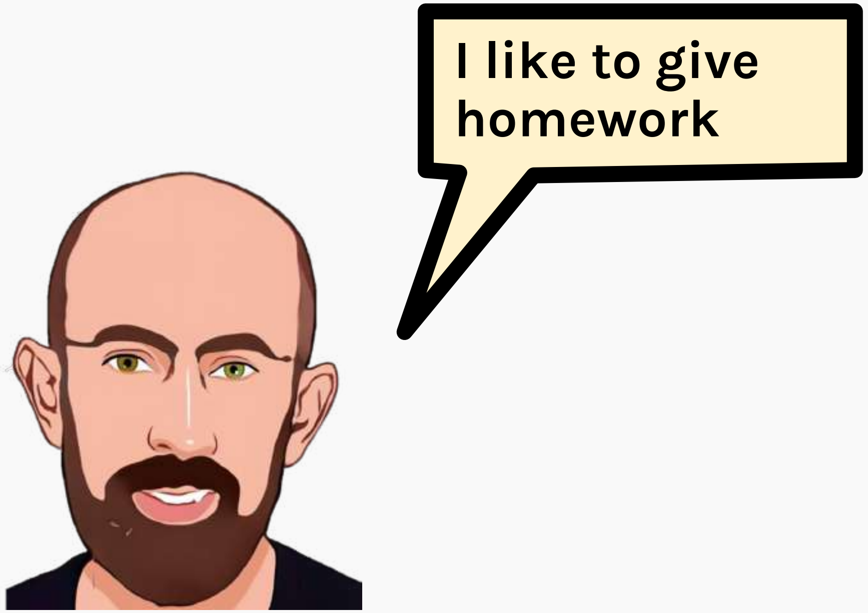
Language Modeling

- Unigrams
- Bigrams
- Neural Networks for Language Modeling

Evaluation

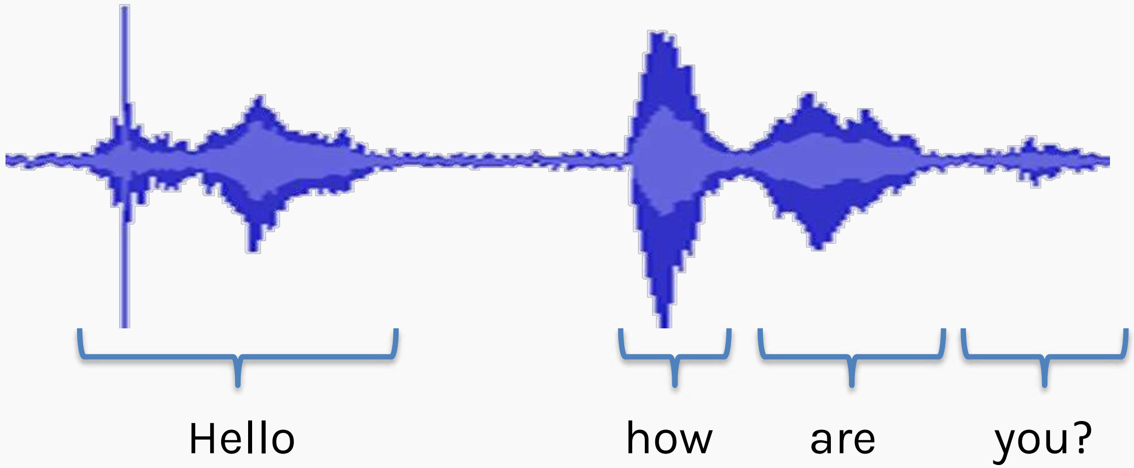
# Natural Language Processing

Natural Language Processing (NLP) is the field of study that focuses on the interaction between computers and humans through natural language, aiming to enable machines to understand, interpret, and respond to human language in a meaningful way.



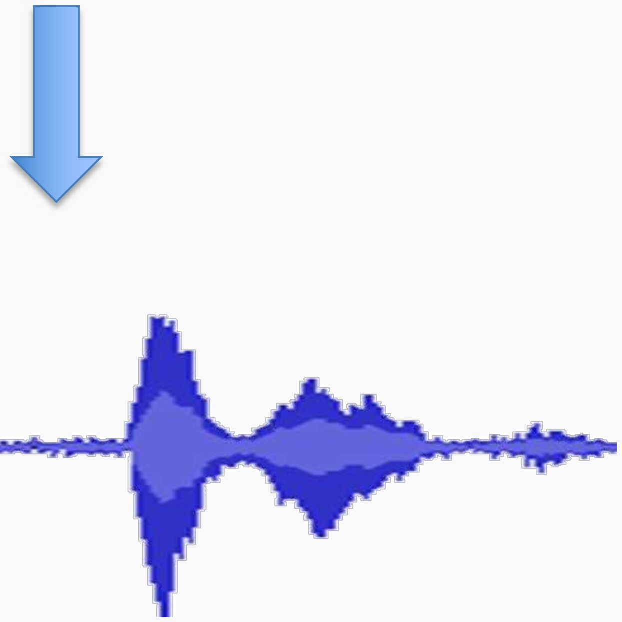

## Text recognition

## Speech recognition

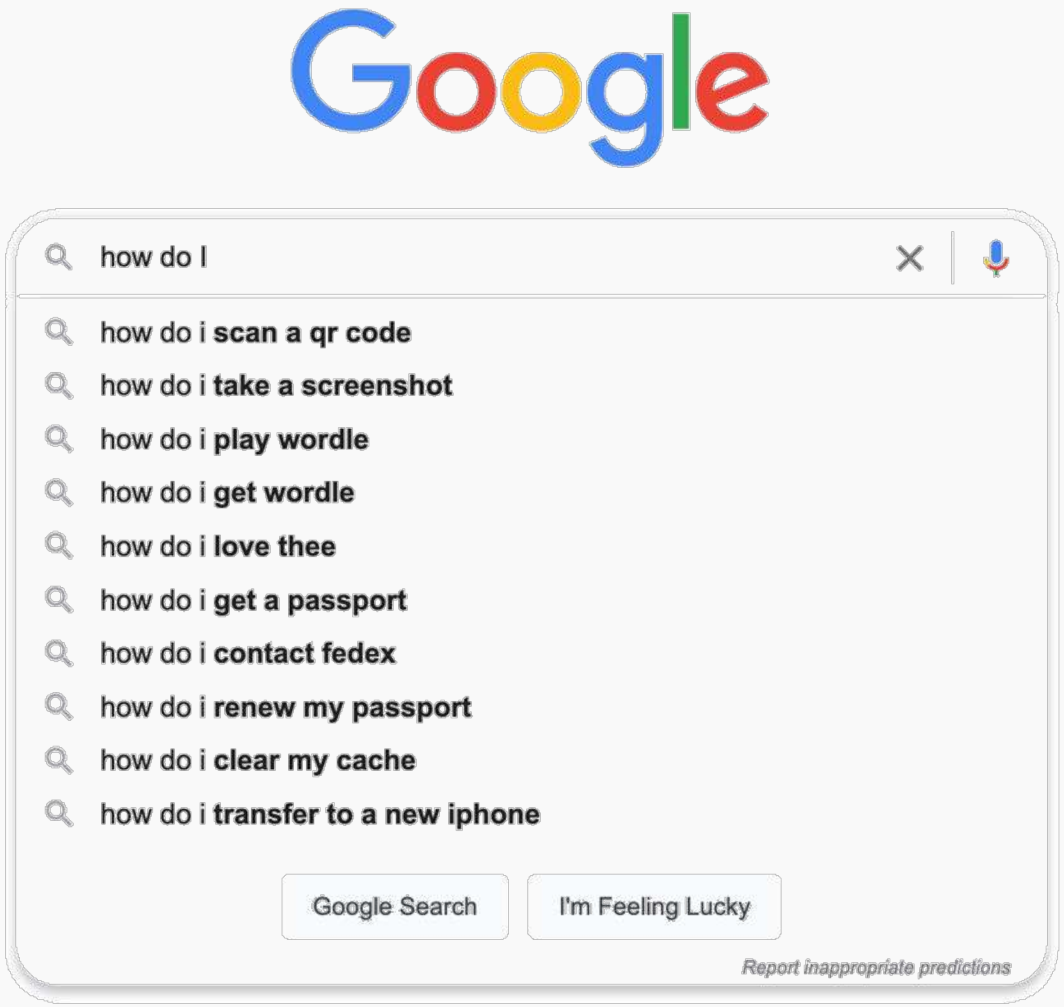


## Text to speech

Hello how are you?

