

# NLP basics

# Language Models for Text

MVA - Speech and Language Processing #5 (NLP 1)

Paul Michel & Benoît Sagot

# Outline

- Basic yet important linguistic concepts
- Elementary processing units: characters, subwords, “words”
- Language models: what? why?
- $n$ -gram language models
- Recurrent neural language models
- Transformer-based language models
- Analysing language models
- Generative language models: BART, T5 and the GPT family
- Biases and other limitations in language models

# Processing textual data

- A text is a **sequence of characters**
  - letters, ideograms, syllabograms...
  - punctuation marks
  - whitespace characters (not in all writing systems)
- Most NLP systems rely on a **double structuring** of such a sequence
  - Macroscopic units: “**sentences**” (utterances, speech turns)
  - Microscopic units: “**words**”
- Typically, sentences are processed individually as sequences of words
- This raises **two challenges**:

# Processing textual data

- A text is a **sequence of characters**
  - letters, ideograms, syllabograms...
  - punctuation marks
  - whitespace characters (not in all writing systems)
- Most NLP systems rely on a **double structuring** of such a sequence
  - Macroscopic units: “**sentences**” (utterances, speech turns)
  - Microscopic units: “**words**”
- Typically, sentences are processed individually as sequences of words
- This raises **two challenges**:
  - How do we represent words?
  - How do we identify words and sentences in raw texts?

# Basic linguistic concepts

# What is a sentence?

- Output of a **macroscopic segmentation**
  - **Self-contained syntactic structure**
  - Semantically related to other sentences at the “**discursive**” level
    - anaphora
    - discourse relations
  - (dubious) Prosodically marked (pauses, intonation)
  - Typographically marked
    - In the Latin script, the full stop (or period) is an ambiguous, sometimes overloaded symbol
    - The same holds for the uppercase first letter of a “sentence”

# What is a sentence?

- This is an **idealised view**
  - Self-contained syntactic structures are not that frequent in speech
  - ...and do not always match the “intuitive” notion of sentence  
*“Malheur à toi si tu refuses”, le menaça-t-il.*
  - Typography is sometimes misleading  
*Best. Movie. Ever.*  
*Dès maintenant, la mobilisation est de mise. Pour l'amour des mots.*  
*The grocery sells cucumbers, lettuce, radishes, etc.*
  - Nested structures, e.g. with quotes:  
*“It is basically the perfect sort of tool to find objects like 'Oumuamua. We expect to find 100s of them with the LSST,” Dr Fraser says.*



# What is a word?

- No linguist would dare define a unique notion “word”
- At least four notions must be distinguished:
  - The prosodic word
  - The typographic word, or **token** (or **pre-token**)
  - The morphosyntactic word, or **wordform** (or **form**)
  - The **semantic word**
- Let us review the last three concepts
  - And mismatches between these three notions

# Tokens (today, sometimes “pre-tokens”)

- A token is a purely typographic unit
  - Conventionally, deterministically defined
- Starting point:
  - Many writing systems have “punctuation marks”
  - Some of them have a typographic separator (e.g. whitespace)
- In writing systems with a typographic separator, a token can be defined as:
  - A sequence of characters containing no separators and no punctuation marks,
  - or a punctuation mark (anything that is not a letter or a digit)
  - Ex.: *All of a sudden, he started playing table tennis with John Doe.*

# Tokens (today, sometimes “pre-tokens”)

- A token is a purely typographic unit
  - Conventionally, deterministically defined
- Starting point:
  - Many writing systems have “punctuation marks”
  - Some of them have a typographic separator (e.g. whitespace)
- In writing systems with a typographic separator, a token can be defined as:
  - A sequence of characters containing no separators and no punctuation marks,
  - or a punctuation mark (anything that is not a letter or a digit)
  - Ex.: *All of a sudden , he started playing table tennis with John Doe .*

# Tokens (today, sometimes “pre-tokens”)

- In writing systems without a typographic separator, the simplest way to define a token is to consider each individual character as a token
  - Remember splitting a sentence into tokens must be **deterministic** by definition
  - Ex.: 我的漢語說得不太好。  
ฉันพังไม่เข้าใจ

□ □ □ □ □



# Tokens (today, sometimes “pre-tokens”)

- In writing systems without a typographic separator, the simplest way to define a token is to consider each individual character as a token
  - Remember splitting a sentence into tokens must be **deterministic** by definition
  - Ex.: 我 的 漢 語 說 得 不 太 好 。  
ฉัน พัง ไม่ เข้า ใจ

□ □ □ □ □



# “Subwords” or “tokens”

- In some situations, we need to deal with a **finite, fixed-size vocabulary**
  - E.g. certain classes of language models
- In such situations, an “optimal” vocabulary of character sequences is extracted
  - Vocabulary items are called BPEs (“byte pair encoding”), WordPieces, SentencePieces, etc., following the name of the vocabulary extraction tool/method
    - Cf. for instance Sennrich et al. (2016), Kudo & Richardson (2018)
  - The generic terms “subword” or “token” are often used, despite their misleading surface relation with the word “word” and the confusion with the classical “token”
- In practice, subword boundaries are often represented differently whether they correspond or not to a whitespace (this information is relevant and must be preserved)
  - Ex.: *All\_of\_a\_sud den\_,\_he\_start ed\_playing\_table\_tennis\_with\_John  
Do e ..*

# Wordforms (or forms)

- A **wordform** is a syntactically atomic unit

- It can receive annotations such as a part-of-speech (noun, adjective, etc...), morphological features (plural, dative...)
- It corresponds to the leaves in syntactic structures
- Not easy to identify! Ambiguities...
- Sometimes, the token modifies the form (uppercase, errors...)

- **Mismatches** between tokens and forms are numerous:

- 1 token corresponding to multiple forms = **amalgam**  
Ex.: aux (= à les), du (= de les OR du)      won't (= will not)      Sp. dámelo (= da me lo)
- multiple tokens corresponding to 1 form = **compound word**  
Ex.: au fur et à mesure      all of a sudden
- multiple tokens corresponding to multiple forms  
Ex.: au fur et à mesure du (= au\_fur\_et\_à\_mesure\_de le)

# Named entities

- A **named entity** is a real-world object that can be denoted individually
  - Standard named entities: people, locations, organisations
  - Extended named entities: dates, addresses, URLs, e-mail addresses, numbers, etc.
- A named entity **mention** is an utterance denoting a named entity
  - Examples: Emmanuel Macron, Los Angeles, Apple Inc.
  - This denotation can be ambiguous...
- Named entity mentions have specific properties, apart from their specific, individual denotation:
  - Specific internal structure (**local grammar**), often culture-dependent more than language-dependent
    - > **They can be viewed as atomic for the grammar of the language**, and therefore as **special forms**
- Ex.: *All of a sudden , he started playing table tennis with John Doe .*

# Named entities

- A **named entity** is a real-world object that can be denoted individually
  - Standard named entities: people, locations, organisations
  - Extended named entities: dates, addresses, URLs, e-mail addresses, numbers, etc.
- A named entity **mention** is an utterance denoting a named entity
  - Examples: Emmanuel Macron, Los Angeles, Apple Inc.
  - This denotation can be ambiguous...
- Named entity mentions have specific properties, apart from their specific, individual denotation:
  - Specific internal structure (**local grammar**), often culture-dependent more than language-dependent
    - > **They can be viewed as atomic for the grammar of the language**, and therefore as **special forms**
- Ex.: *all\_of\_a\_sudden , he started playing table tennis with John\_Doe .*

# Semantic words

- A semantic word is a (sequence of) form(s) with non-compositional meaning
  - I.e. its meaning cannot be inferred from the meaning of its constitutive forms  
Ex.: *pomme de terre*, *red herring*
  - Often ambiguous!  
Ex.: *Il a sorti la pomme de terre* / *Il a modelé une pomme de terre cuite*  
*W. C. wrote how he used red herring to lay a false trail, while training hunting dogs*
  - Close but distinct from the notion of term (a conventional unit)  
Ex.: *machine à laver* (not *\*appareil à nettoyer* !)
- Ex.: *all\_of\_a\_sudden* , *he started playing table\_tennis with John\_Doe* .



# Noisy tokens?

- Discrepancies between all these “types” of “words” are a massive cause of complexity
  - One of the additional sources of complexity is language variation. Example: web texts
  - It can be helpful to correct/normalise this before (or jointly with) further processing (including wordform identification)

Phenomenon	Attested example	Std. counterpart	Gloss
<b>Ergographic phenomena</b>			
Diacritic removal	<i>demain c'est l'ete</i>	<i>demain c'est l'été</i>	'tomorrow is summer'
Phonetization	<i>je suis oqp</i>	<i>je suis occupé</i>	'I'm busy'
Simplification	<i>je sé</i>	<i>je sais</i>	'I know'
Spelling errors	<i>tous mes examen</i> <i>son normaux</i>	<i>tous mes examens</i> <i>sont normaux</i>	'All my examinations are normal'
<b>Transverse phenomena</b>			
Contraction	<i>nimp</i>	<i>n'importe quoi</i>	'rubbish'
	<i>qil</i>	<i>qu'il</i>	'that he'
Typographic diaeresis	<i>c a dire</i>	<i>c'est-à-dire</i>	'namely'
	<i>c t</i>	<i>c'était</i>	'it was'
<b>Marks of expressiveness</b>			
Punctuation transgression	<i>Joli !!!!!</i>	<i>Joli !</i>	'nice!'
Graphemic stretching	<i>superrrrrrrr</i>	<i>super</i>	'great'
Emoticons/smileyes	<i>:-), &lt;3</i>	—	—

# Introduction to Language Models

# Language Models

- Probability distribution over text utterances

$$p_{LM}(\text{"I like donuts"}) = 0.03874$$

# Language Models

- Probability distribution over text utterances

$$p_{LM}(\text{"I like donuts"}) = 0.03874$$

$$p_{LM}(\text{"The green dog eats the moon"}) = 0.00648$$

# Language Models

- Probability distribution over text utterances

$$p_{LM}(\text{"I like donuts"}) = 0.03874$$

$$p_{LM}(\text{"The green dog eats the moon"}) = 0.00648$$

$$p_{LM}(\text{"I donuts likes"}) = 0.000035413$$

# Language Models

- Probability distribution over text utterances

$$p_{LM}(\text{"I like donuts"}) = 0.03874$$

$$p_{LM}(\text{"The green dog eats the moon"}) = 0.00648$$

$$p_{LM}(\text{"I donuts likes"}) = 0.000035413$$

$$p_{LM}(\text{"J'aime les donuts"}) = 0.0$$

# Language Models

- Probability distribution over text utterances

$$p_{LM}(\text{"I like donuts"}) = 0.03874$$

$$p_{LM}(\text{"The green dog eats the moon"}) = 0.00648$$

$$p_{LM}(\text{"I donuts likes"}) = 0.000035413$$

$$p_{LM}(\text{"J'aime les donuts"}) = 0.0$$

$$p_{LM}(\text{"a6ca61 cvnon a58"}) = 0.0$$

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation

Eat your green beans!

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation

Eat your green beans!

eat	mange	1.0
your	tes	0.5
your	ton	0.5
green	vert	0.5
green	verts	0.5
beans	haricots	1.0
[...]	[...]	[...]

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation

Eat your green beans!

eat	mange	1.0
your	tes	0.5
your	ton	0.5
green	vert	0.5
green	verts	0.5
beans	haricots	1.0
[...]	[...]	[...]

Mange ton haricots verts!

Mange tes verts haricots!

Mange tes haricots verts!

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation

Eat your green beans!

eat	mange	1.0
your	tes	0.5
your	ton	0.5
green	vert	0.5
green	verts	0.5
beans	haricots	1.0
[...]	[...]	[...]

$$p_{LM}(\text{ Mange ton haricots verts! })=0.035$$

$$p_{LM}(\text{ Mange tes verts haricots! })=0.0094$$

$$p_{LM}(\text{ Mange tes haricots verts! })=0.15$$

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation

Eat your green beans!

eat	mange	1.0
your	tes	0.5
your	ton	0.5
green	vert	0.5
green	verts	0.5
beans	haricots	1.0
[...]	[...]	[...]

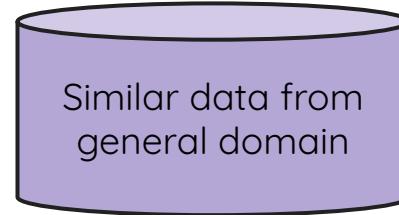
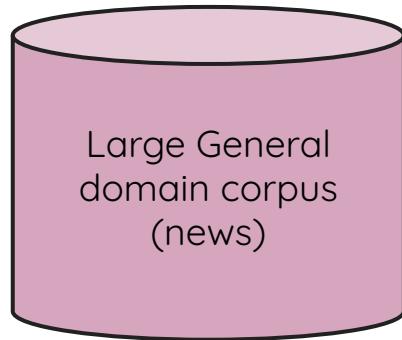
$$p_{LM}(\text{ Mange ton haricots verts! })=0.035$$

$$p_{LM}(\text{ Mange tes verts haricots! })=0.0094$$

$$p_{LM}(\text{ Mange tes haricots verts! })=0.15$$

# Why Language Models? Scoring

- “How likely is this sentence?”
  - Useful as part of the pipeline of other NLP tasks
- Speech
  - Seen in previous classes
- Machine translation
- Data filtering/selection
  - Select data from general domain for transfer learning on specific domain



# Why Language Models? Generation

- Sample from the distribution

The course on speech and natural language processing is cool because you learn how to make a sentence that can be interpreted by humans. But if I'm teaching that course to a machine, I can't tell the machine what the sentence means . A lot of the time I have to go back to a text and start from scratch or even re-create a sentence from scratch .



**Write With Transformer**

# Why Language Models? Generation

- Sample from the distribution

The course on speech and natural language processing is cool because you learn how to make a sentence that can be interpreted by humans. But if I'm teaching that course to a machine, I can't tell the machine what the sentence means . A lot of the time I have to go back to a text and start from scratch or even re-create a sentence from scratch .

- Generate text from conditional language model
  - Machine translation



Write With Transformer

Mange tes haricots verts!  $\sim p_{MT}(\cdot | \text{ Eat your green beans! } )$

# Why Language Models? Generation

- Sample from the distribution

The course on speech and natural language processing is cool because you learn how to make a sentence that can be interpreted by humans. But if I'm teaching that course to a machine, I can't tell the machine what the sentence means . A lot of the time I have to go back to a text and start from scratch or even re-create a sentence from scratch .

- Generate text from conditional language model
  - Machine translation
  - Text summarization

11 people were injured in a bridge collapse in Pittsburgh.  $\sim p_{\text{summ}}(\cdot |$  A 50-year-old bridge collapsed in Pittsburgh early Friday, requiring rescuers to rappel down a ravine and... )



Write With Transformer

# Why Language Models? Generation

- Sample from the distribution

The course on speech and natural language processing is cool because you learn how to make a sentence that can be interpreted by humans. But if I'm teaching that course to a machine, I can't tell the machine what the sentence means . A lot of the time I have to go back to a text and start from scratch or even re-create a sentence from scratch .

- Generate text from conditional language model
  - Machine translation
  - Text summarization
  - Question answering

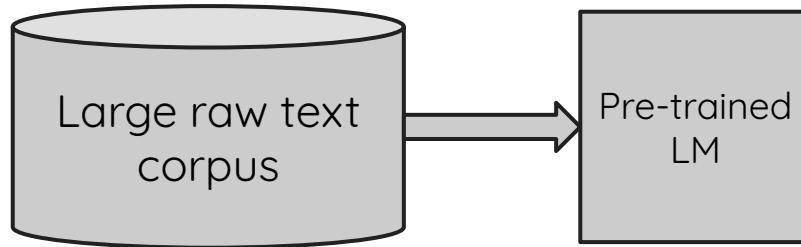


Write With Transformer

$$\text{Paris} \sim p_{\text{QA}}( \cdot | \text{What is the capital of France?} )$$

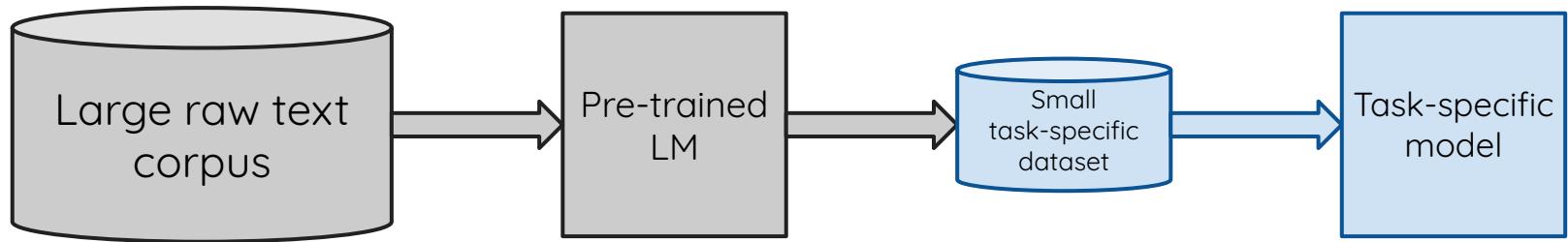
# Why Language Models? Transfer Learning

- (Pre-)Train on language modeling objective



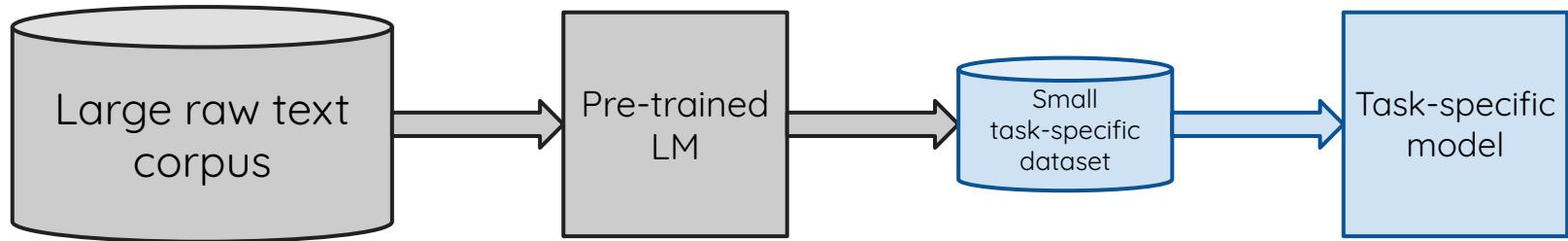
# Why Language Models? Transfer Learning

- (Pre-)Train on language modeling objective



# Why Language Models? Transfer Learning

- (Pre-)Train on language modeling objective



- Learning how to generate text requires skills that are useful for other tasks
- Language modeling requires little annotation: only raw text is needed
  - Just scrape the entire internet
- Pretrained language models are SOTA in today's NLP

# How do we Model the Distribution?

## Global normalization (EBM)?

$$p(w_1, w_2, \dots, w_n) \propto f(w_1, w_2, \dots, w_n)$$

# How do we Model the Distribution?

## Global normalization (EBM)?

$$p(w_1, w_2, \dots, w_n) \propto f(w_1, w_2, \dots, w_n)$$

- Normalizer is intractable
  - Need to sum over all possible sentences

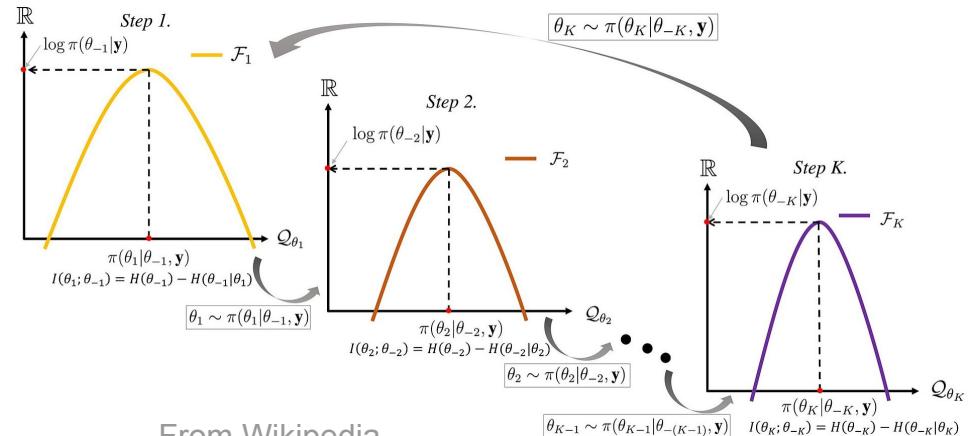
$$Z = \sum_s f(s)$$

# How do we Model the Distribution?

## Global normalization (EBM)?

$$p(w_1, w_2, \dots, w_n) \propto f(w_1, w_2, \dots, w_n)$$

- Normalizer is intractable
  - Need to sum over all possible sentences
- Sampling is hard
  - Gibbs sampling



From Wikipedia

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2|w_1) \dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2 | w_1) \dots p(w_n | w_1, w_2, \dots, w_{n-1})$$

- Generate with ancestral sampling

$$\text{“like”} \sim p_{LM}(\cdot | \text{“I”})$$

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2 | w_1) \dots p(w_n | w_1, w_2, \dots, w_{n-1})$$

- Generate with ancestral sampling

$$\text{"like"} \sim p_{LM}(\cdot | \text{"I"})$$

$$\text{"donuts"} \sim p_{LM}(\cdot | \text{"I like"})$$

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2 | w_1) \dots p(w_n | w_1, w_2, \dots, w_{n-1})$$

- Generate with ancestral sampling

$$\text{"like"} \sim p_{LM}(\cdot | \text{"I"})$$

$$\text{"donuts"} \sim p_{LM}(\cdot | \text{"I like"})$$

$$\text{"!"} \sim p_{LM}(\cdot | \text{"I like donuts"})$$

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2|w_1) \dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

- Generate with ancestral sampling

$$\text{"like"} \sim p_{LM}(\cdot | \text{"I"})$$

$$\text{"donuts"} \sim p_{LM}(\cdot | \text{"I like"})$$

$$\text{"!"} \sim p_{LM}(\cdot | \text{"I like donuts"})$$

- Parameterize the next-word probability

$$p(w_i|w_1, w_2, \dots, w_{i-1}) \propto f(w_1, w_2, \dots, w_i)$$

# How do we Model the Distribution?

## Locally normalized models

- Decompose probability of sentence as sequence of next-word predictions

$$p(w_1, w_2, \dots, w_n) = p(w_1) p(w_2|w_1) \dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

- Generate with ancestral sampling

$$\text{"like"} \sim p_{LM}(\cdot | \text{"I"})$$

$$\text{"donuts"} \sim p_{LM}(\cdot | \text{"I like"})$$

$$\text{"!"} \sim p_{LM}(\cdot | \text{"I like donuts"})$$

- Parameterize the next-word probability

$$p(w_i|w_1, w_2, \dots, w_{i-1}) \propto f(w_1, w_2, \dots, w_i)$$

$$Z = \sum_w f(w_1, \dots, w_{i-1}, w)$$

# Training Language Models

Train to maximize expected log likelihood on dataset

- Collection of sentences, documents

- Wikipedia
- Web crawls
- Books

= Gold dollar =

The gold dollar or gold one @-@ dollar piece was a coin struck as a regular

# Training Language Models

Train to maximize expected log likelihood on dataset

- Collection of sentences, documents
  - Wikipedia = Gold dollar =
  - Web crawls
  - Books The gold dollar or gold one @-@ dollar piece was a coin struck as a regular
- Calculate log likelihood of sequences under model

$$\log p_{\theta}(\text{"I like donuts"}) = \log p_{\theta}(\text{"donuts" | "I like"}) + \log p_{\theta}(\text{"like" | "I"}) + \log p_{\theta}(\text{"I" | "[SOS]"}))$$

# Training Language Models

Train to maximize expected log likelihood on dataset

- Collection of sentences, documents

- Wikipedia = Gold dollar =
- Web crawls
- Books

The gold dollar or gold one @-@ dollar piece was a coin struck as a regular

- Calculate log likelihood of sequences under model

$$\log p_{\theta}("I \text{ like donuts}") = \log p_{\theta}("donuts" | "I \text{ like"}) + \log p_{\theta}("like" | "I") + \log p_{\theta}("I" | "[SOS]")$$

- Minimize NLL over training dataset wrt. parameters

- Gradient descent or exact computation

$$L(\theta) = \sum_{s \in \text{dataset}} -\log p_{\theta}(s)$$

# Evaluating Language Models: Perplexity

- Formally: exponentiated cross-entropy per word

$$\exp \left[ \frac{1}{n} \sum_{i=1}^n -\log p(w_i | w_{i-1}, \dots, w_0) \right]$$

- Intuitively
  - How often do you need to guess a word before you make the correct prediction

# Evaluating Language Models: Perplexity

- Formally: exponentiated cross-entropy per word

$$\exp \left[ \frac{1}{n} \sum_{i=1}^n -\log p(w_i | w_{i-1}, \dots, w_0) \right]$$

- Intuitively
  - How often do you need to guess a word before you make the correct prediction
- Used to compare language modeling architectures
  - Lower is better
- Dependent on vocabulary size, tokenization, etc...
  - Beware!

# Extrinsic Evaluation of Language Models

- Task specific metric
  - Generate from (conditional) LM, then compare to ground truth
  - BLEU, METEOR, ROUGE, etc...

★ Guidelines ▾

Please use the below guidelines to give your rate:

• **1-10 Completely incorrect**

The translation is completely incorrect and inaccurate. It bears no resemblance to the source string and there are major grammar mistakes and/or typos, or the meaning is impossible to understand.

• **11-30 Overall meaning is different / lost**

The translation has a few correct keywords from the source string, but the overall meaning is different, or lost, or impossible to understand.

• **31-50 Major grammar mistakes / major typos**

The translation represents parts of the source string, but does not cover the overall meaning, or there are major grammar mistakes and/or major typos.

• **51-70 Understandable, with few translation errors / grammar mistakes**

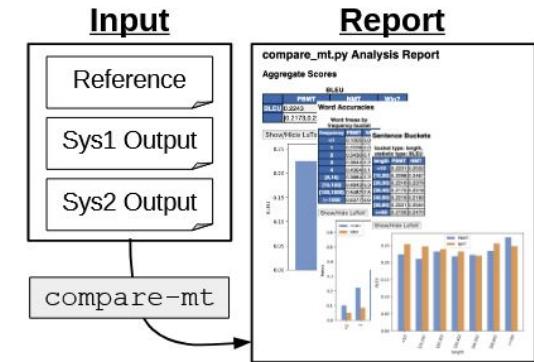
The translation is understandable and represents the overall meaning of source string. There are a few translation errors, grammar mistakes, and/or major typos.

• **71-90 Very good translation, only a couple of minor mistakes**

The translation renders the source string in the target language very closely. Only a couple of minor grammar mistakes and/or minor typos are present.

• **91-100 Accurate translation, no mistakes**

The translation renders the source string in the target language accurately. There are no mistakes.



# Extrinsic Evaluation of Language Models

- Task specific metric
  - Generate from (conditional) LM, then compare to ground truth
  - BLEU, METEOR, ROUGE, etc...

- Downstream evaluation in transfer learning
  - Leaderboards

The image shows two views of the SuperGLUE leaderboards. The top view is a mobile phone screen displaying the SuperGLUE logo and a list of top models. The bottom view is a larger desktop version of the same page.

**Mobile View (Top Left):**

Rank Name	Model
1 JDExplore d-team	Vega v1
2 Microsoft Alexander v-team	Turing NLP v5
3 ERNIE Team - Baidu	ERNIE
4 Zirui Wang	CLEVER

**Desktop View (Bottom Right):**

Leaderboard Version: 2.0	
RankName	Model
1 Liam Fedus	SS-MoE
2 Microsoft Alexander v-team	Turing NLP v5
3 ERNIE Team - Baidu	ERNIE 3.0
4 Zirui Wang	T5 + UDG, Single Model (Google Brain)
5 DeBERTa Team - Microsoft	DeBERTa / TuringNLVRv4

# *n*-gram Language Models

# N-gram Models

- Markov assumption
  - Only condition on k previous words

$$p_{LM}( w_n | w_{n-1}, w_{n-2}, \dots, w_3, w_2, w_1 ) = p_{LM}( w_n | w_{n-1}, \dots, w_{n-k} )$$

# N-gram Models

- Markov assumption
  - Only condition on k previous words

$$p_{LM}( w_n | w_{n-1}, w_{n-2}, \dots, w_3, w_2, w_1 ) = p_{LM}( w_n | w_{n-1}, \dots, w_{n-k} )$$

$$p_{LM}( \text{"France"} | \text{"Paris is the capital of"} ) = p_{LM}( \text{"France"} | \text{"capital of"} )$$

# N-gram Models

- Markov assumption
  - Only condition on k previous words

$$p_{LM}( w_n | w_{n-1}, w_{n-2}, \dots, w_3, w_2, w_1 ) = p_{LM}( w_n | w_{n-1}, \dots, w_{n-k} )$$

$$p_{LM}( \text{"France"} | \text{"Paris is the capital of"} ) = p_{LM}( \text{"France"} | \text{"capital of"} )$$

- “Tabular” learning
  - One parameter per n-gram
  - MLE has analytical solution!