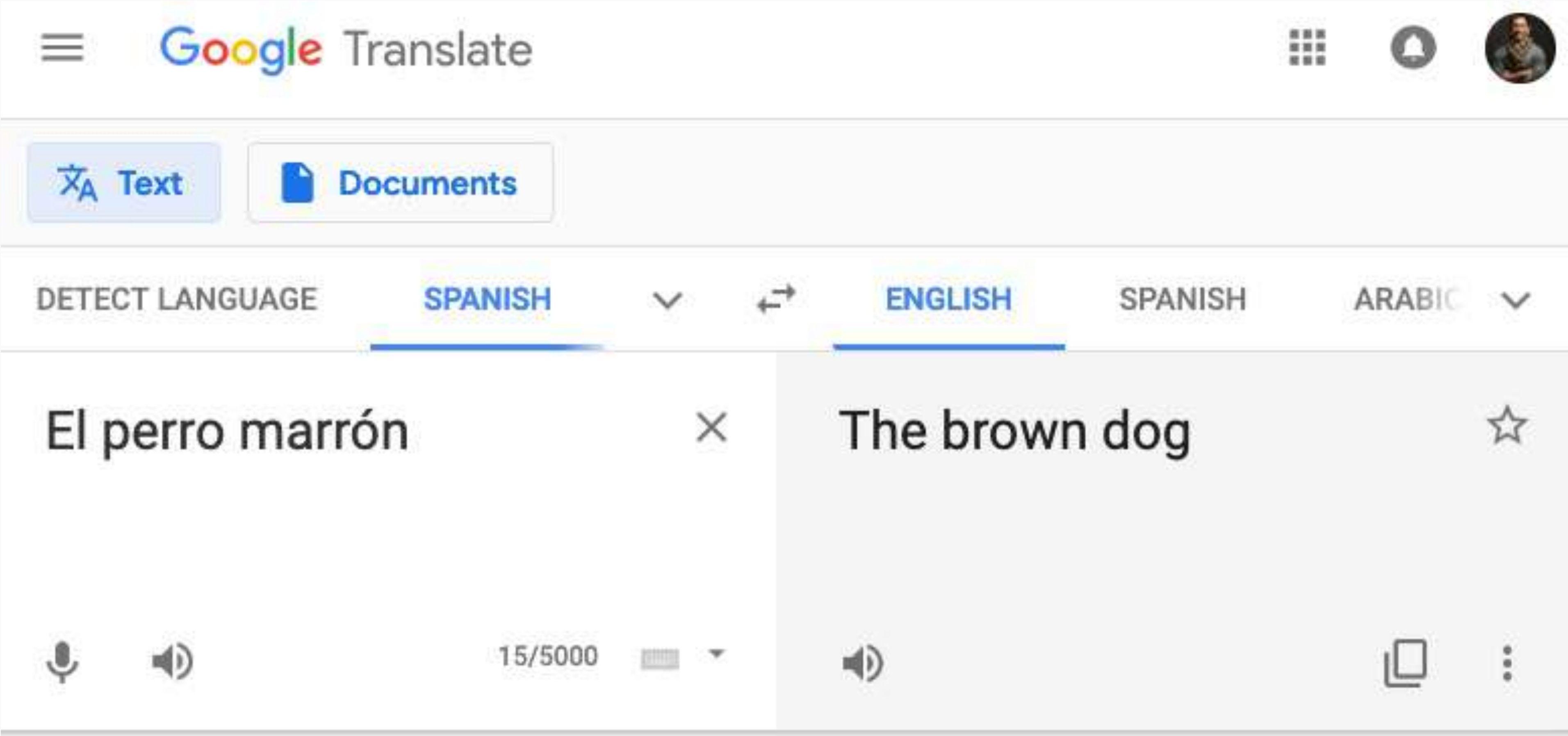


## Sentence prediction



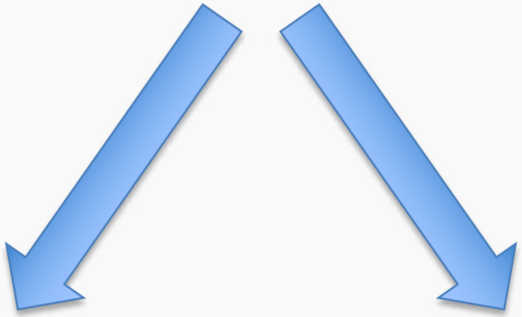
## Sentence prediction

## Translation



# Sentiment analysis

"The movie could have been better. The lead actress did an amazing performance."



Good movie

Bad movie

## Named Entity Recognition

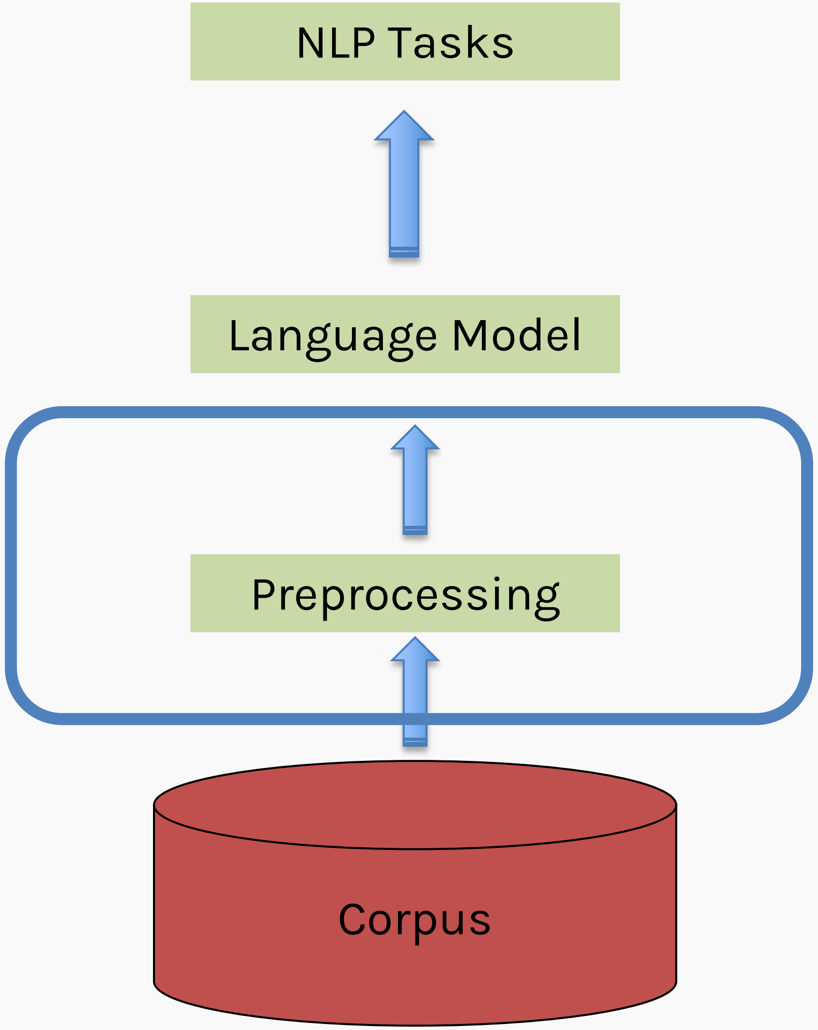
contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

## ChatGPT


Examples
Capabilities
Limitations
"Explain quantum computing in simple terms"
Remembers what user said earlier in the conversation
May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?"
Allows user to provide follow-up corrections
May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?"
Trained to decline inappropriate requests
Limited knowledge of world and events after 2021

# Pipeline

Before we jump into language modelling, let's see how we process our data for the model to be able to understand it.



# Outline

Natural Language Processing

**Text Preprocessing**

Language Modeling

- Unigrams
- Bigrams
- Neural Networks for Language Modeling

Evaluation

# Language Modelling: **starting point**

- Text is composed of a series of words that carry meaning. To process it, we break down the text into smaller units known as tokens.
- So, a sentence transforms into a sequence of these tokens.
- All the unique tokens from a dataset make up the vocabulary.
- The bigger the vocabulary the bigger the training set required.

# Tokenization

We need to define the basic unit (token) of a sentence.

**First Approach:** whitespace

Split the words on whitespaces only.

# Tokenization

We need to define the basic unit (token) of a sentence.

**First Approach:** whitespace

Split the words on whitespaces only.

“The award-winning actor arrived today.”

# Tokenization

We need to define the basic unit (token) of a sentence.

**First Approach:** whitespace

Split the words on whitespaces only.

Usually, punctuations are removed before this stage and capitalization is set to lower case

| | | | | |
|---|---|---|---|---|
| “the | award-winning | actor | arrived | today” |
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |

# Tokenization

**First Approach:** whitespace

Split the words on whitespaces only.

| “the | award-winning | actor | arrived | today” |
|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |

Hyphenated phrases like “*award-winning*” are not split.

| “i | haven’t | seen | her | since | yesterday” |
|---|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ |

Conjunctions such as *haven’t* are not split.

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

The fragment appears enough times in the corpus

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

The token loses their direct interpretability but makes a more flexible approach.

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

How can we define a token?

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

How can we define a token?

| “the | award | - | winning | actor | arrived | yesterday” |
|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ | $w_7$ |

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

How can we define a token?

| | | | | | | |
|---|---|---|---|---|---|---|
| “the | award | - | winning | actor | arrived | yesterday” |
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ | $w_7$ |

“turn the handle counterclockwise”

# Tokenization

**Second Approach:** sub-word tokenization

Split the words on statistically significant fragments.

How can we define a token?

| “the | award | - | winning | actor | arrived | yesterday” |
|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ | $w_7$ |

| “turn | the | handle | counter | c | lock | wise” |
|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ | $w_7$ |

Tokenization is an area of research in itself.

# Outline

Natural Language Processing

Preprocessing

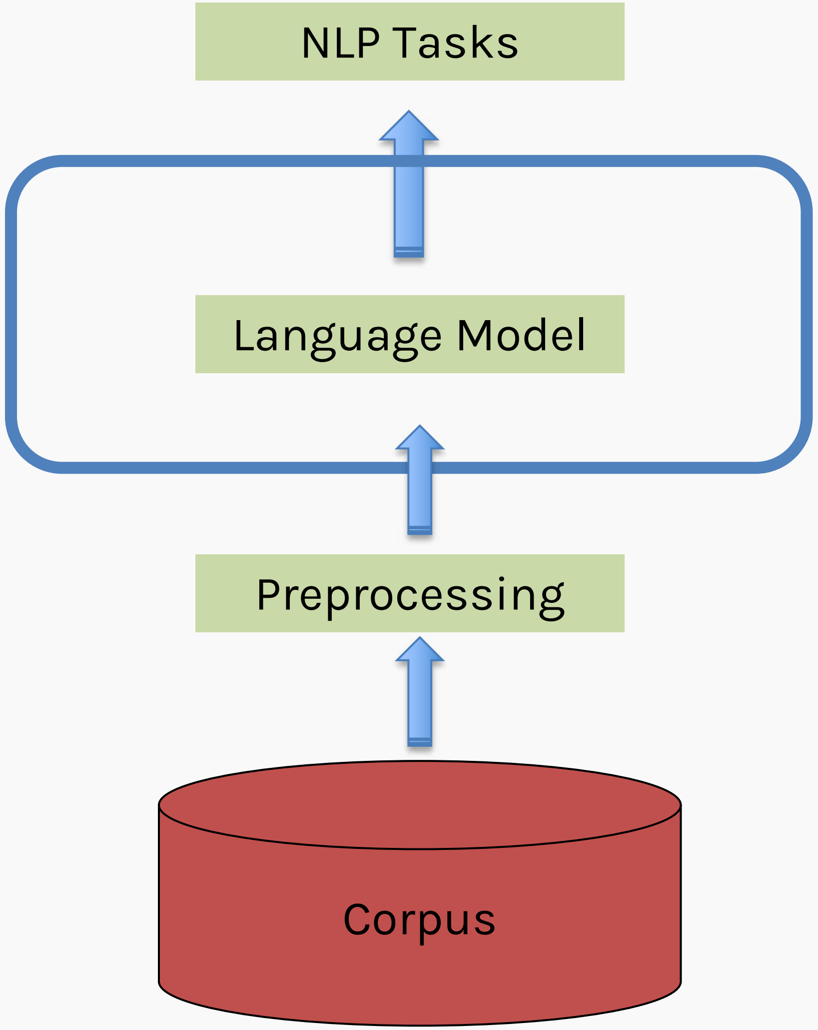
**Language Modeling**

- Unigrams
- Bigrams
- Neural Networks for Language Modeling

Evaluation

# Pipeline

Next step in the language modelling



# Language Modelling: Formal Definition

We model any sequential data as follows:

This compounds for all subsequent events, too

$$P(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1}, \dots, x_1)$$

Joint distribution of all measurements

Conditional probability of an event, depends on all of the events that occurred before it.