$N_{\text{"I"}, \text{"like"},}$

$$p_{LM}( \text{"donuts"} | \text{"I", "like"} ) = \frac{N_{\text{"donuts"}, \text{"I"}, \text{"like"},}}{N_{\text{"I"}, \text{"like"},}}$$

# N-gram Models: Pros

- Super fast
  - One pass over training data

$$N_{\text{"I", "like", "donuts"}} = 2$$

$$N_{\text{"I", "like", "dogs"}} = 10$$

$$N_{\text{"I", "like", "running"}} = 8$$

# N-gram Models: Pros

- Super fast
  - One pass over training data
- Interpretable
  - Model parameters = frequencies

$$p_{LM}(\text{"donuts"} \mid \text{"I", "like"}) = 0.1$$

# N-gram Models: Pros

- Super fast
  - One pass over training data
- Interpretable
  - Model parameters = frequencies
- Useful for analysis, etc..

$$p_{LM}(\text{"donuts"} \mid \text{"I", "like"}) = 0.1$$

## N-gram Models: Cons

- Natural language requires longer context

“Alice went to the veterinary to pick up [her/his] dog”

## N-gram Models: Cons

- Natural language requires longer context

“**Alice** went to the veterinary to pick up [**her/his**] dog”

## N-gram Models: Cons

- Natural language requires longer context

“**Alice** went to the veterinary to pick up [**her/his**] dog”



## N-gram Models: Cons

- Natural language requires longer context

“**Alice** went to the veterinary to pick up [**her/his**] dog”

- Scales poorly with context window
  - Number of parameters grows exponentially
  - Sparsity issues...

# Recurrent Neural Language Models

# Enter Neural Networks

- Parameterize next word probability with neural network

$$p_{LM}(w_n | w_{n-1}, w_{n-2}, \dots, w_1) \propto f(w_n, w_{n-1}, w_{n-2}, \dots, w_1)$$

# Enter Neural Networks

- Parameterize next word probability with neural network

$$p_{LM}(w_n | w_{n-1}, w_{n-2}, \dots, w_1) \propto f(w_n, w_{n-1}, w_{n-2}, \dots, w_1)$$

$$\log p_{LM}(w_n | w_{n-1}, w_{n-2}, \dots, w_1) \propto e[w_n]^T h(w_{n-1}, w_{n-2}, \dots, w_1)$$



Next word embedding      Context encoder  
(neural network)

# Recurrent Neural Networks

“[SOS]”

# Recurrent Neural Networks

Input word embeddings  $w_1$   
“[SOS]”



# Recurrent Neural Networks

Initial  
hidden state

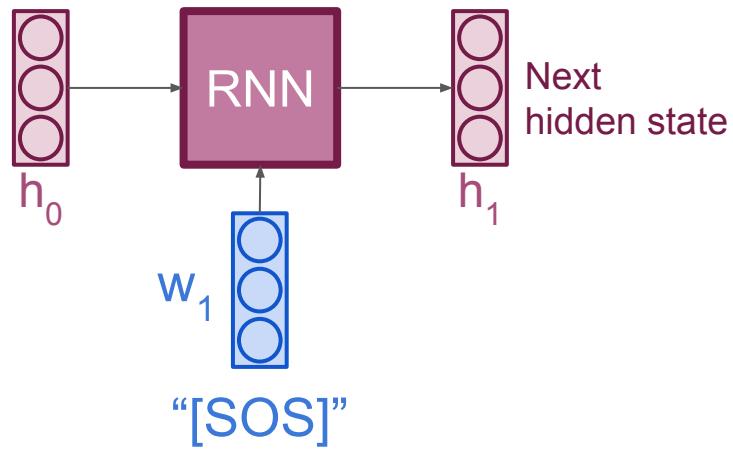


Input word  
embeddings  $w_1$

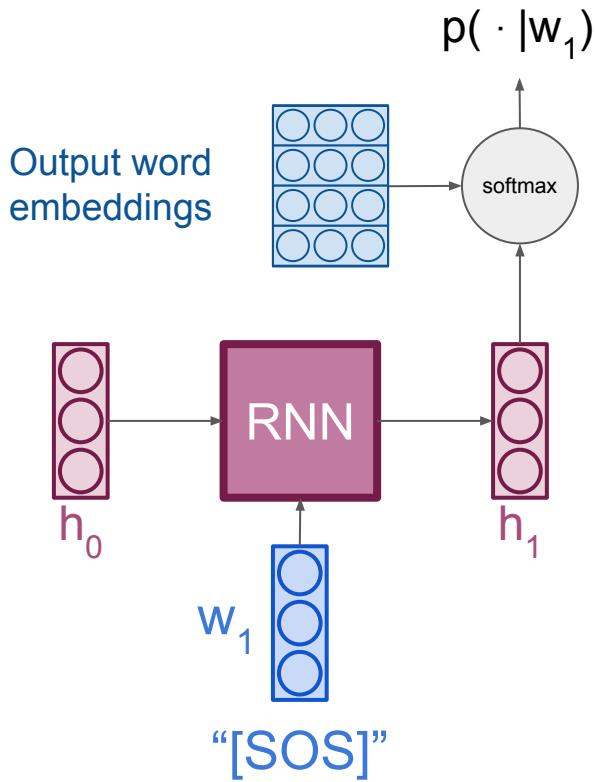


“[SOS]”