# Example

If we want to know the probability of the the next on-screen Sesame Street character:

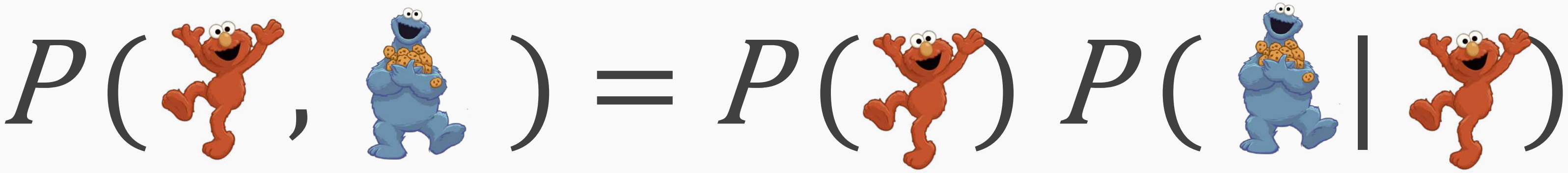
Scene 1

Scene 2

Scene 3

# Example

Remember that, when we evaluate a distribution, we mean

# Example

The probability of the the next on-screen Sesame Street character can be computed as

Scene 1

Scene 2

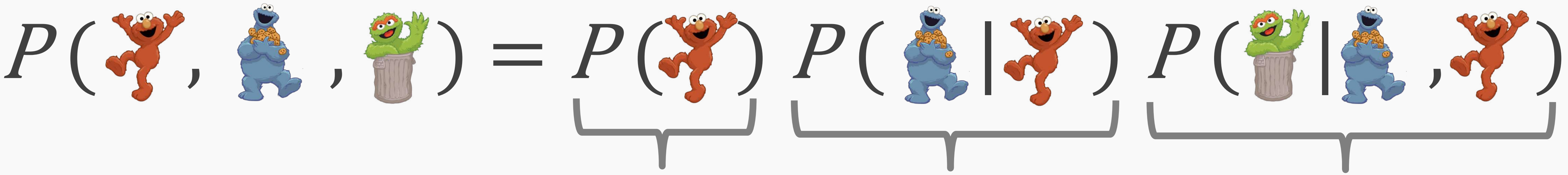
Scene 3

$P($  ,  ,  $) =$

# Example

The probability of the the next on-screen Sesame Street character can be computed as

Scene 1 Scene 2 Scene 3



Scene 1 Scene 2 Scene 3

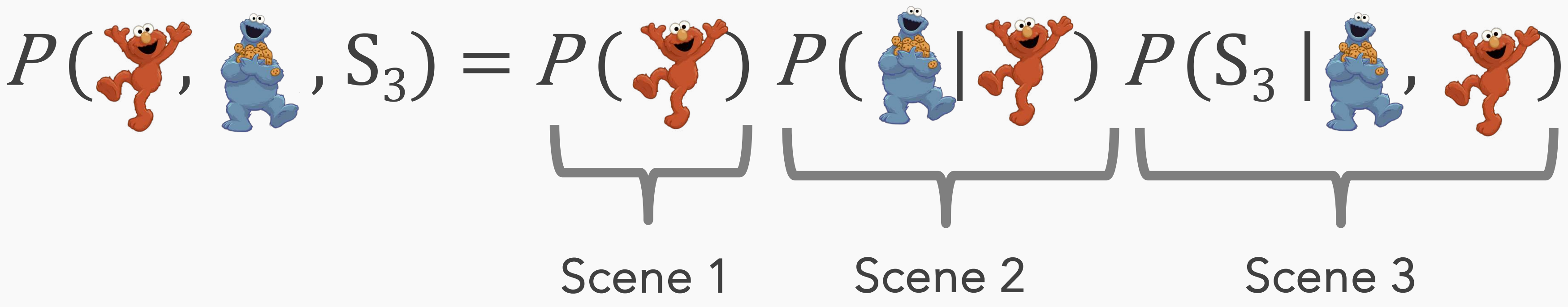
# Example

Why is it useful to accurately estimate the joint probability of any given sequence of length $N$?

# Example

Having learned a Language Model means that we know the behavior of the sequences.

If we have a sequence of length $N$, we can determine the most likely next event (i.e., sequence of length $N+1$)



# Language Modeling as a sequence of events

A Language Model estimates the probability of any sequence of words

Let $\boldsymbol{X}$ = "Shiv was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$P(\boldsymbol{X}) = P($"Shiv was late for class"$)$

# Outline

Natural Language Processing

Preprocessing

Language Modeling

- **Unigrams**
- Bigrams
- Neural Networks for Language Modeling

Evaluation

# Language Modelling: **Unigrams**

How can we build a language model?

**Naive Approach: Unigram model**

Assume each word is independent of all others

Count how often each word occurs (in the training data).

Let $\boldsymbol{X}$ = "Pavlos loves giving surprise quizzes"

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

# Language Modelling: **Unigrams**

Consider our corpus $d$ has 100,000 words.

| Word | Number of occurrences |
|---|---|
| Pavlos | 15 |
| loves | 1,000 |
| giving | 400 |
| surprise | 3,000 |
| quizzes | 350 |

$n_{w_i}$ = number of times a word $w_i$ appears in the corpus

$|W|$ = Total Number of words, 100000 (corpus size)

Let $\boldsymbol{X}$ = "Pavlos loves giving surprise quizzes"

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$P(w_i) = \frac{n_{w_i}(d)}{|W|}$$

$$P(w_1) = P(Pavlos) = \frac{n_{w_1}(d)}{|W|} = \frac{15}{100{,}000} = 0.00015$$

# Language Modelling: **Unigrams**

Consider our corpus $d$ has 100,000 words.

| Word | Number of occurrences |
|---|---|
| Pavlos | 15 |
| loves | 1,000 |
| giving | 400 |
| surprise | 3,000 |
| quizzes | 350 |

$n_{w_i}$ = number of times a word $w_i$ appears in the corpus

$|W|$ = Total Number of words, 100000 (corpus size)

Let $\boldsymbol{X}$ = "Pavlos loves giving surprise quizzes"

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

What is the dimension of $P(w_i)$?

$$P(w_i) = \frac{n_{w_i}(d)}{|W|}$$

$$P(w_2) = P(loves) = \frac{n_{w_2}(d)}{|W|} = \frac{1{,}000}{100{,}000} = 0.01$$

# Language Modelling: **Unigrams**

How can we build a language model?

**Naive Approach: Unigram model**

Assume each word is independent of all others

You calculate and store each of these probabilities from the training corpus

Let $\boldsymbol{X}$ = "Pavlos loves giving surprise quizzes"

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{Pavlos})\, P(\text{loves})\, P(\text{giving})\, P(\text{surprise})\, P(\text{quizzes})$$

$$= 0.00015 * 0.01 * 0.004 * 0.03 * 0.0035$$

$$= 6.3\text{x}10^{-13}$$

# Language Modelling: **Unigram issues**

## Context doesn't play a role at all

$P($"Pavlos loves giving surprise quizzes"$) = P($"quizzes loves giving surprise Pavlos"$)$

## Sequence generation: What's the most likely next word?

Pavlos loves giving surprise quizzes all ___

Pavlos loves giving surprise quizzes all the

Pavlos loves giving surprise quizzes all the the

## Out of vocabulary words

$P($"hungry"$) = 0$

# Language Modelling: **Unigram issues**

## Out of vocabulary words

$$P(\text{"hungry"}) = 0$$

### Solution: **Additive** Smoothing

$$P(\mathrm{w}) = \frac{n_w(d)}{|W|} = \frac{n_w(d) + \alpha}{|W| + \alpha\,|V|}$$

$\boldsymbol{\alpha}$ values are usually small: 0.5 – 0.2

|V| is the number of unique words in the training corpus – vocabulary size - including an additional token for unknown words

Whenever a word $w$ is not found in the vocabulary it is replaced with a token <UNK> representing unknown

# Language Modelling: **Unigram issues**

**Before Smoothing :**

$$P(\text{“hungry”}) = 0$$

**After Smoothing :**

$$P(\text{"UNK"}) = \frac{\alpha}{|W| + \alpha\,|V|} > 0$$

Smoothing allows probability of “UNK” token to be non-zero enabling the model to predict Out of Vocabulary words.

# Language Modelling: **Unigram issues**

**Context doesn't play a role at all**

$P($"Pavlos loves giving surprise quizzes"$) = P($"quizzes loves giving surprise Pavlos"$)$

**Sequence generation: What's the most likely next word?**

Pavlos loves giving surprise quizzes all ___

Pavlos loves giving surprise quizzes all <u>the</u>

Pavlos loves giving surprise quizzes all the <u>the</u>

**Out of vocabulary words**

$P($"hungry"$) = 0$

# Outline

Natural Language Processing

Preprocessing

Language Modeling

- Unigrams
- **Bigrams**
- Neural Networks for Language Modeling

Evaluation

# Language Modelling: **Bigrams**

How can we build a language model that uses context?

**Easiest Approach: bigram model**

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = " Pavlos loves giving surprise quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})$$

How can we build a language model that uses context?

**Easiest Approach: bigram model**

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = " Pavlos loves giving surprise quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\, P(\text{giving}|\text{loves})$

# Language Modelling: **Bigrams**

How can we build a language model that uses context?

**Easiest Approach: bigram model**

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = " Pavlos loves $\boxed{\text{giving surprise}}$ quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\, P(\text{giving}|\text{loves})\, P(\text{surprise}|\text{giving})$$

# Language Modelling: **Bigrams**

How can we build a language model that uses context?

**Easiest Approach: bigram model**

Look at *pairs* of consecutive words

Let $\boldsymbol{X}$ = " Pavlos loves giving $\boxed{\text{surprise quizzes}}$ "

probability

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\, P(\text{giving}|\text{loves})\, P(\text{surprise}|\text{giving})\, P(\text{quizzes}|\text{surprise})$$

# Language Modelling: **Bigrams**

You calculate each of these probabilities by simply counting the occurrences:

$$P(quizzes|surprise) = \frac{C(W_1W_2)}{C(W_1w)}$$

probability

Let $\boldsymbol{X}$ = " Pavlos loves giving surprise quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\, P(\text{giving}|\text{loves})\, P(\text{surprise}|\text{giving})\, P(\text{quizzes}|\text{surprise})$$

# Language Modelling: **Bigrams**

You calculate each of th ities b ng the occurrences:

Count of pair $(W_1, W_2)$

How?

$$P(quizzes|surprise) = \frac{C(W_1 W_2)}{C(W_1 w)} = \frac{C(W_1 W_2)}{C(W_1)}$$

Count of pair $(W_1, w)$,
Where w denotes any word in the corpus

probability

Let $\boldsymbol{X}$ = " Pavlos loves giving surprise quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\, P(\text{giving}|\text{loves})\, P(\text{surprise}|\text{giving})\, P(\text{quizzes}|\text{surprise})$$

# Language Modelling: **Bigrams**

How?

$$\frac{C(W_1 W_2)}{C(W_1 w)} = \frac{C(W_1 W_2)}{C(W_1)}$$

Let's suppose we have a sentence,

How much wood would a woodchuck chuck if a woodchuck could chuck wood?

C(woodchuck w) =

Any word that comes after "woodchuck"

# Language Modelling: **Bigrams**

How?

$$\frac{C(W_1W_2)}{C(W_1w)} = \frac{C(W_1W_2)}{C(W_1)}$$

Let's suppose we have a sentence,

How much wood would a woodchuck chuck if a woodchuck could chuck wood?

C(woodchuck w) = 2 C(woodchuck) = 2

# Language Modelling: **Bigrams**

You calculate each of these probabilities by simply counting the occurrences:

$$P(quizzes|surprise) = \frac{C(W_1W_2)}{C(W_1w)} = \frac{C(W_1W_2)}{C(W_1)} = \frac{Total\ number\ of\ \text{“}surprise\ quizzes\text{”}}{Total\ number\ \text{“}surprise\text{”}}$$

probability

Let $\boldsymbol{X}$ = " Pavlos loves giving surprise quizzes "

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{loves}|\text{Pavlos})\ P(\text{giving}|\text{loves})\ P(\text{surprise}|\text{giving})\ P(\text{quizzes}|\text{surprise})$$

# Language Modelling: **Bigram issues**

- When a word is out-of-vocabulary, it's given a probability of 0, causing the whole sentence or sequence to also have a probability of 0
- More context (like trigrams, 4-grams) is often desired, but sparsity is a challenge due to infrequent subsequences.
- As we expand the window size for context, storage issues arise.
- Raw counts fail to convey deep semantic relationships, such as the similarity between 'vehicle' and 'car'.

# Outline

Natural Language Processing

Preprocessing

Language Modeling

- Unigrams
- Bigrams
- Neural Networks for Language Modeling

Evaluation

# Outline

Natural Language Processing

Preprocessing

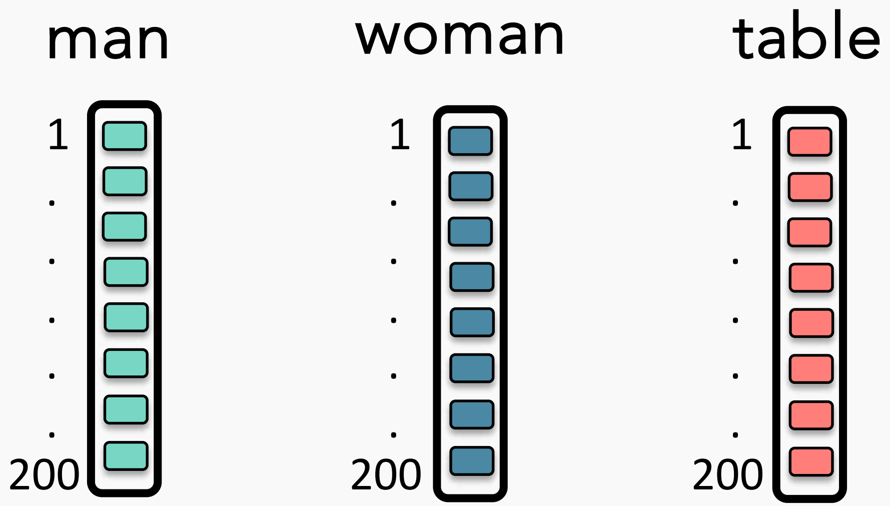
Language Modeling

- Unigrams
- Bigrams
- **Neural Networks for Language Modeling**

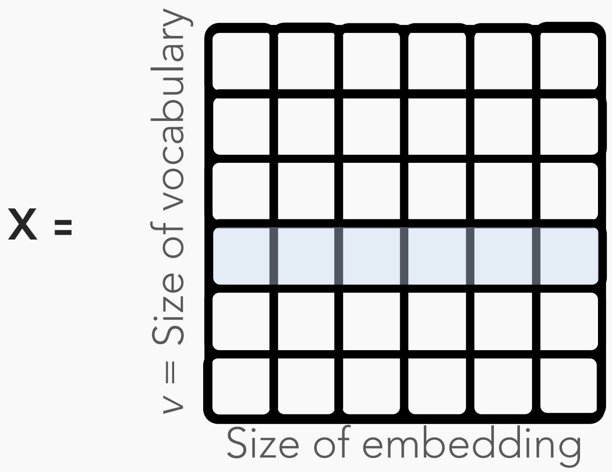
Evaluation

**IDEA:** Let's use a **neural network**!

First, each word is represented by a word embedding (e.g., vector of length 200)

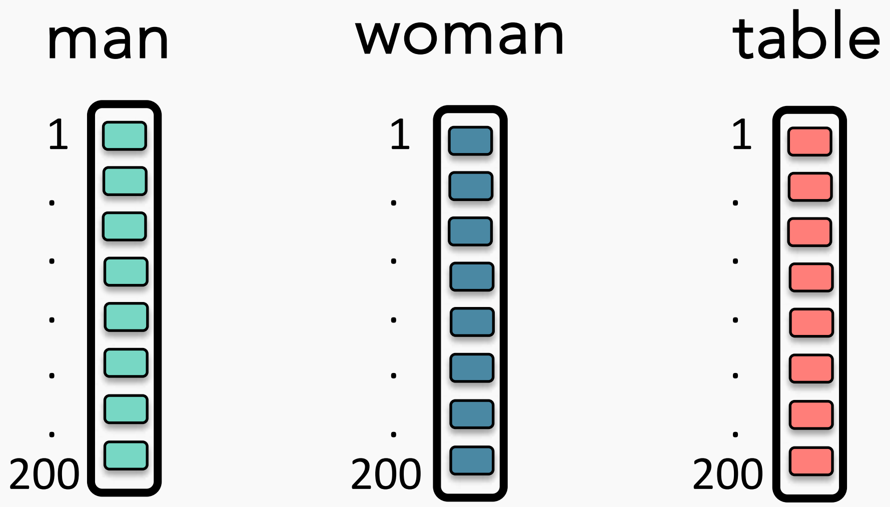



Embedding matrix

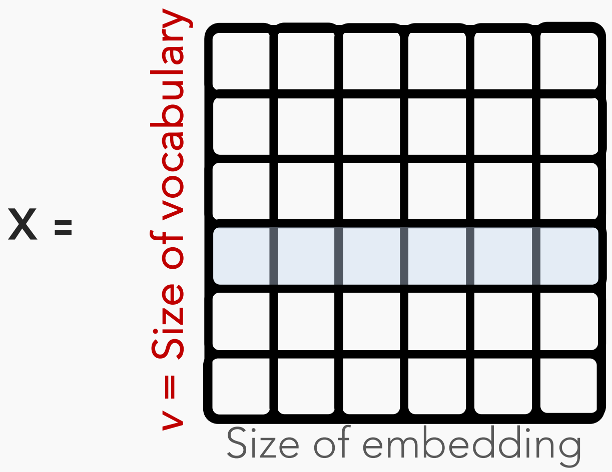


**IDEA:** Let's use a **neural network**!

First, each word is represented by a word embedding (e.g., vector of length 200)



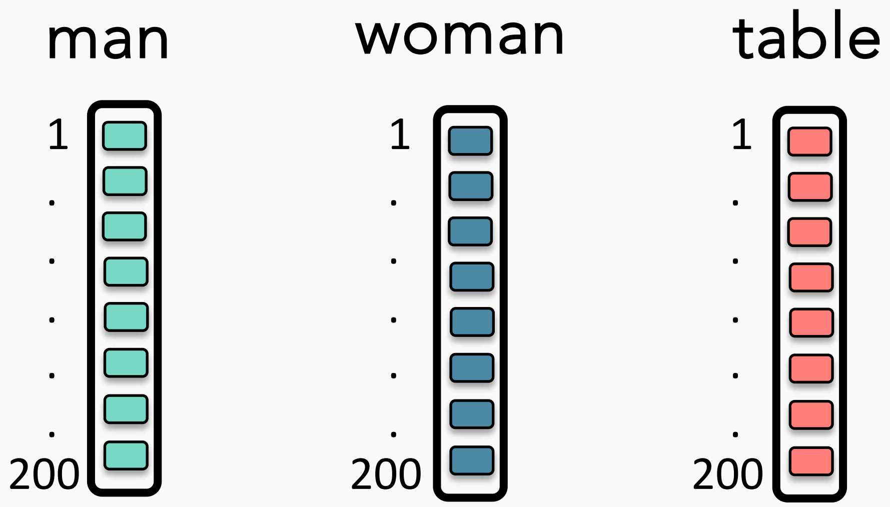

Embedding matrix



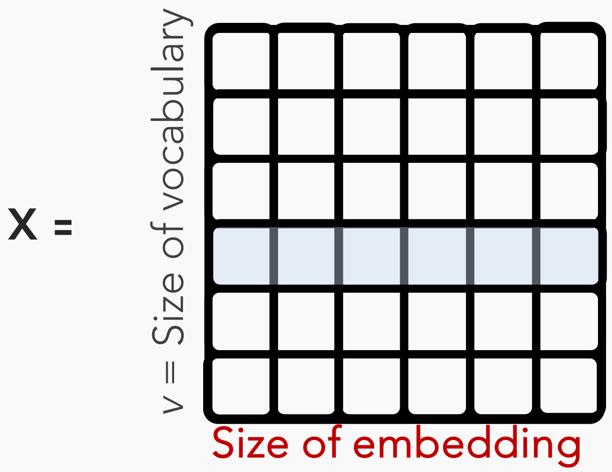
# Language Modelling: **Neural Networks**

**IDEA:** Let's use a **neural network**!

First, each word is represented by a word embedding (e.g., vector of length 200)