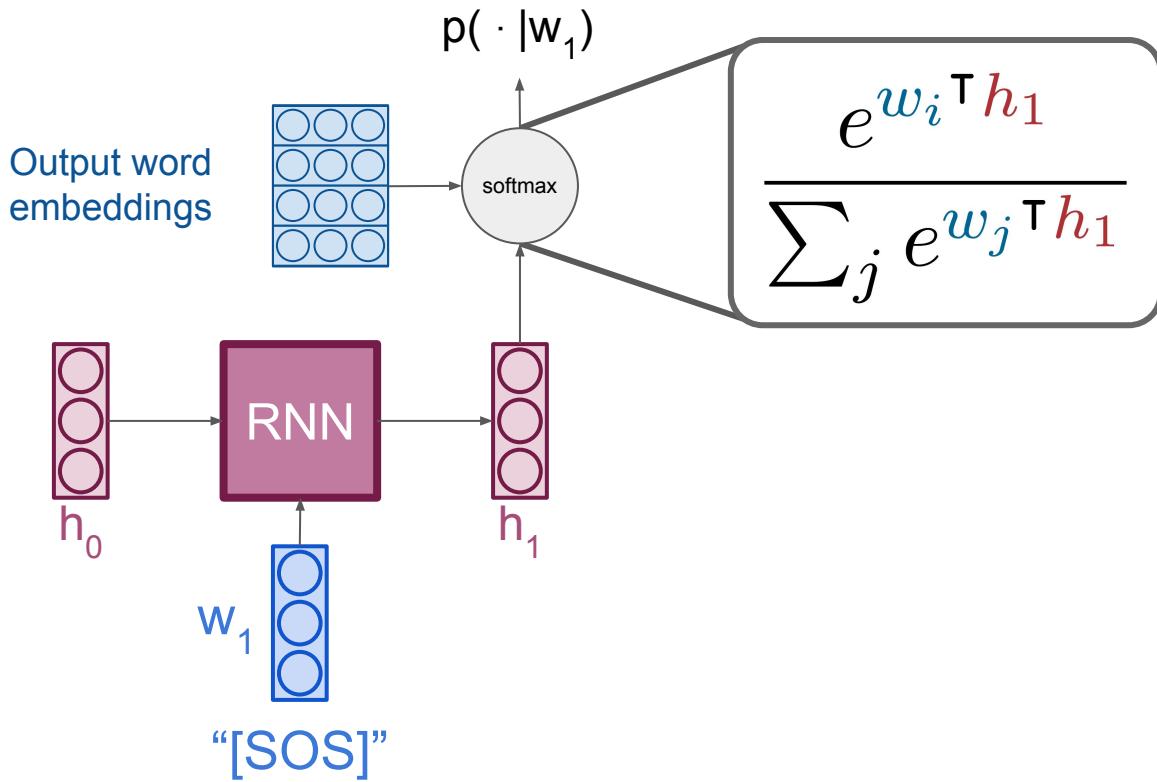
# Recurrent Neural Networks



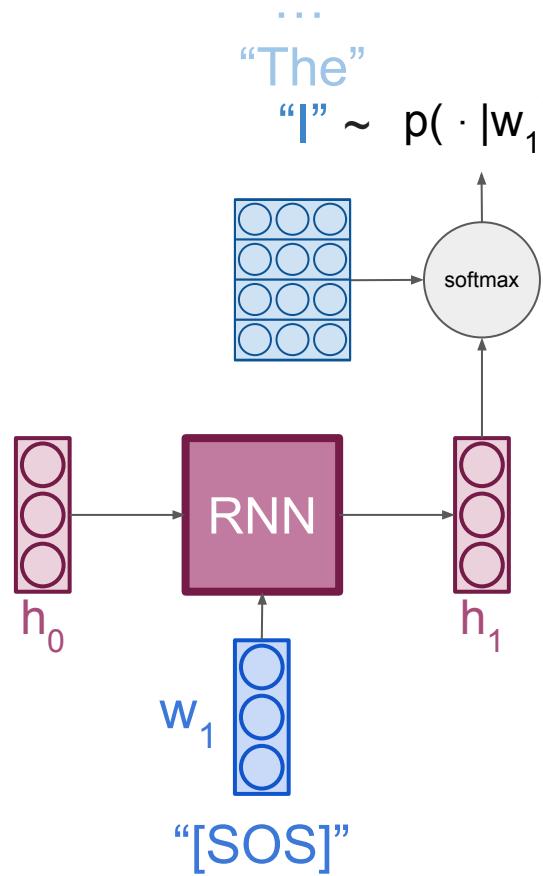
# Recurrent Neural Networks



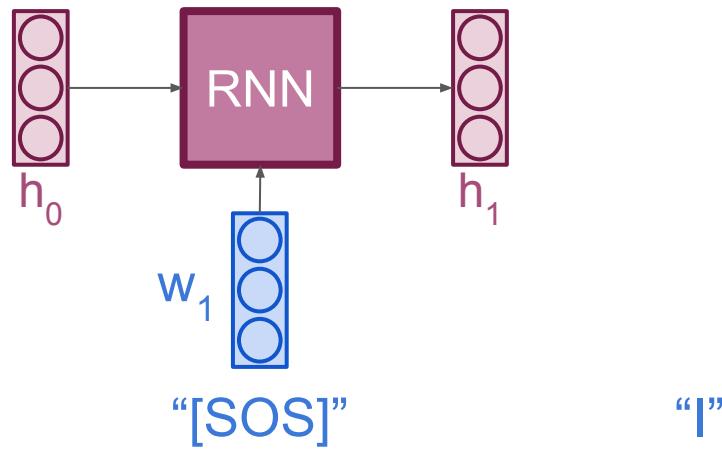
# Recurrent Neural Networks



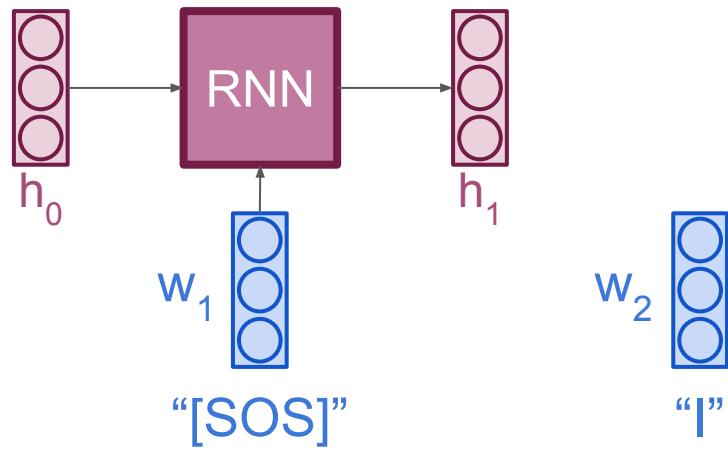
# Recurrent Neural Networks



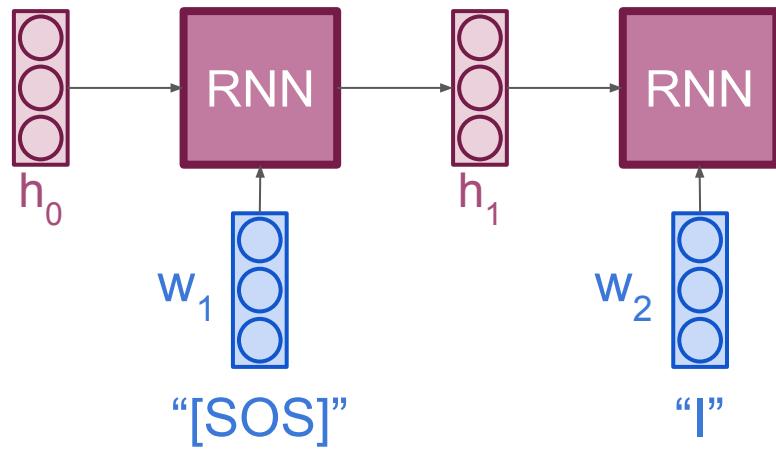
# Recurrent Neural Networks



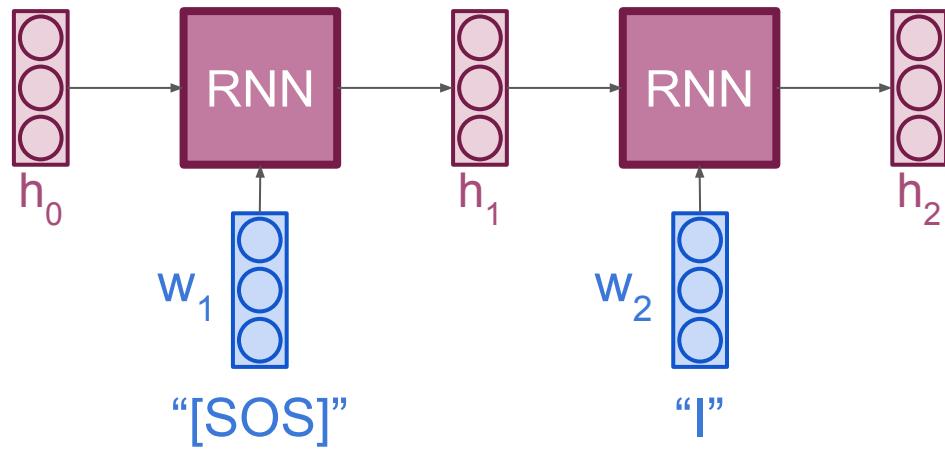
# Recurrent Neural Networks



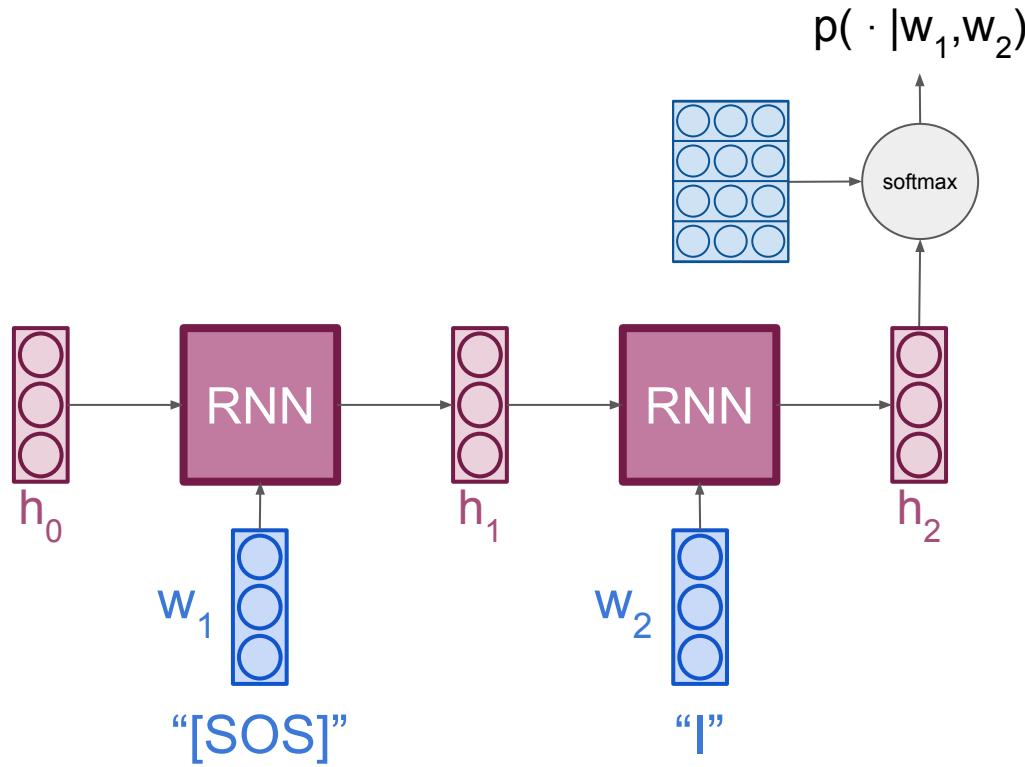
# Recurrent Neural Networks



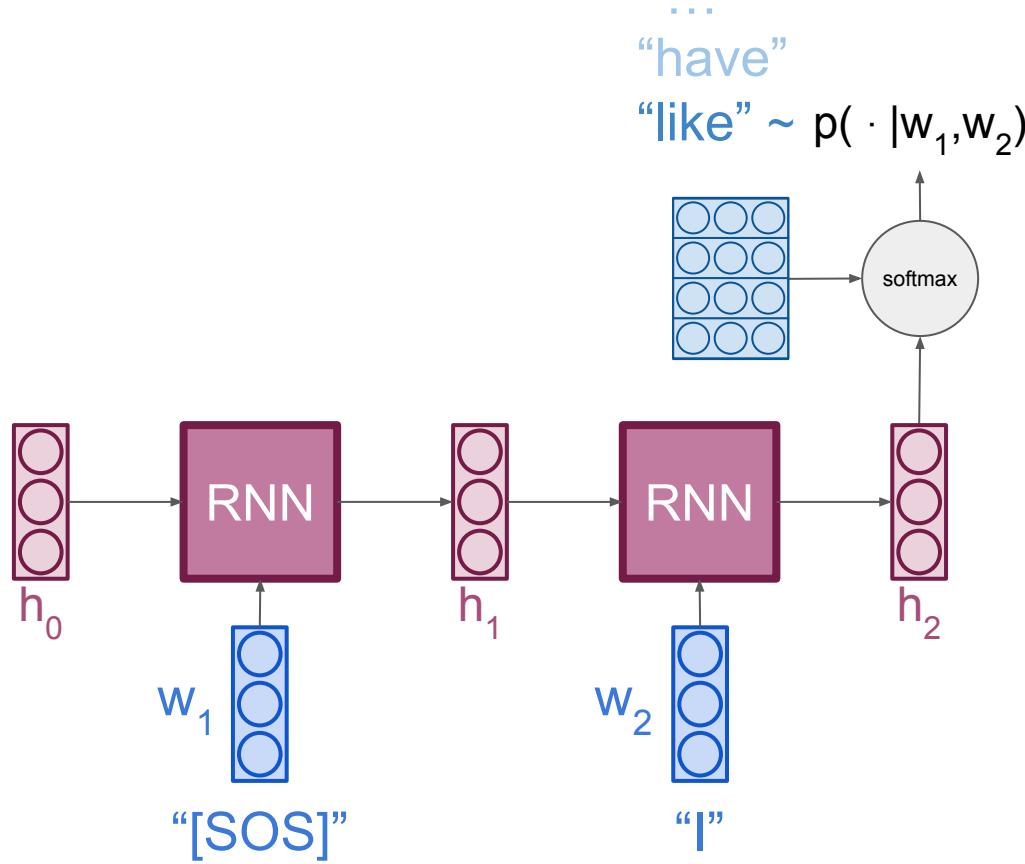