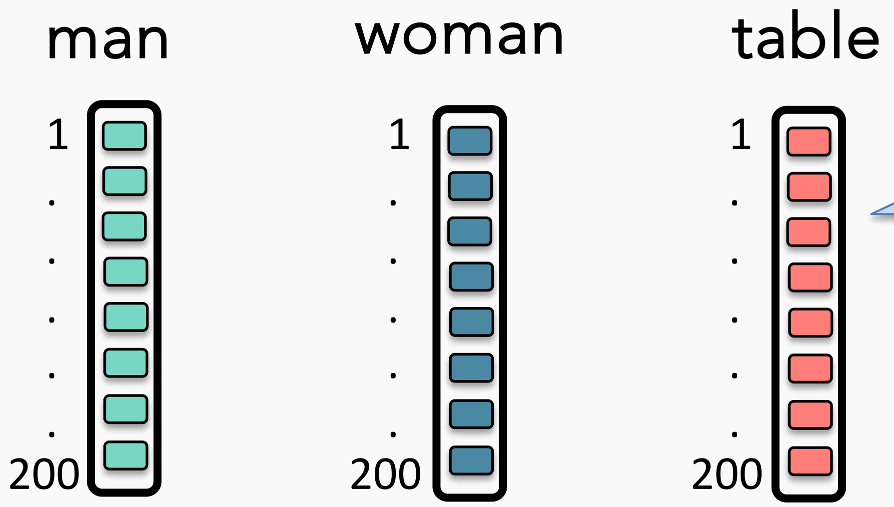

Embedding matrix



# Language Modelling: **Neural Networks**

**IDEA:** Let's use a **neural network**!

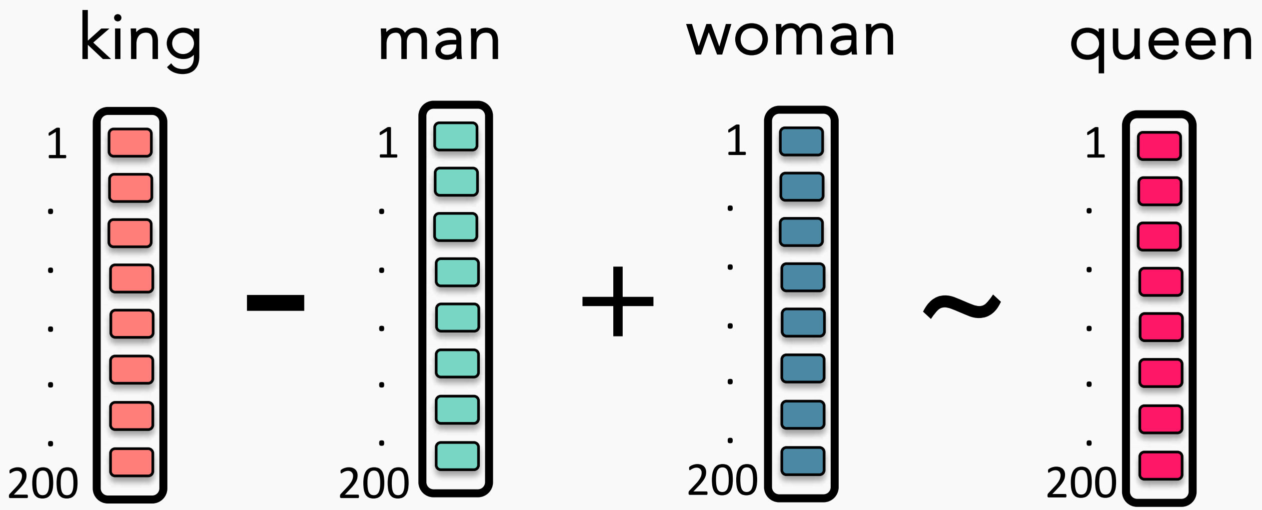
First, each word is represented by a word embedding (e.g., vector of length 200)




- Each rectangle is a *floating-point* scalar
- Words that are more *semantically similar* to one another will have **embeddings** that are also proportionally similar
- We can *use pre-existing* word embeddings that have been trained on gigantic corpora

# Word Embeddings

These word embeddings are so rich that you get nice properties:

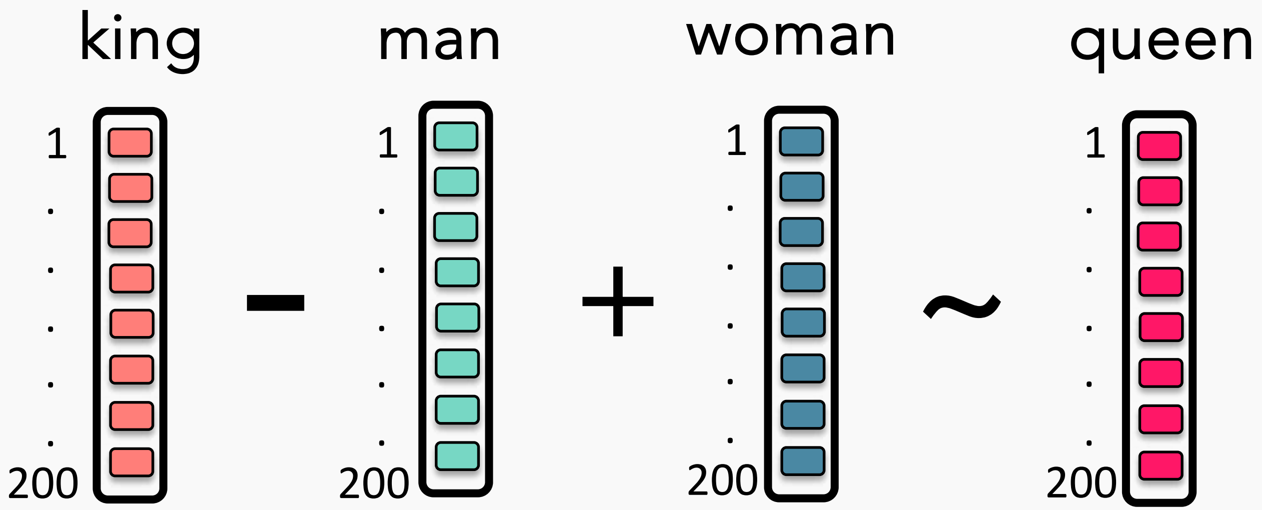





Word2vec: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
GloVe: https://www.aclweb.org/anthology/D14-1162.pdf

# Word Embeddings

These word embeddings are so rich that you get nice properties:




Word2Vec a popular word embedding see supplementary materials



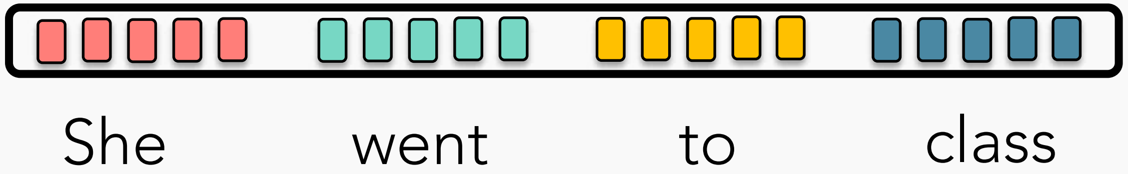
Word2vec: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
GloVe: https://www.aclweb.org/anthology/D14-1162.pdf

How can we use these embeddings to build a Language Model?

Remember, we only need a system that can estimate:

$$P(\underbrace{x_{t+1}}_{\text{next word}} | \underbrace{x_t, x_{t-1}, \dots, x_1}_{\text{previous words}})$$