# Recurrent Neural Networks



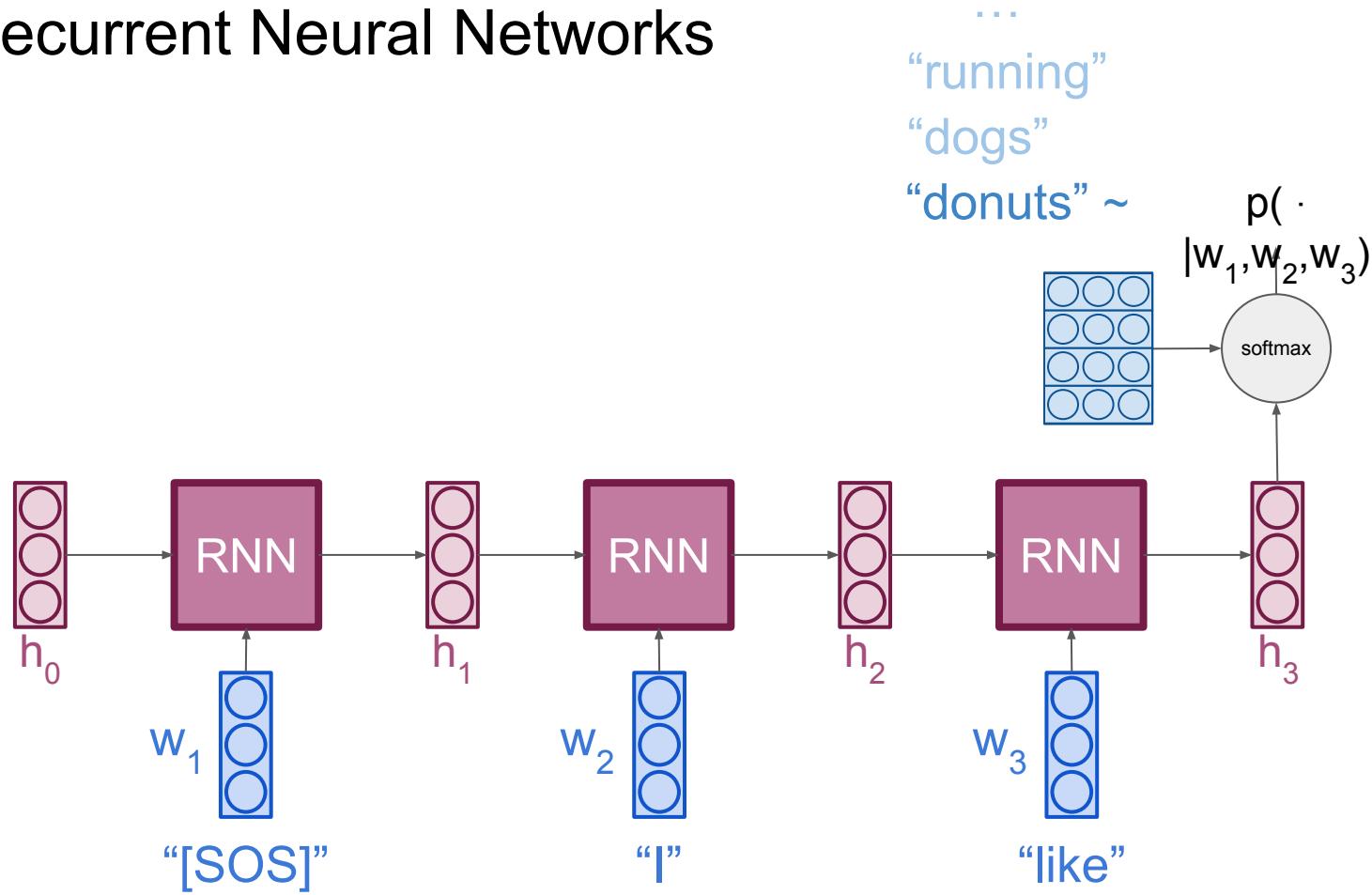
# Recurrent Neural Networks



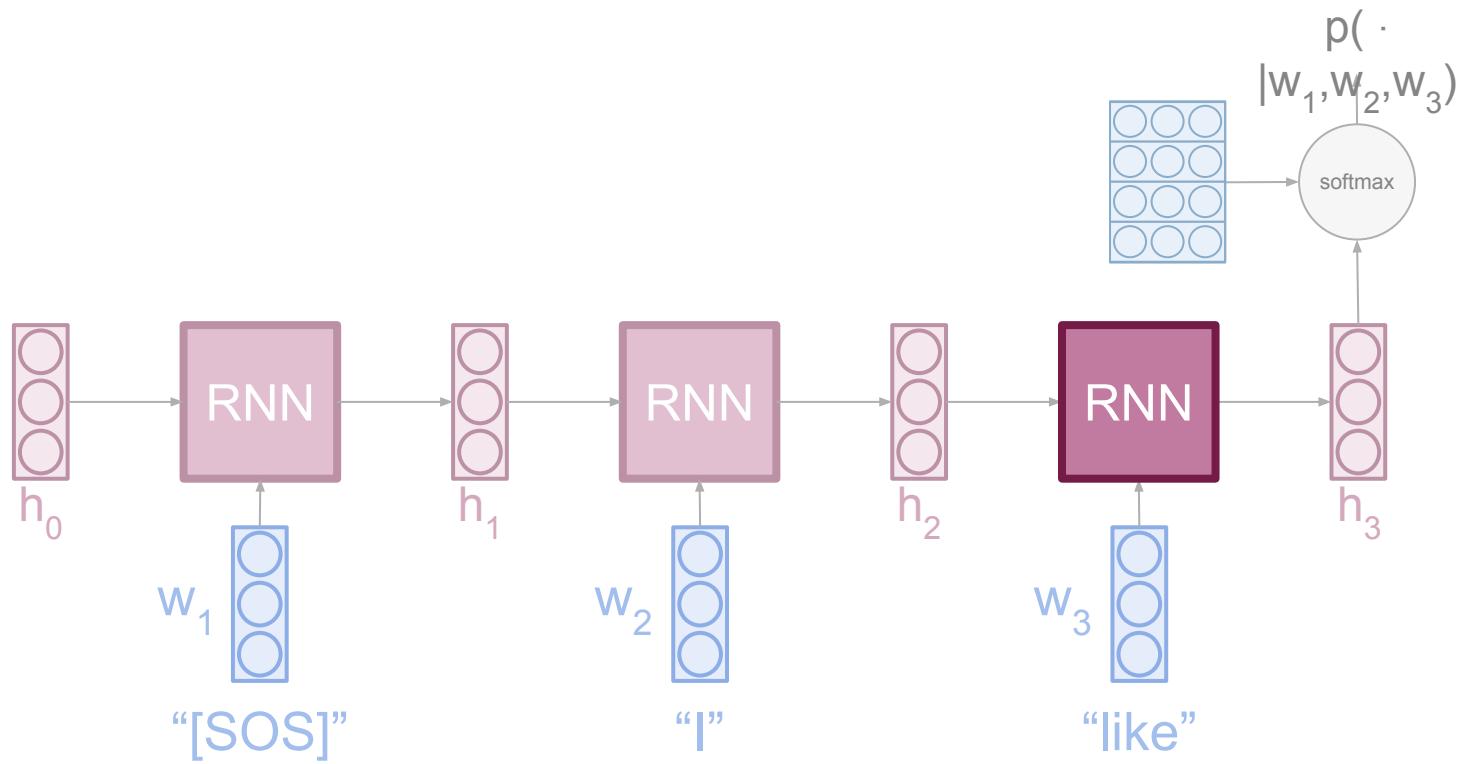
# Recurrent Neural Networks



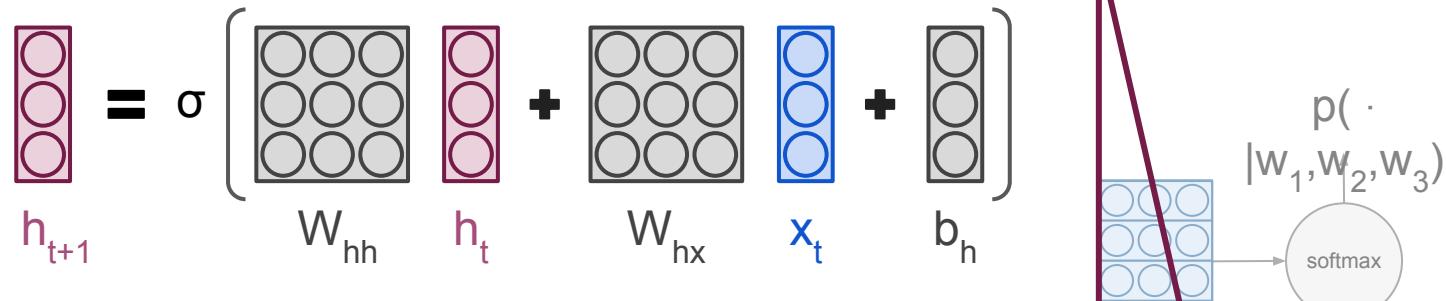
# Recurrent Neural Networks

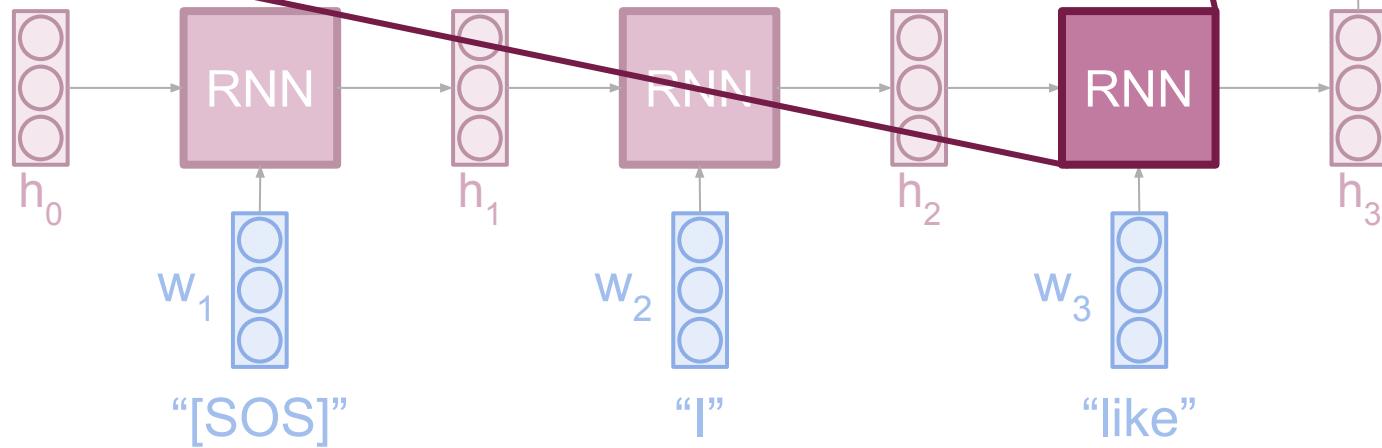


# Recurrent Neural Networks

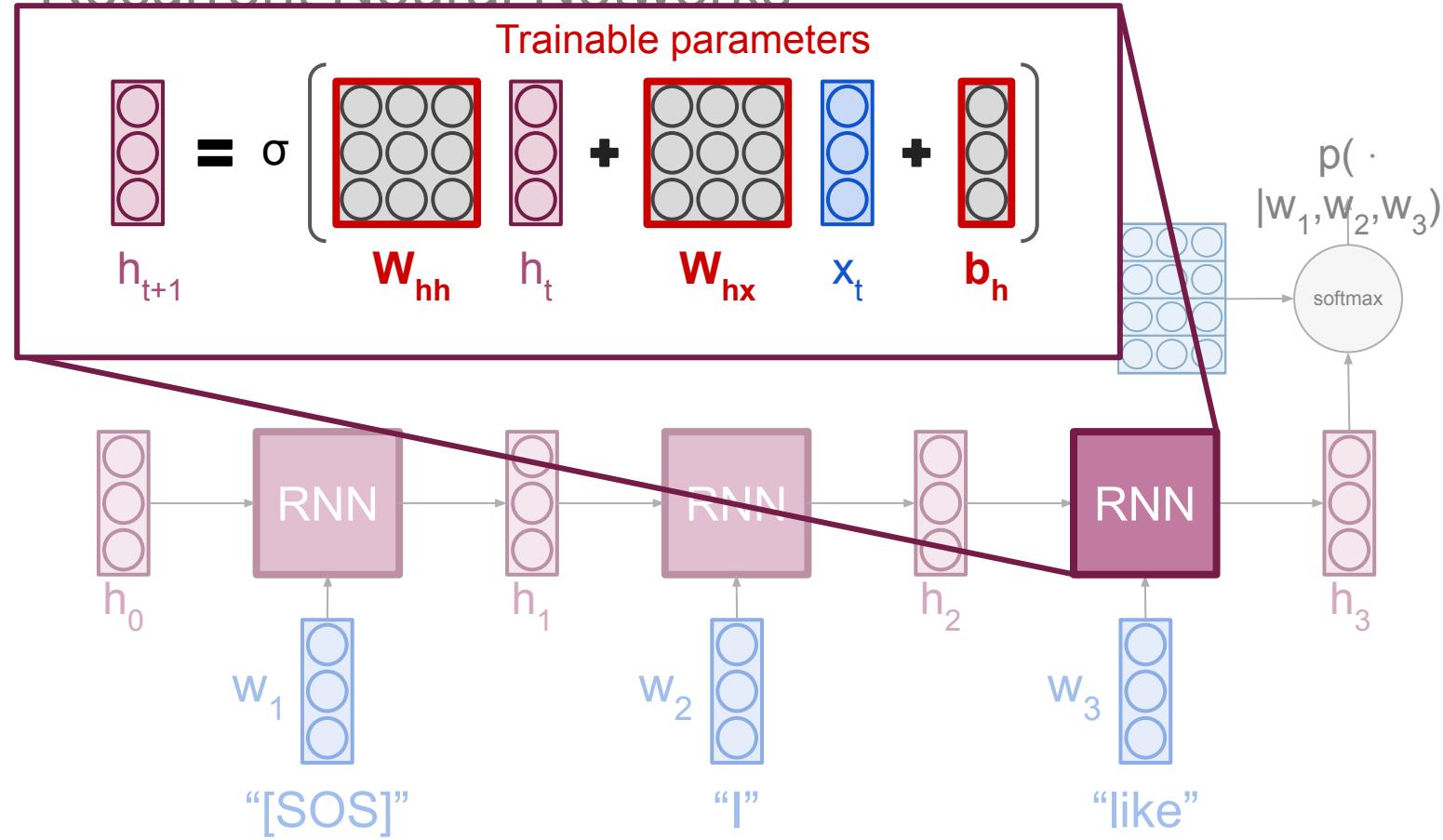


# Recurrent Neural Networks

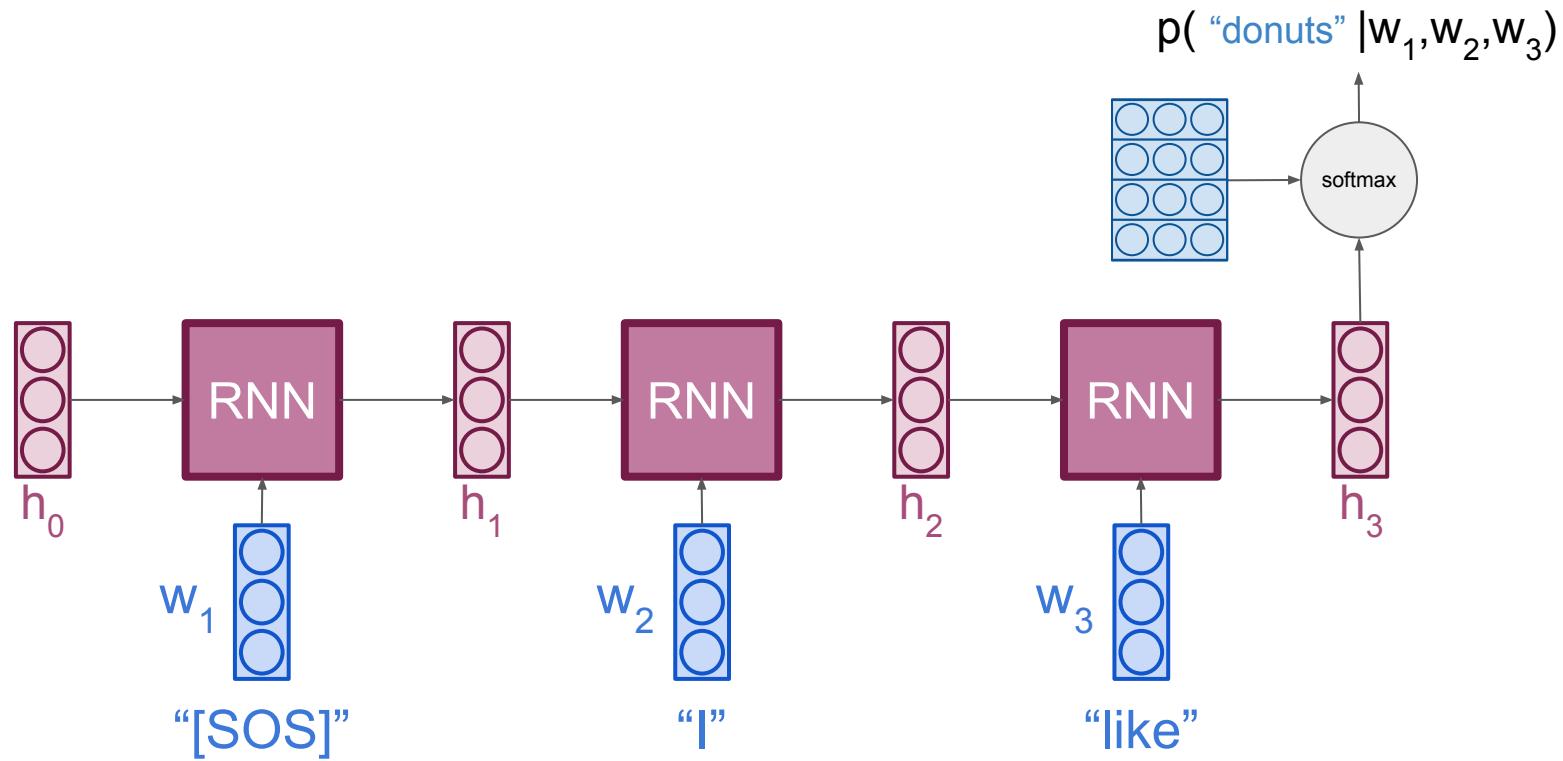
$$h_{t+1} = \sigma \left( W_{hh} h_t + W_{hx} x_t + b_h \right)$$




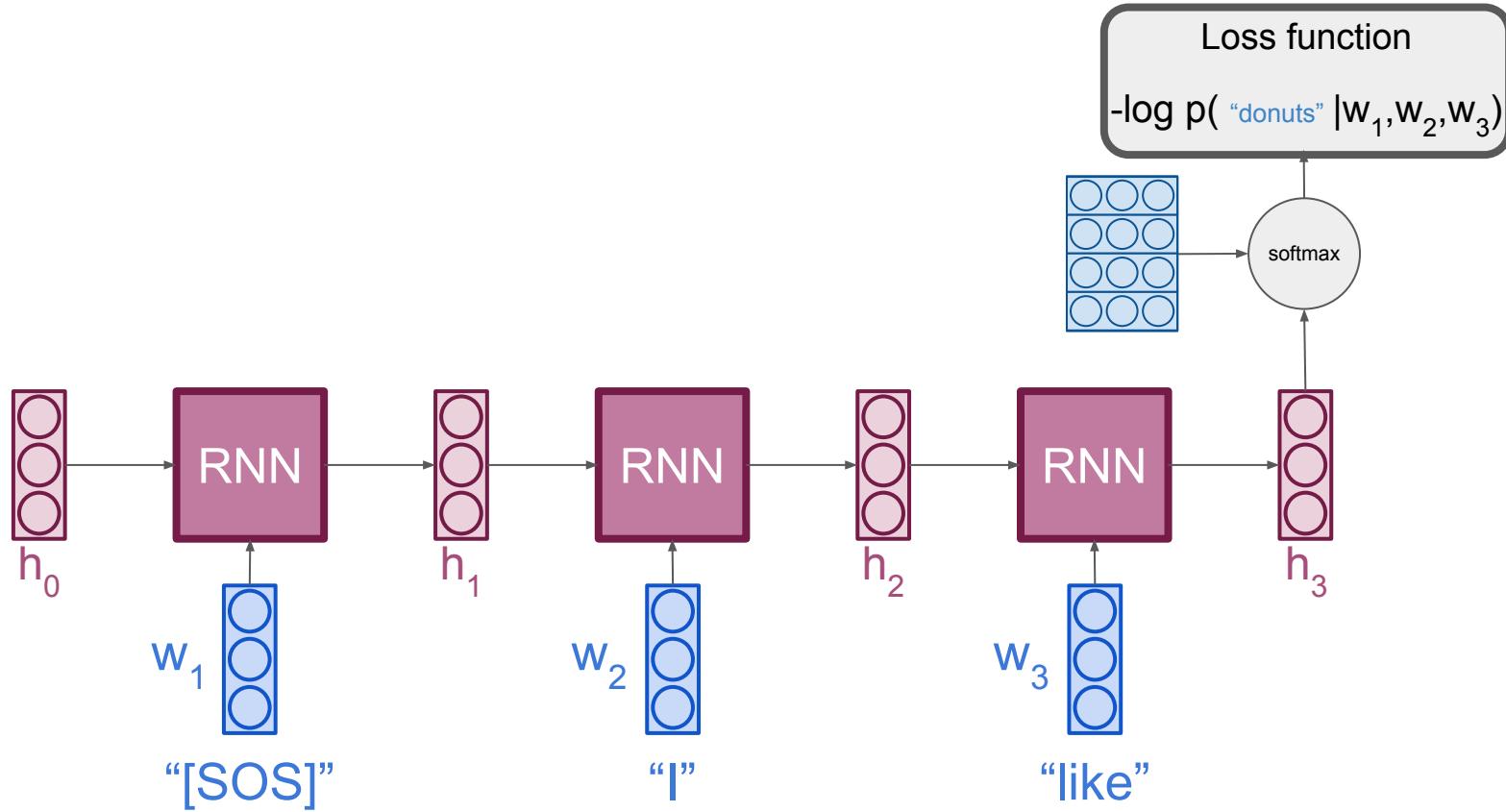
# Recurrent Neural Networks



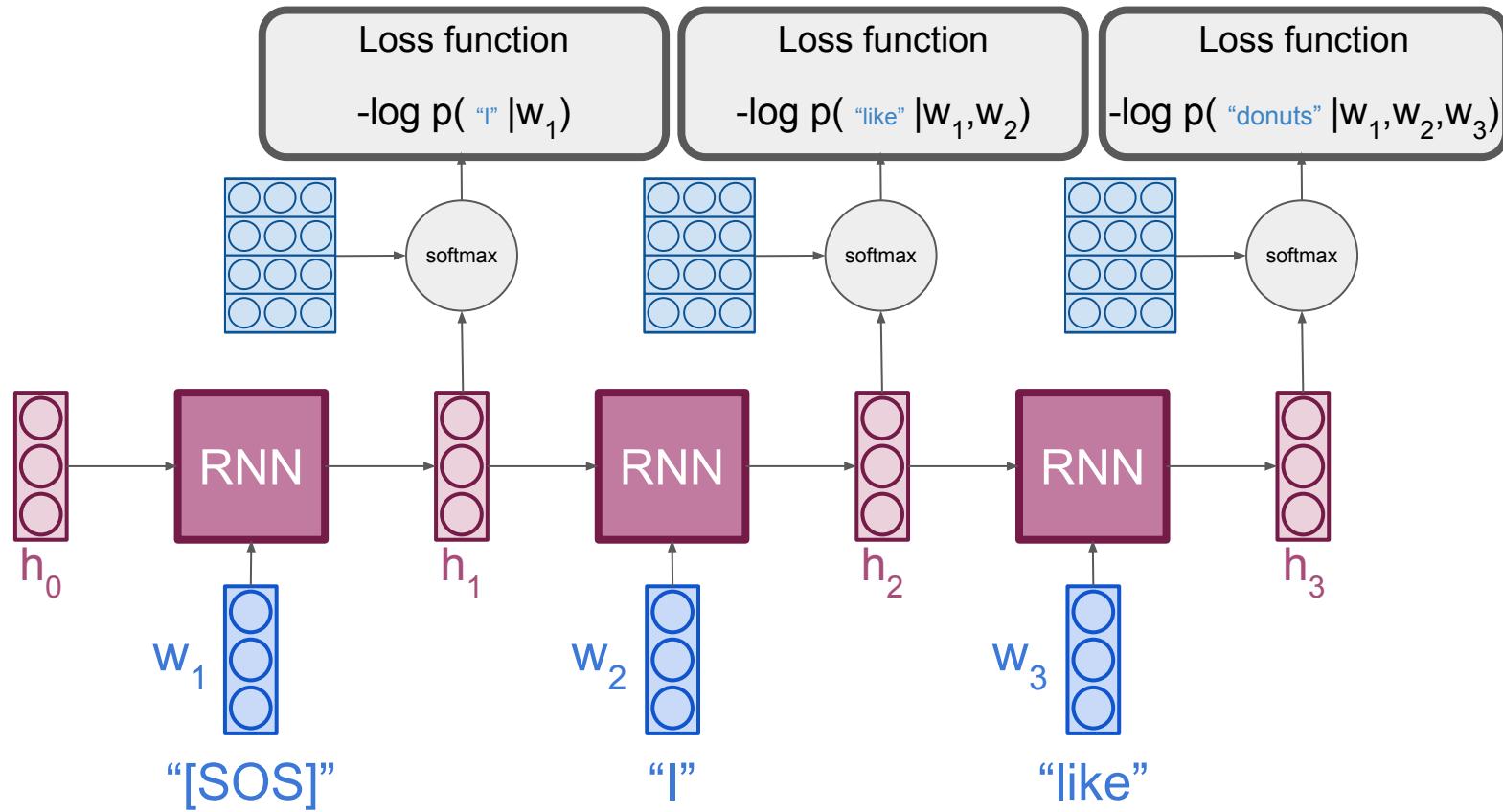
# Recurrent Neural Networks Training



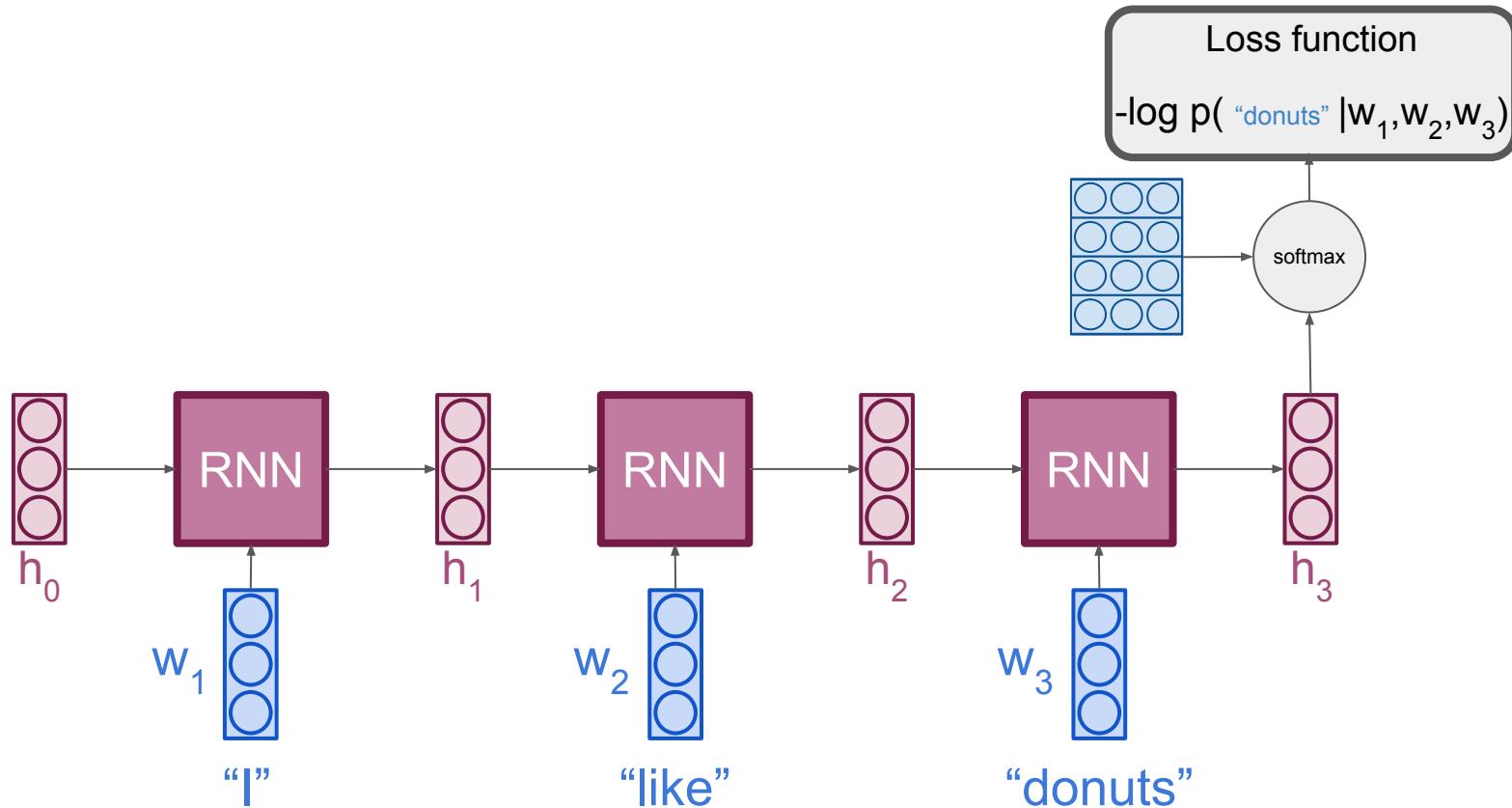
# Recurrent Neural Networks Training



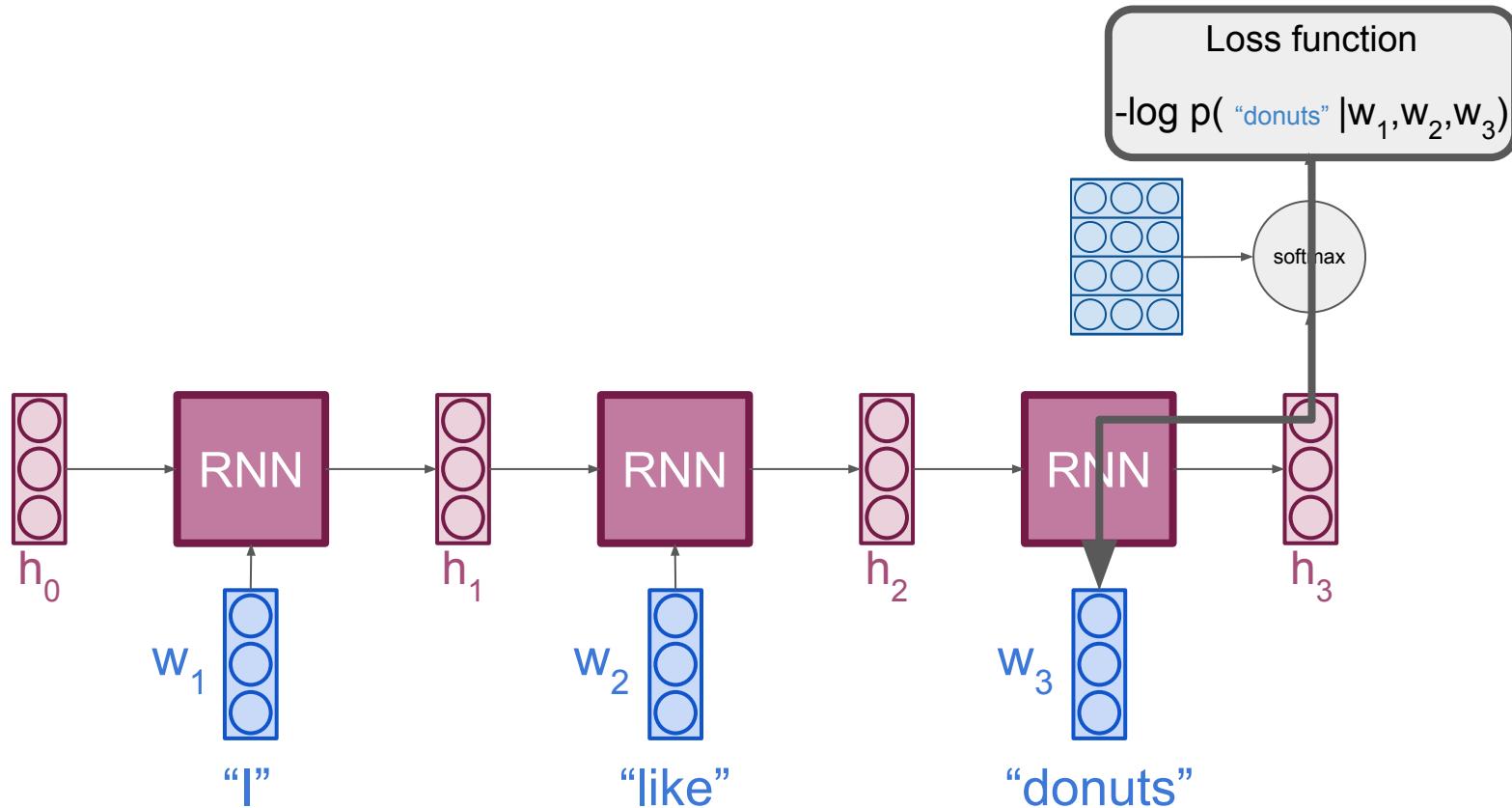
# Recurrent Neural Networks Training



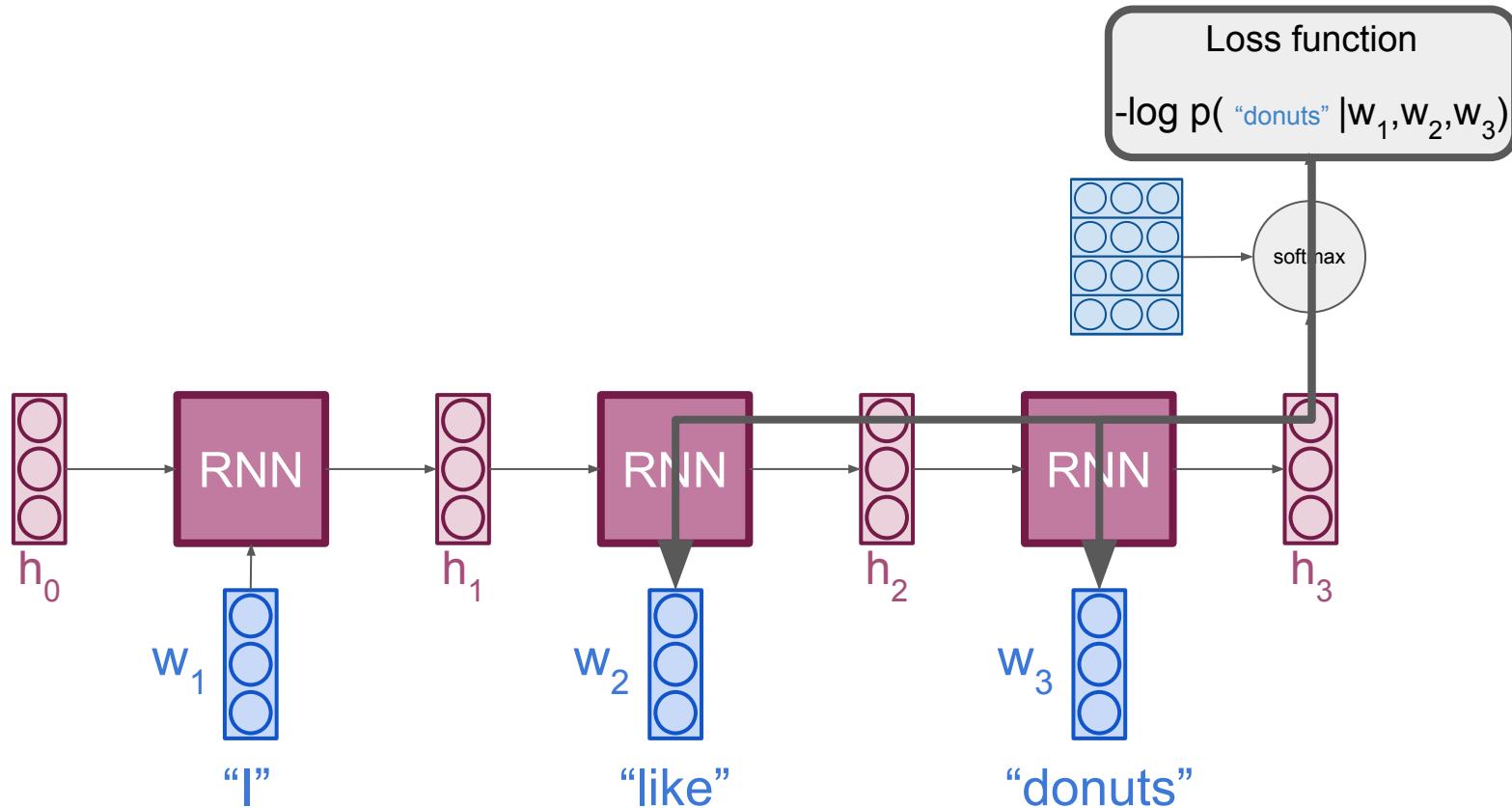
# Backpropagation Through Time (BPTT)