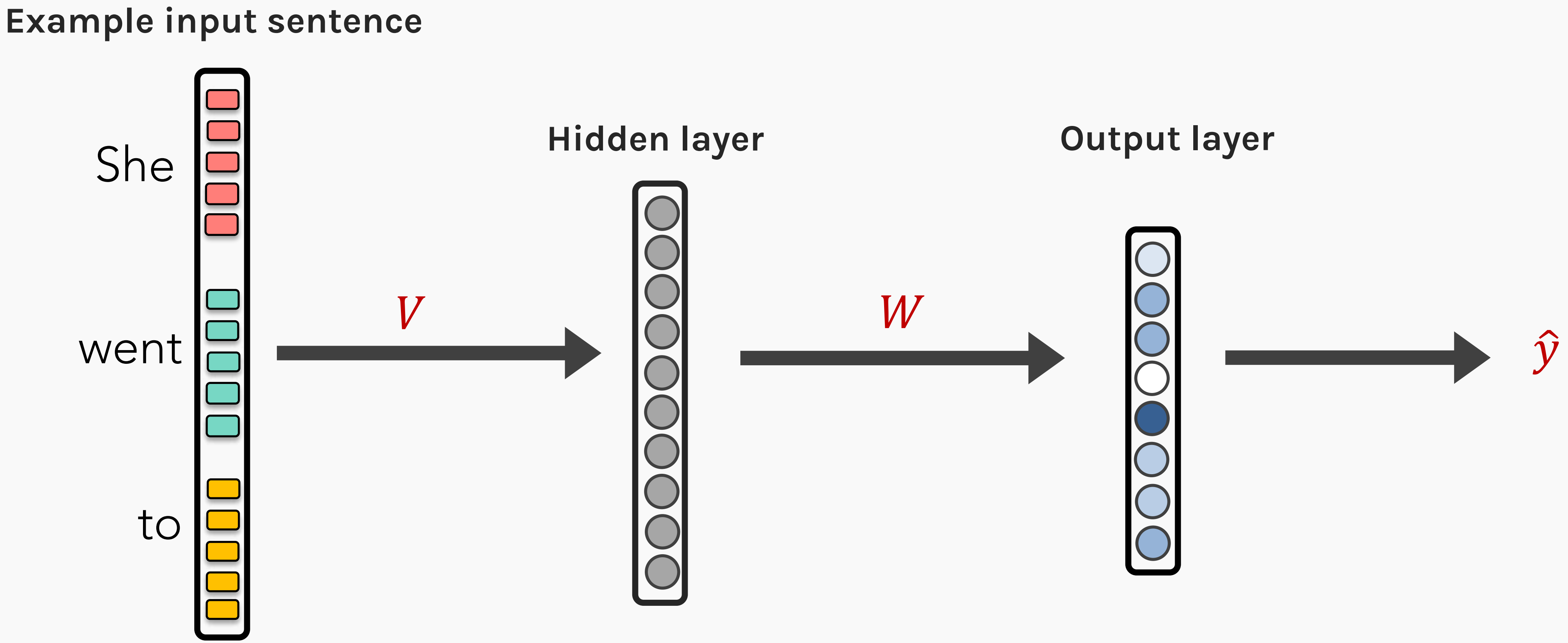
**Example input sentence**



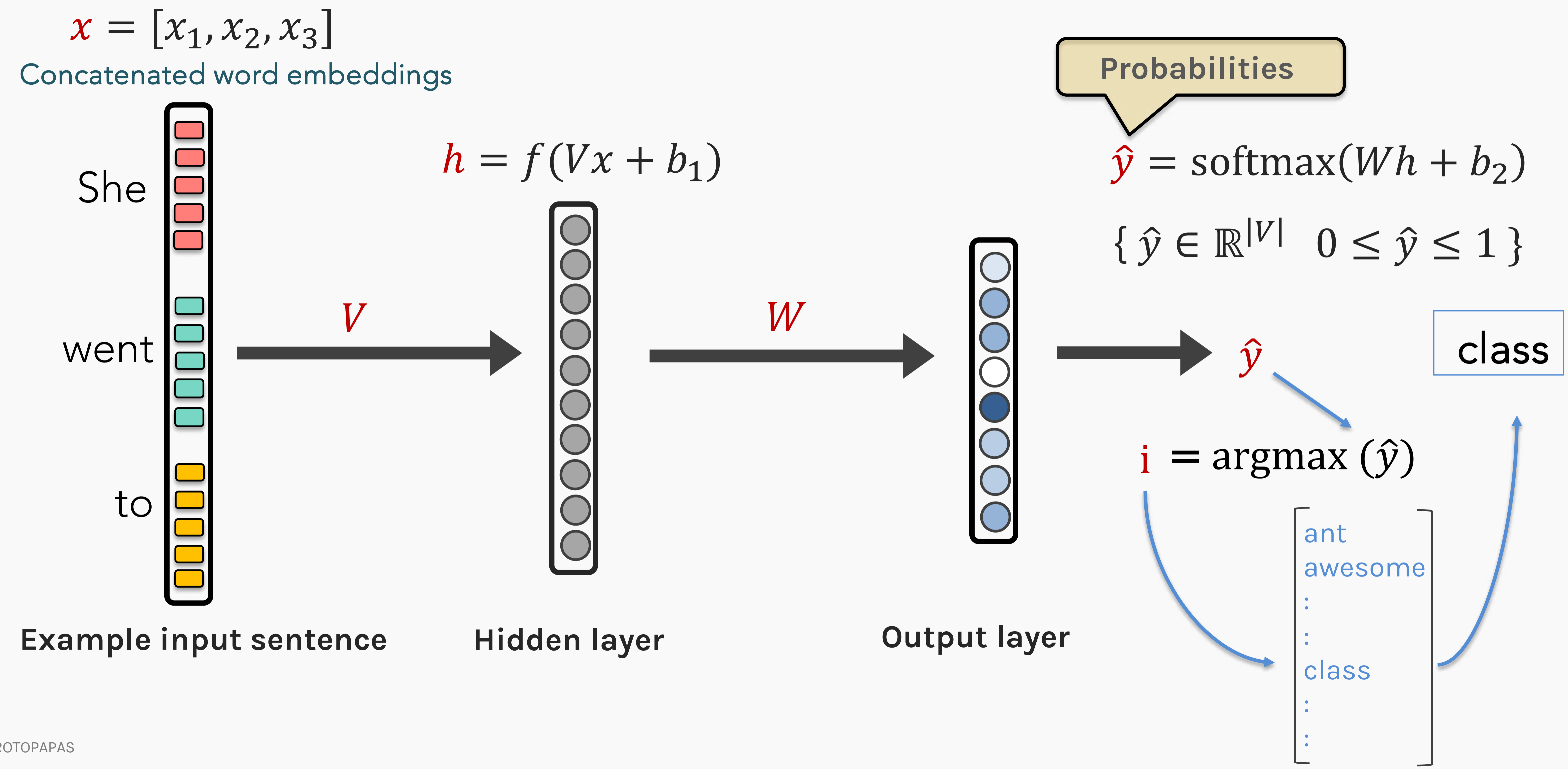
# Language Modelling: **Feed-Forward Neural Network**

**Neural Approach #1:** Feed-forward neural net

**General Idea:** using *windows* of words, predict the next word



# Language Modelling: **Feed-Forward Neural Network**



# Language Modelling: **Feed-Forward Neural Network**

$x = [x_1, x_2, x_3]$

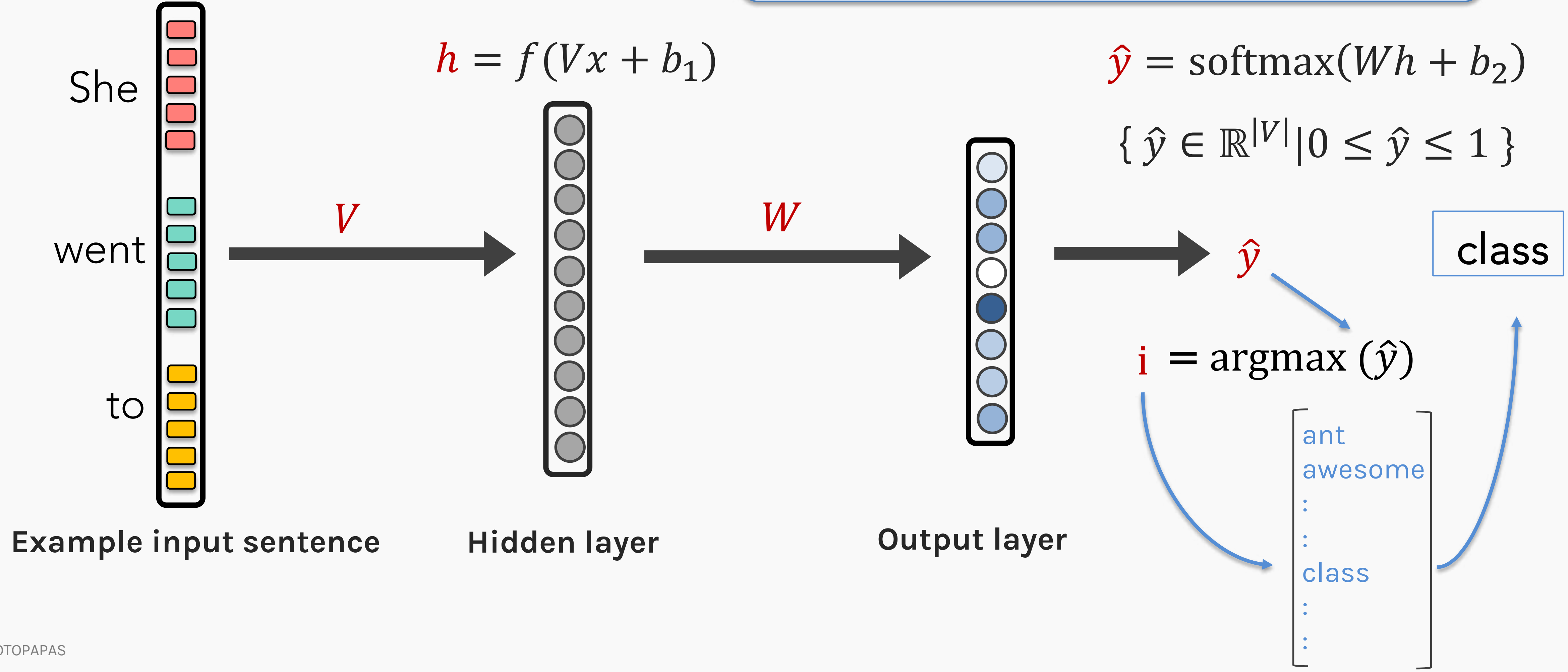
Concatenated word embeddings

She went to class

$h = f(Vx + b_1)$

$\hat{y} = \text{softmax}(Wh + b_2)$

$\{ \hat{y} \in \mathbb{R}^{|V|} | 0 \leq \hat{y} \leq 1 \}$



# Language Modelling: **Feed-Forward Neural Network**

$x = [x_2, x_3, x_4]$

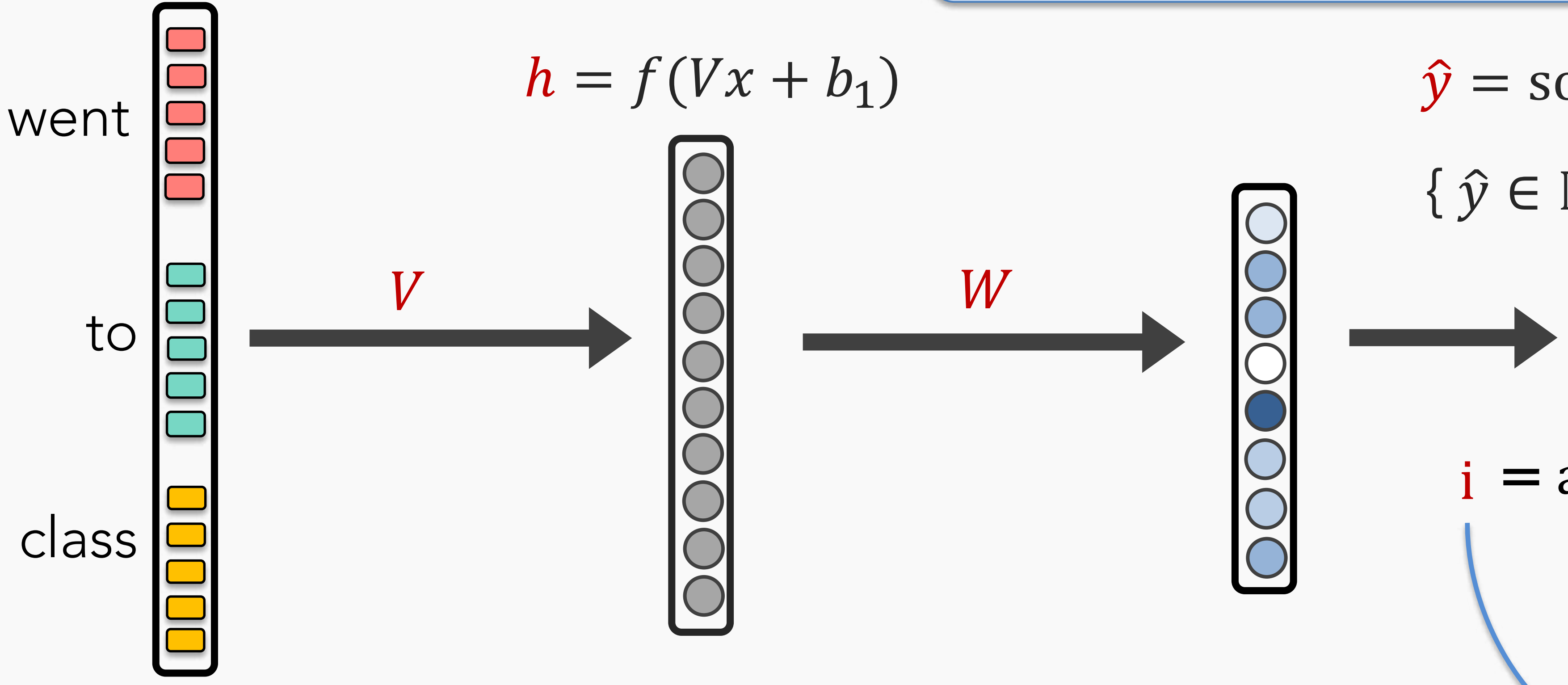
Concatenated word embeddings

She went to class after

$h = f(Vx + b_1)$

$\hat{y} = \text{softmax}(Wh + b_2)$

$\{\, \hat{y} \in \mathbb{R}^{|V|} \quad 0 \le \hat{y} \le 1 \,\}$




$\hat{y}$

after

$\text{i} = \text{argmax}\,(\hat{y})$

$\begin{bmatrix} \text{ant} \\ \text{awesome} \\ \text{after} \\ \vdots \\ \vdots \\ \text{class} \\ \vdots \end{bmatrix}$