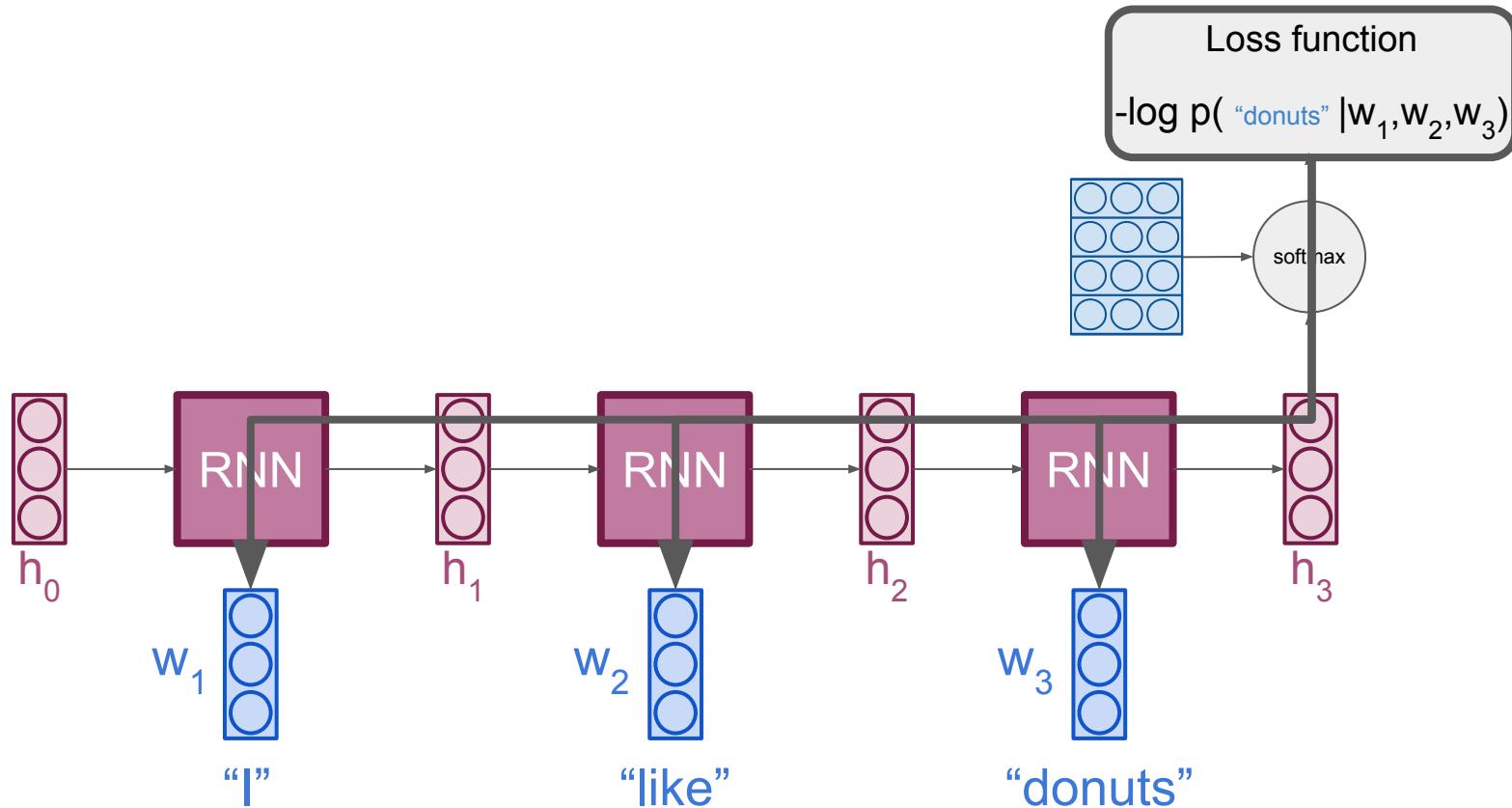
# Backpropagation Through Time (BPTT)



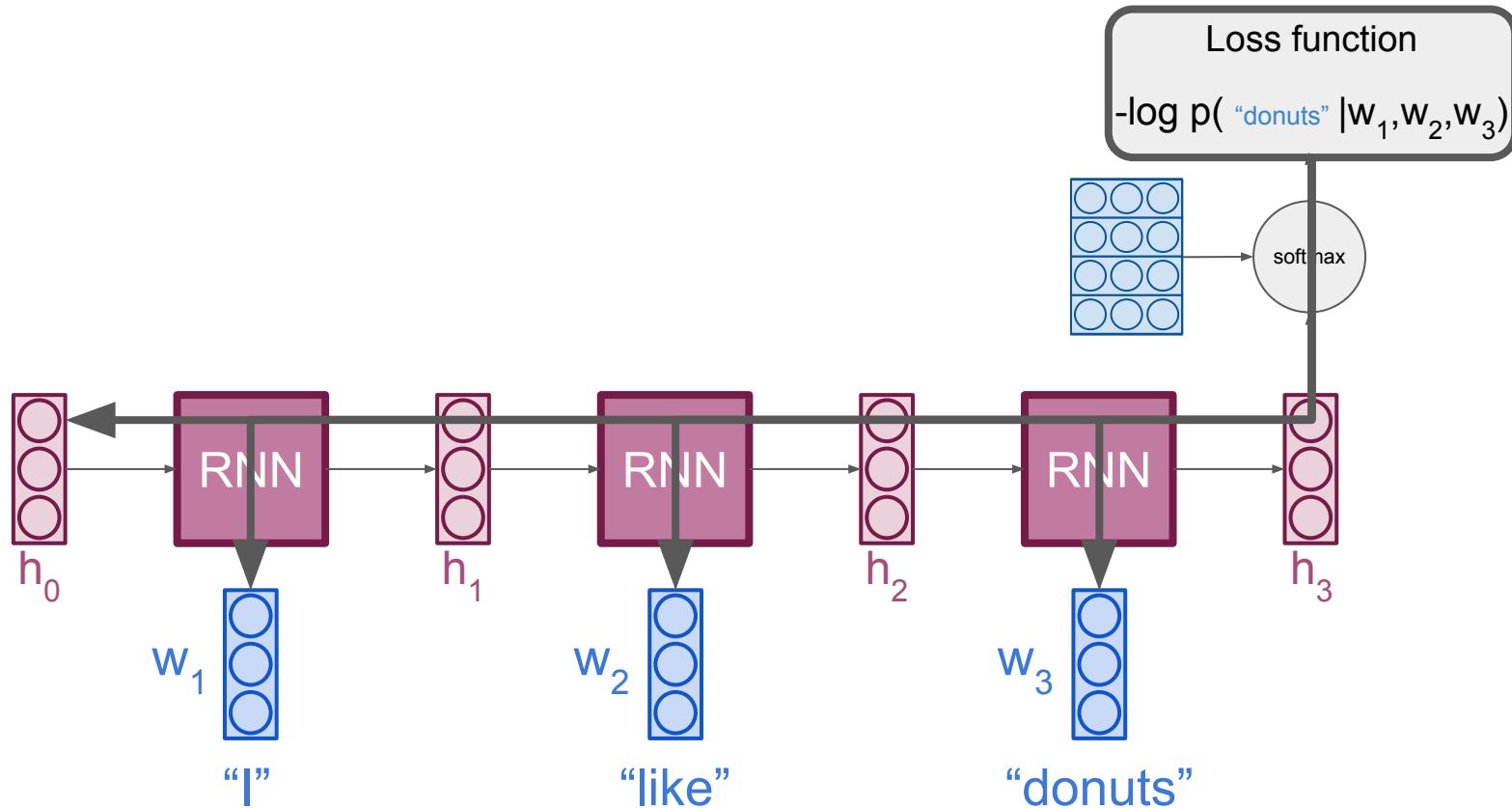
# Backpropagation Through Time (BPTT)



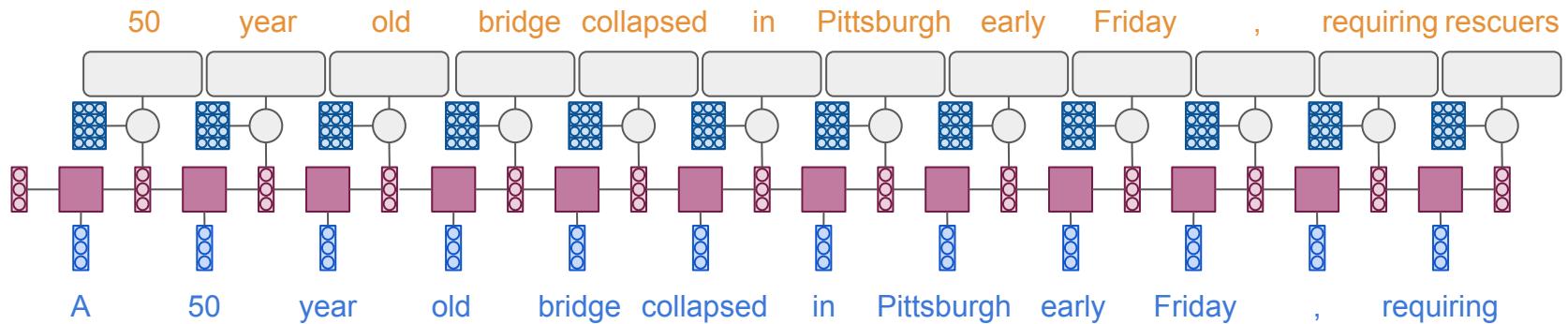
# Backpropagation Through Time (BPTT)



# Backpropagation Through Time (BPTT)



# Long Sequences



# Truncated BPTT

Compute gradient on shorter sub-sequence



# Truncated BPTT

Compute gradient on shorter sub-sequence  
Update parameters



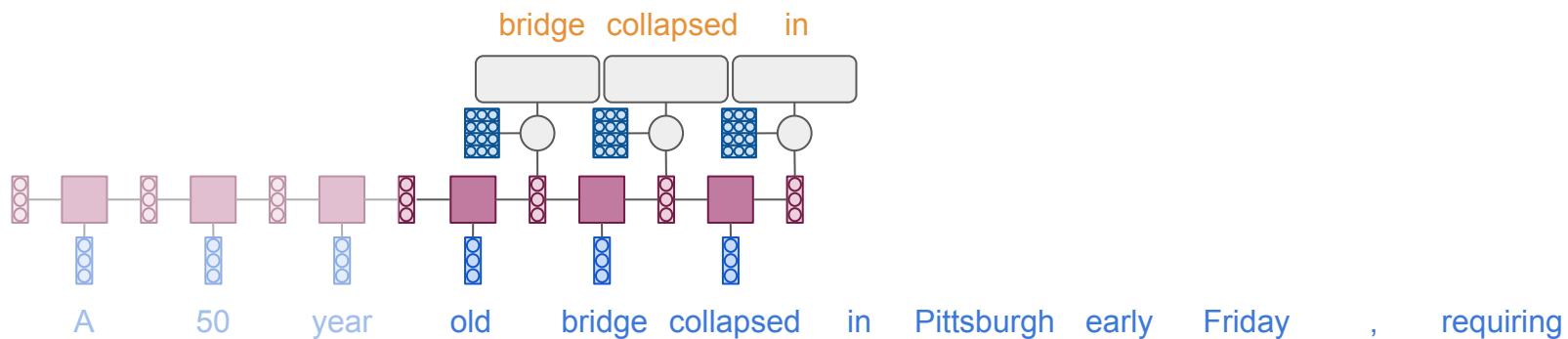
# Truncated BPTT

Keep last hidden state



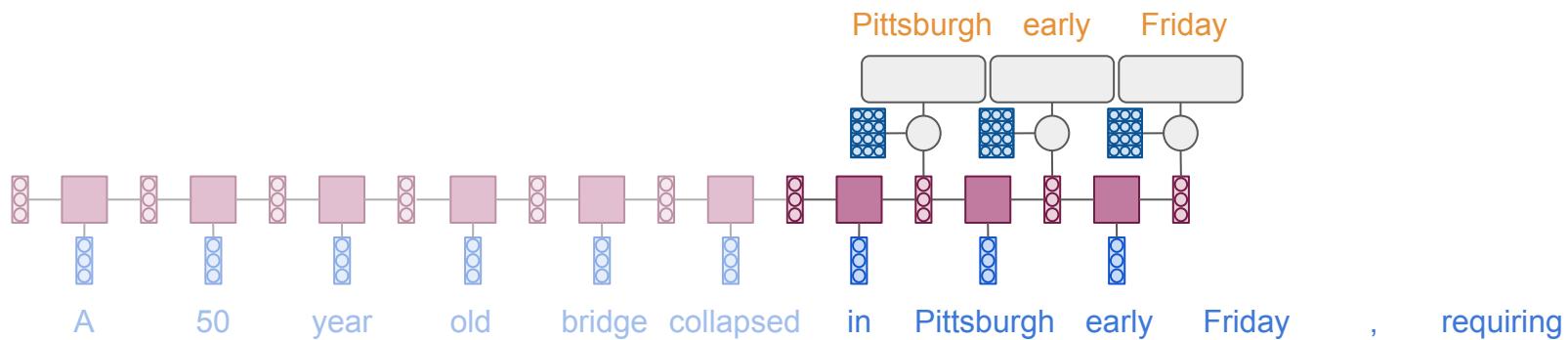
# Truncated BPTT

Compute gradient over next sub-sequence



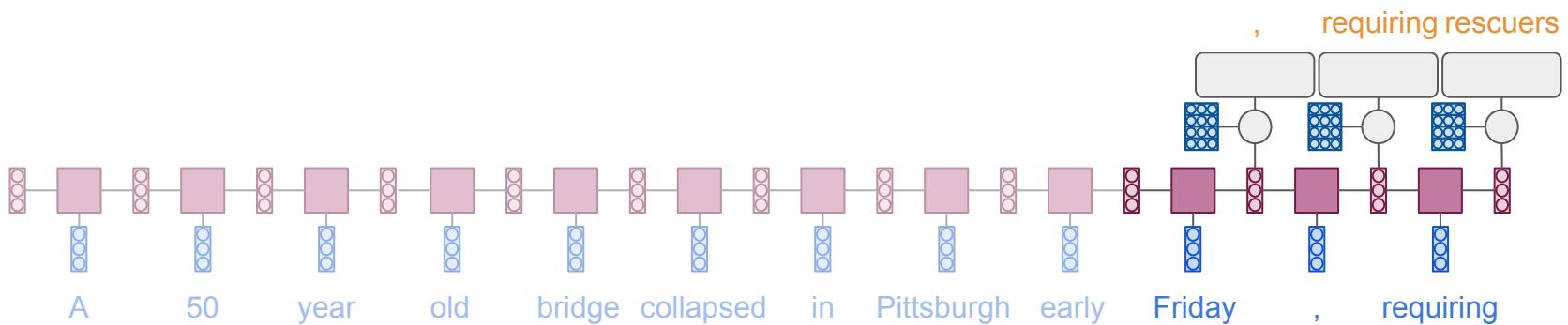
# Truncated BPTT

etc...