**Example input sentence** **Hidden layer** **Output layer**

# Language Modelling: **Feed-Forward Neural Network**

$x = [x_3, x_4, x_5]$

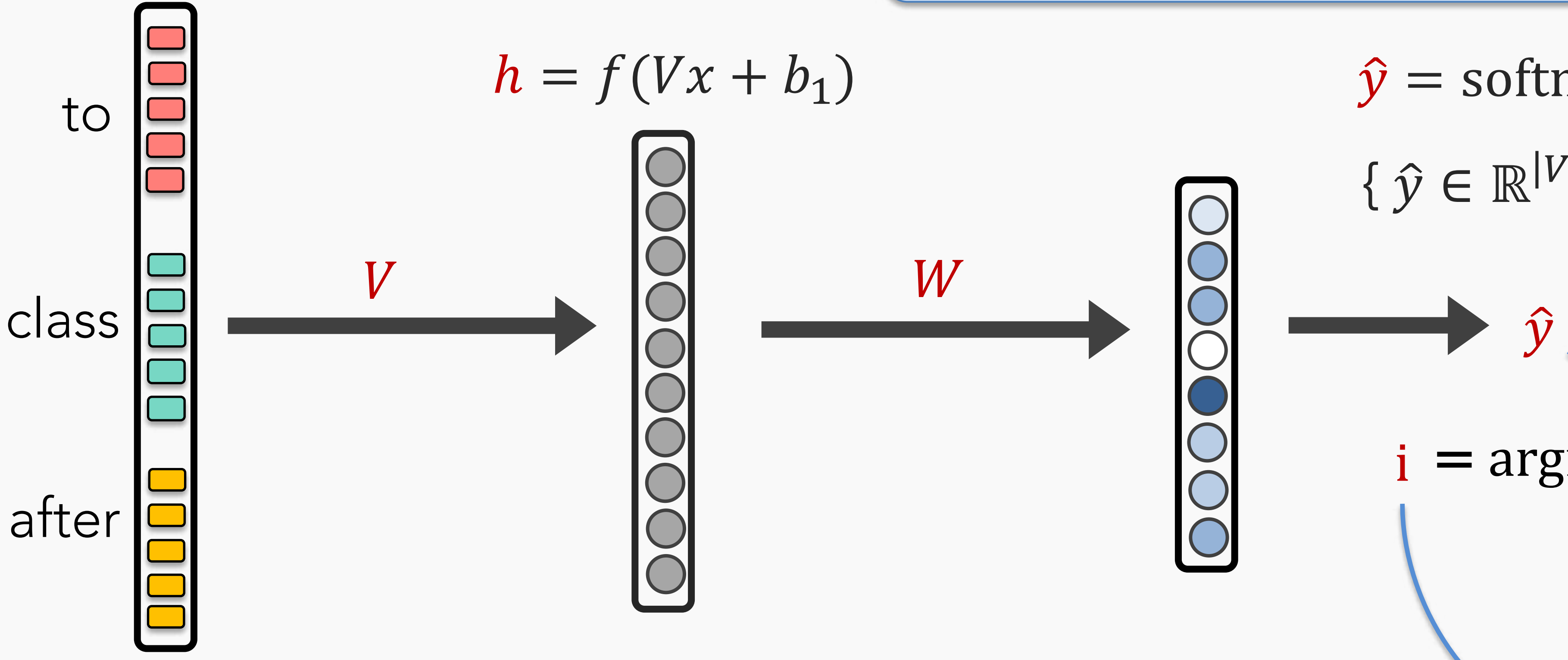
Concatenated word embeddings

She went to class after visiting

$h = f(Vx + b_1)$

$\hat{y} = \text{softmax}(Wh + b_2)$

$\{\, \hat{y} \in \mathbb{R}^{|V|} \quad 0 \leq \hat{y} \leq 1 \,\}$




visiting

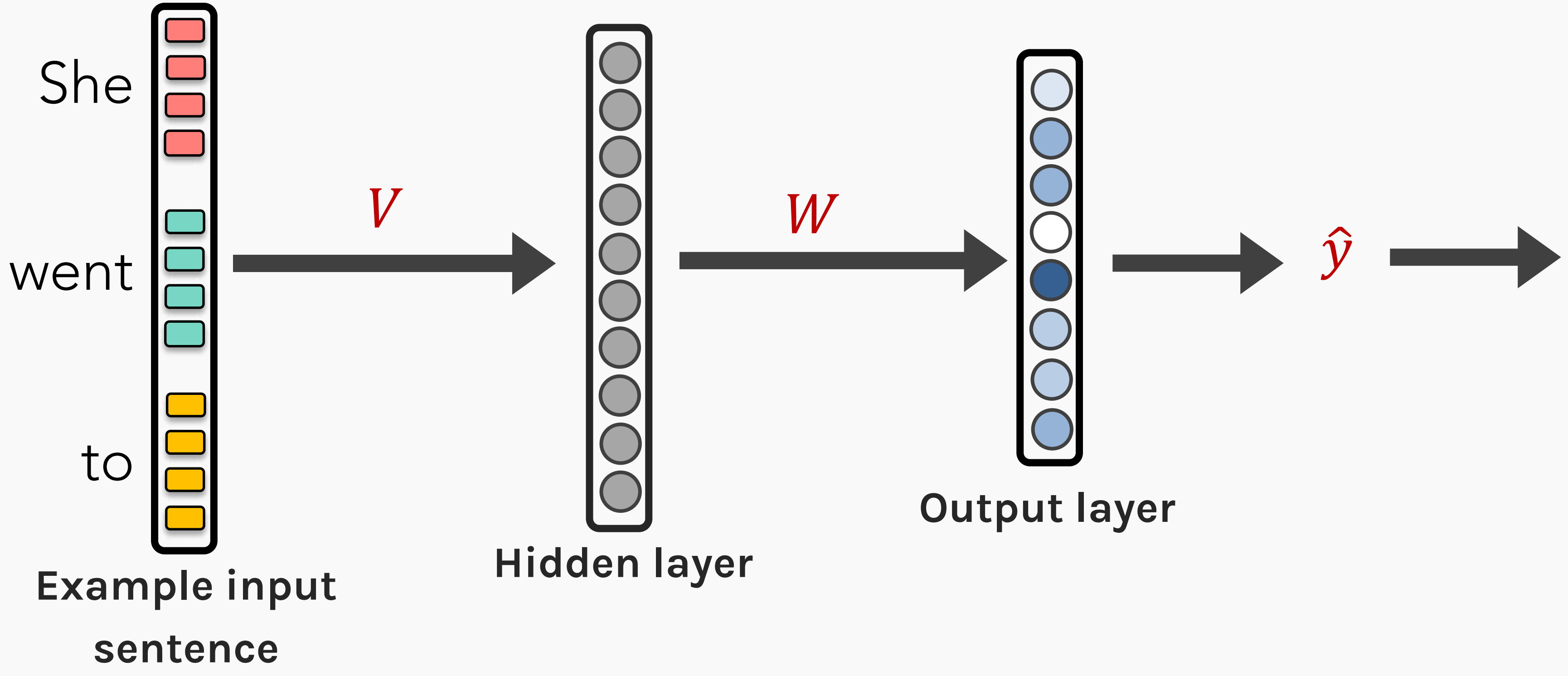
$\text{i} = \text{argmax}\,(\hat{y})$

$$\begin{bmatrix} \text{ant} \\ \text{awesome} \\ : \\ : \\ : \\ \text{class} \\ \text{visiting} \end{bmatrix}$$

**Example input sentence** **Hidden layer** **Output layer**

# Language Modelling: **Feed-Forward Neural Network** Training

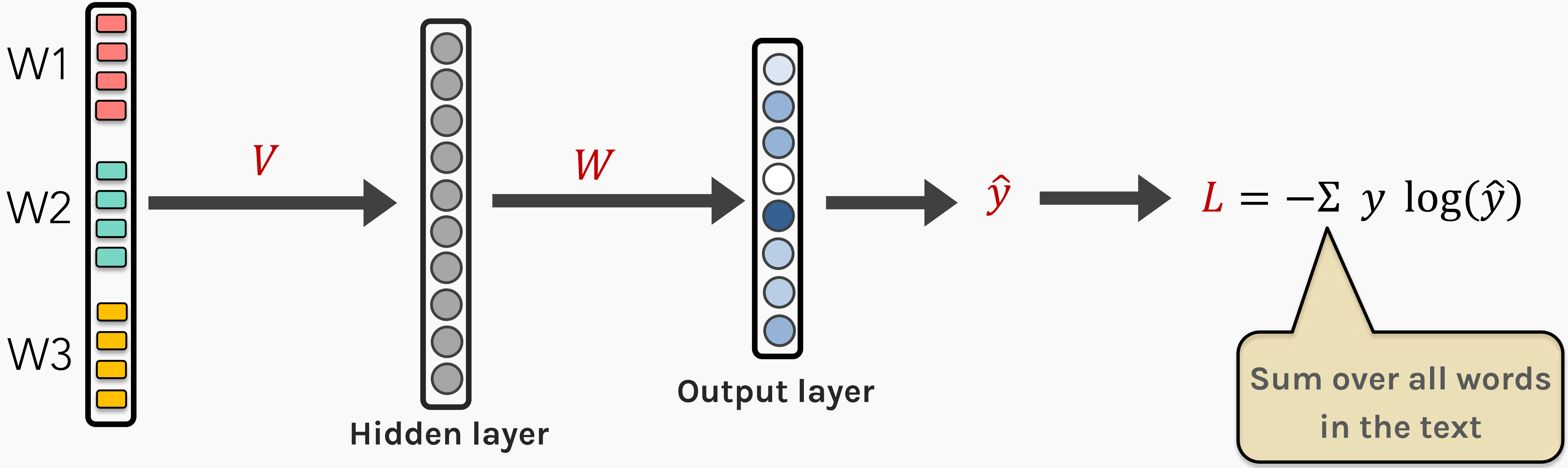
$x = [x_1, x_2, x_3]$ $\quad h = f(Vx + b_1)$ $\quad \hat{y} = \mathrm{softmax}(Wh + b_2) \in \mathbb{R}^{|V|}$




$L_1 = -\, y \, \log(\hat{y})$

# Language Modelling: **Feed-Forward Neural Network** Training

$x = [x_i, x_{i+1}, x_{i+2}]$ $\quad h = f(Vx + b_1)$ $\quad \hat{y} = \text{softmax}(Wh + b_2) \in \mathbb{R}^{|V|}$




## Back Propagation

$$V^* = V - \eta \nabla_V L$$

$$W^* = W - \eta \nabla_W L$$

# Language Modelling : **Feed-Forward Neural Network**

## FFNN Strength

- No sparsity issues (it's okay if we've never seen a word)
- No storage issues (we never store counts)

# Language Modelling : **Feed-Forward Neural Network**

## FFNN Strength

- No sparsity issues (it's okay if we've never seen a word)
- No storage issues (we never store counts)

## FFNN Issues

- Fixed-window size can never be big enough. Need more context
  - Requires inputting entire context just to predict one word
  - Increasing window size adds many more weights
- The weights awkwardly handle word position
- No concept of time

# Language Modelling

**We especially need a system that:**

- Has a concept of an “infinite” past, not just a fixed window
- For each new input, output the most likely next event (e.g., word)

# Outline

Natural Language Processing

Preprocessing

Language Modeling

- Unigrams
- Bigrams
- Neural Networks for Language Modeling

**Evaluation**

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Base Approach:** Transform everything into a classification problem.

During training, we predict a word in the vocabulary as a class.

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Base Approach:** Transform everything into a classification problem.

$$P(\, x \mid \text{the, award-winning, actor, arrived} \,) = \begin{Bmatrix} \text{action} : 0.01 \\ \text{boston} : 0.05 \\ \text{false} : 0.01 \\ \text{today} : 0.31 \\ \text{yesterday} : 0.25 \end{Bmatrix}$$

# Language Modelling: **Evaluation**

How can we evaluate a language model?

Not your usual normalization

**Formal Approach:** Perplexity

Perplexity, denoted as $PP$, is the inverse probability of a sentence in the test set "normalized" by the number of words.

It measures how confused or *perplexed* a model is against an unseen sample.

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity

Given a sentence of $N$ words

$$S = (w_1, w_2, w_3, \ldots, w_N),$$

the language model gives us the probability of the sentence

$$P(w_1, w_2, w_3, \ldots, w_N).$$

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity

We define perplexity of a sentence as:

$$\mathrm{PP(S)} = \mathrm{P}(w_1, w_2,\ w_3, \ldots, w_N)^{-\frac{1}{N}}$$

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity $\text{PP(S)} = \text{P}(w_1, w_2, w_3, \ldots, w_N)^{-\frac{1}{N}}$

A higher probability for a sentence means lower perplexity.

A low perplexity score means that the model learned.

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity

If we have language model that predicts random words, the probability of each word is independent from the rest.

This means that

$$\mathrm{P}(w_1, w_2,\ w_3, \ldots, w_N) = \mathrm{P}(w_1)P(w_2)P(w_3)\ldots P(w_N)$$

# Language Modelling: **Evaluation**

$$P(w_1, w_2, w_3, \ldots, w_N) = P(w_1)P(w_2)P(w_3)\ldots P(w_N)$$

For a random language model, the probability of each word will be the same and equals to:

$$P(w_i) = \frac{1}{|Vocabulary|} = \frac{1}{V}$$

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity

$$\mathrm{P}(w_i) = \frac{1}{|Vocabulary|} = \frac{1}{V}$$

The probability of the sentence is

$$\mathrm{P}(w_1, w_2,\ w_3, \ldots, w_N) = \mathrm{P}(w_1)P(w_2)P(w_3)\ldots\ P(w_N)$$

$$= \frac{1}{V}\frac{1}{V}\frac{1}{V}\ldots\frac{1}{V} = \left(\frac{1}{V}\right)^N$$

# Language Modelling: **Evaluation**

How can we evaluate a language model?

**Formal Approach:** Perplexity

The perplexity of the sentence is

$$PP(S) = P(w_1, w_2, w_3, \ldots, w_N)^{-\frac{1}{N}}$$

How can we evaluate a language model?

**Formal Approach:** Perplexity

The perplexity of the sentence is

$$\mathrm{PP(S)} = \mathrm{P}(w_1, w_2,\ w_3, \ldots, w_N)^{-\frac{1}{N}}$$

$$= \left(\left(\frac{1}{V}\right)^N\right)^{-\frac{1}{N}}$$

How can we evaluate a language model?

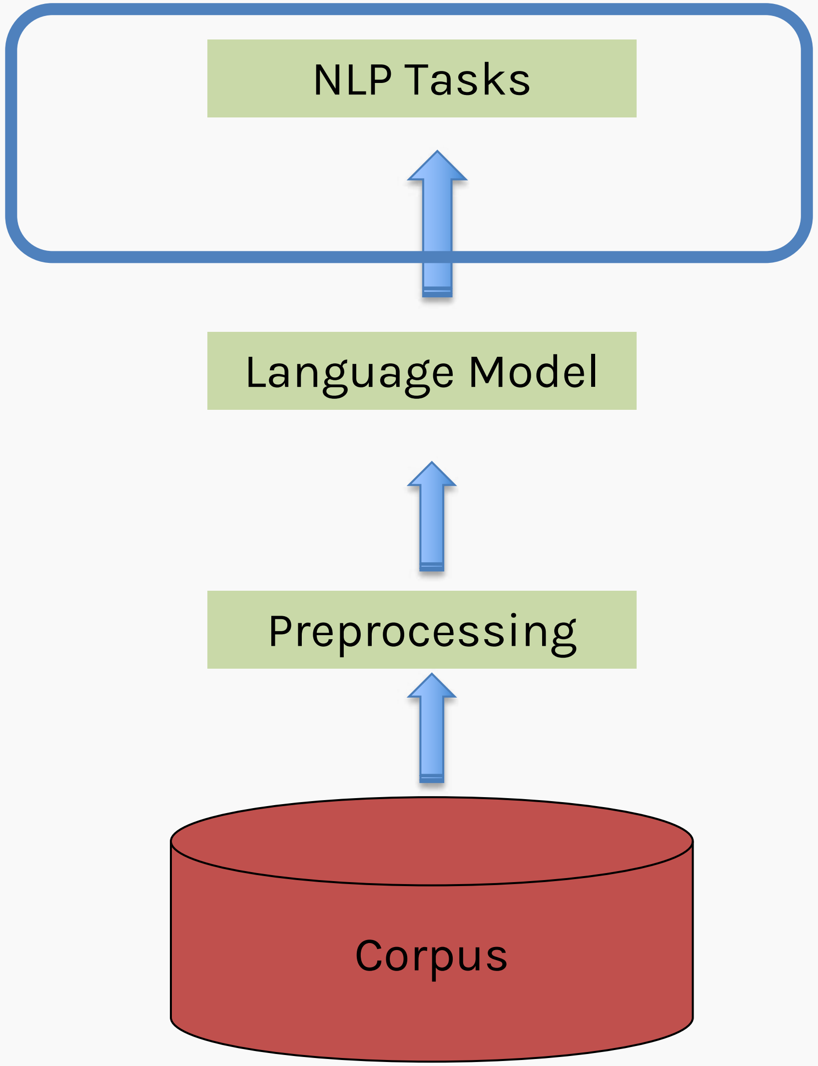
**Formal Approach:** Perplexity

The perplexity is equal to the size of the vocabulary.

$$\mathrm{PP(S)} = \left(\left(\frac{1}{V}\right)^{N}\right)^{-\frac{1}{N}} = \left(\frac{1}{V}\right)^{-1} = V$$

# Language Modelling: **Evaluation**

We can perform various tasks using a language model like sentiment analysis, question answering, NER, among others.



# Language Modelling: **Evaluation**

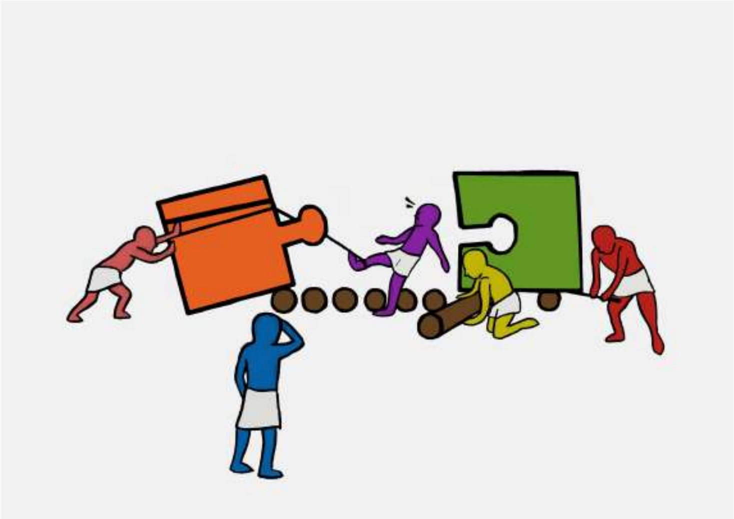
For more general language models, an entire ecosystem of benchmarks exists. They include a variety of different NLP tasks such as sentiment analysis, question answering, NER, among others.

BLEU: **Bi**-**L**ingual **E**valuation **U**nderstudy

SQuAD : **S**tanford **Qu**estion **A**nswering **D**ataset

MS MARCO: **MA**chine **R**eading **CO**mprehension **D**ataset

GLUE and SuperGLUE: **G**eneral **L**anguage **U**nderstanding **E**valuation



# Exercise: Text Classification

In this exercise, you will perform a classification task of sentiment analysis on the IMBD dataset.
Each review was labeled by a person, either positive or negative.

Your job is to define two sets of words that are likely to be present in positive and negative reviews.
Compare the number of positive vs. negative words to assign a class. Can you spot any inconsistencies?

# THANK YOU