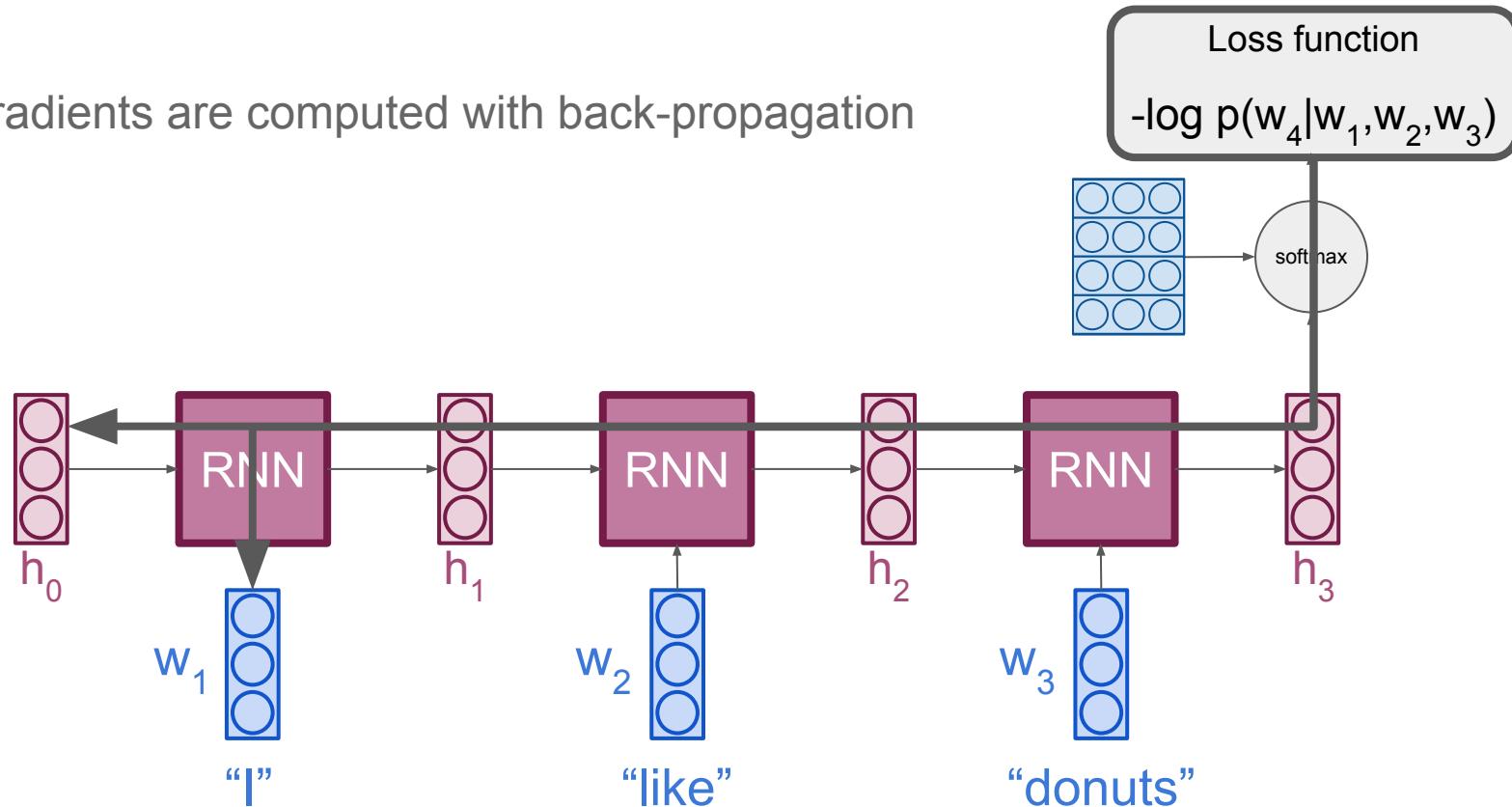
# Truncated BPTT

etc...



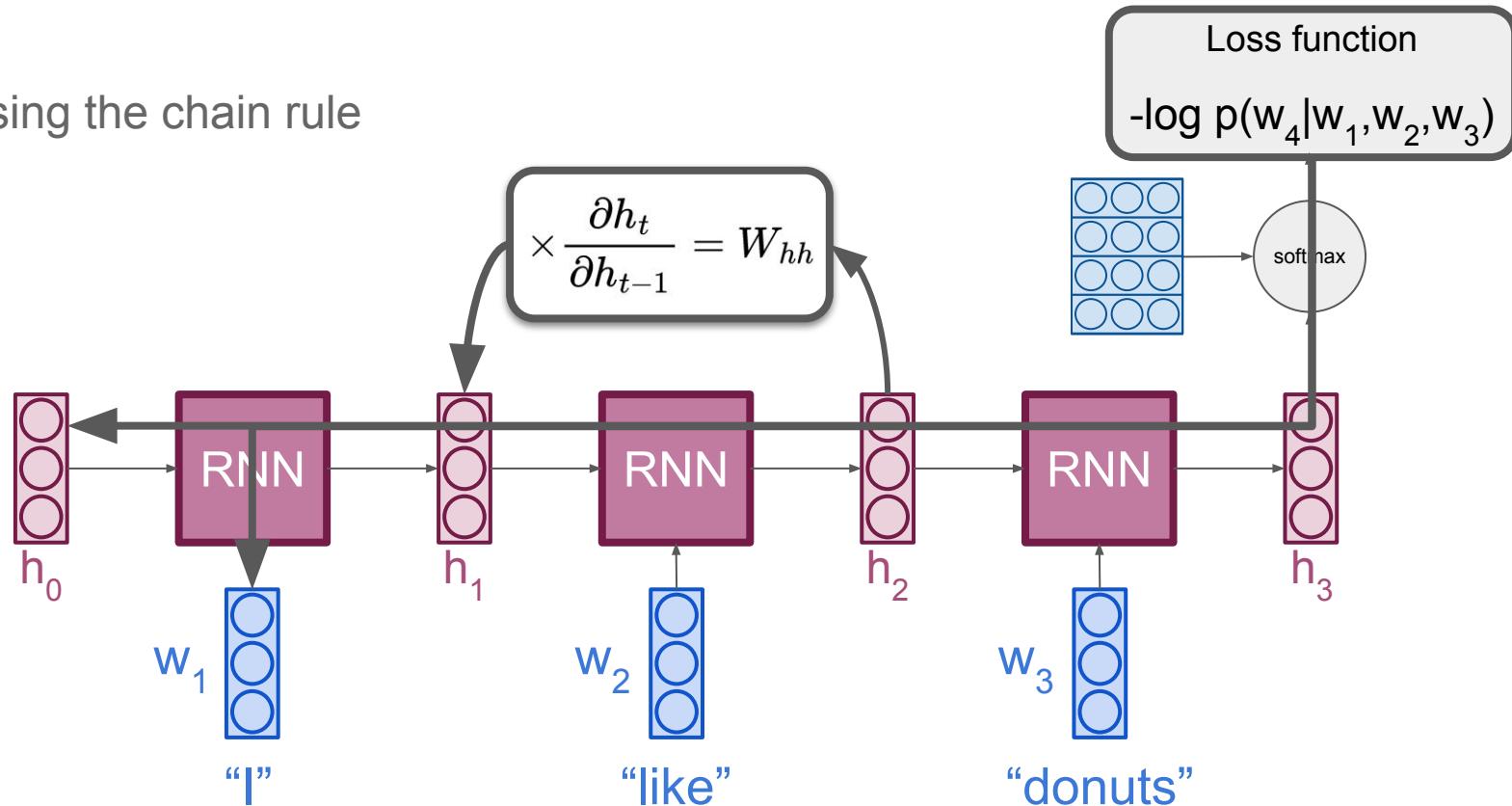
# Vanishing Gradients

Gradients are computed with back-propagation



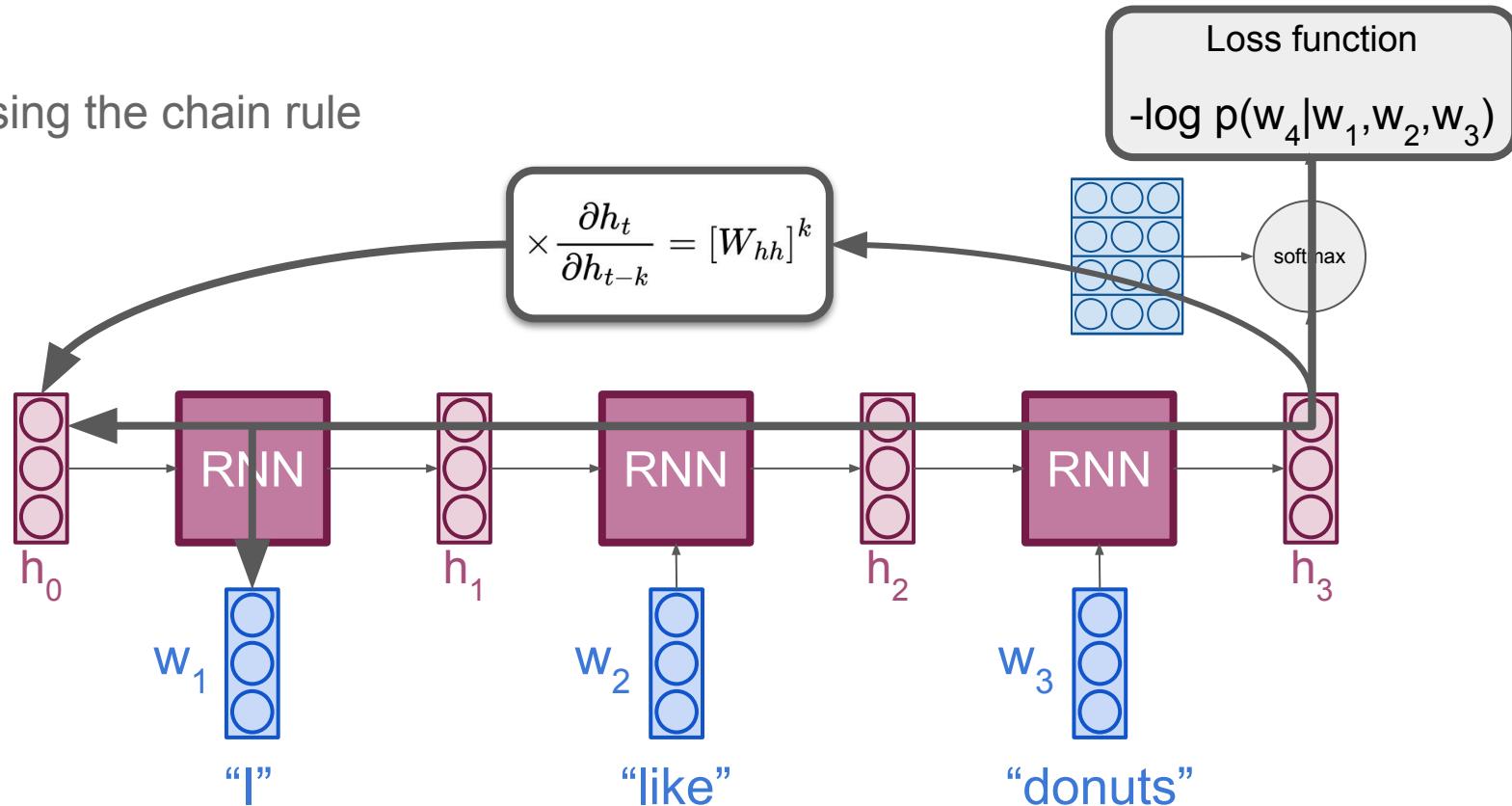
# Vanishing Gradients

Using the chain rule



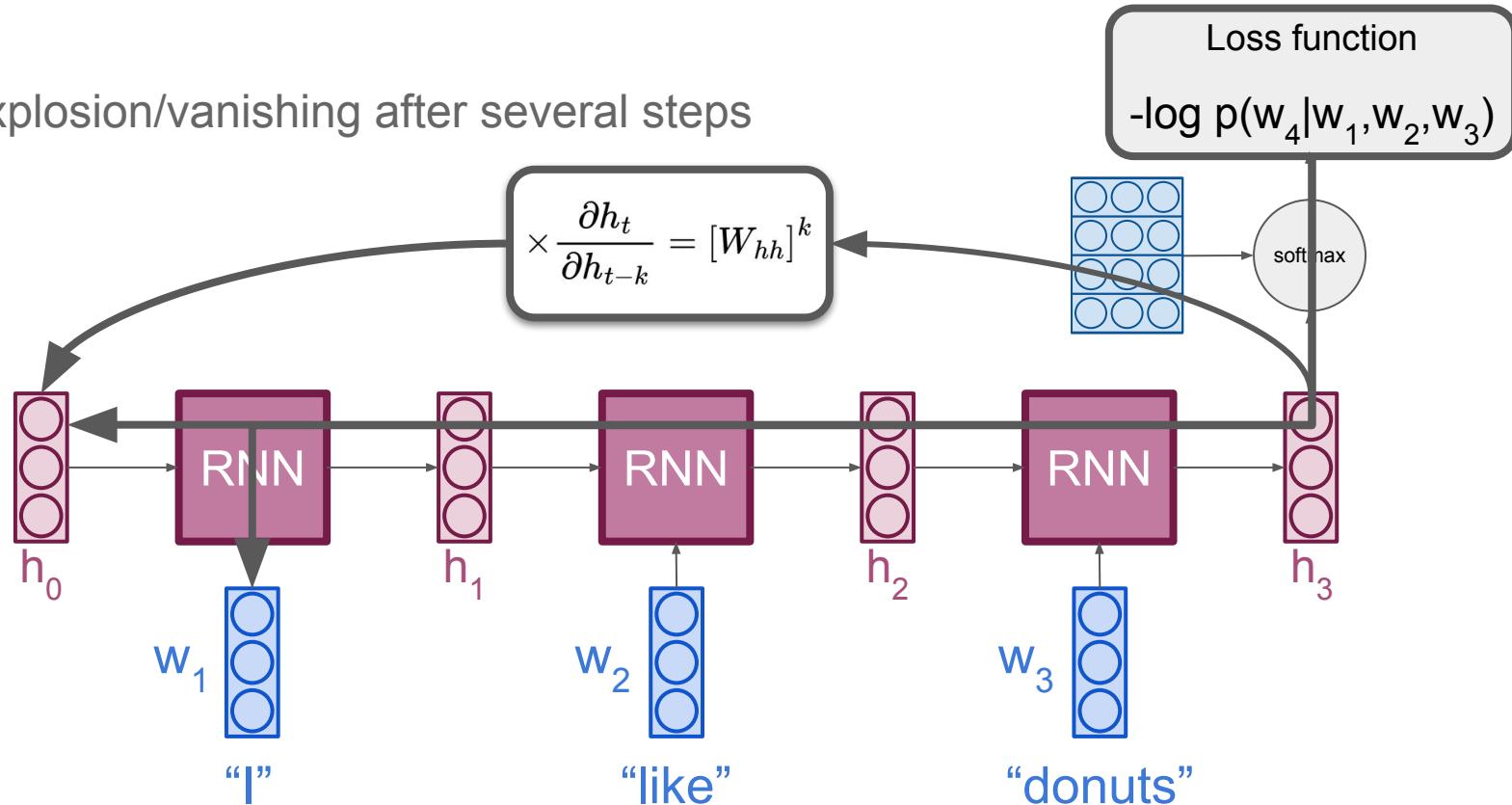
# Vanishing Gradients

Using the chain rule

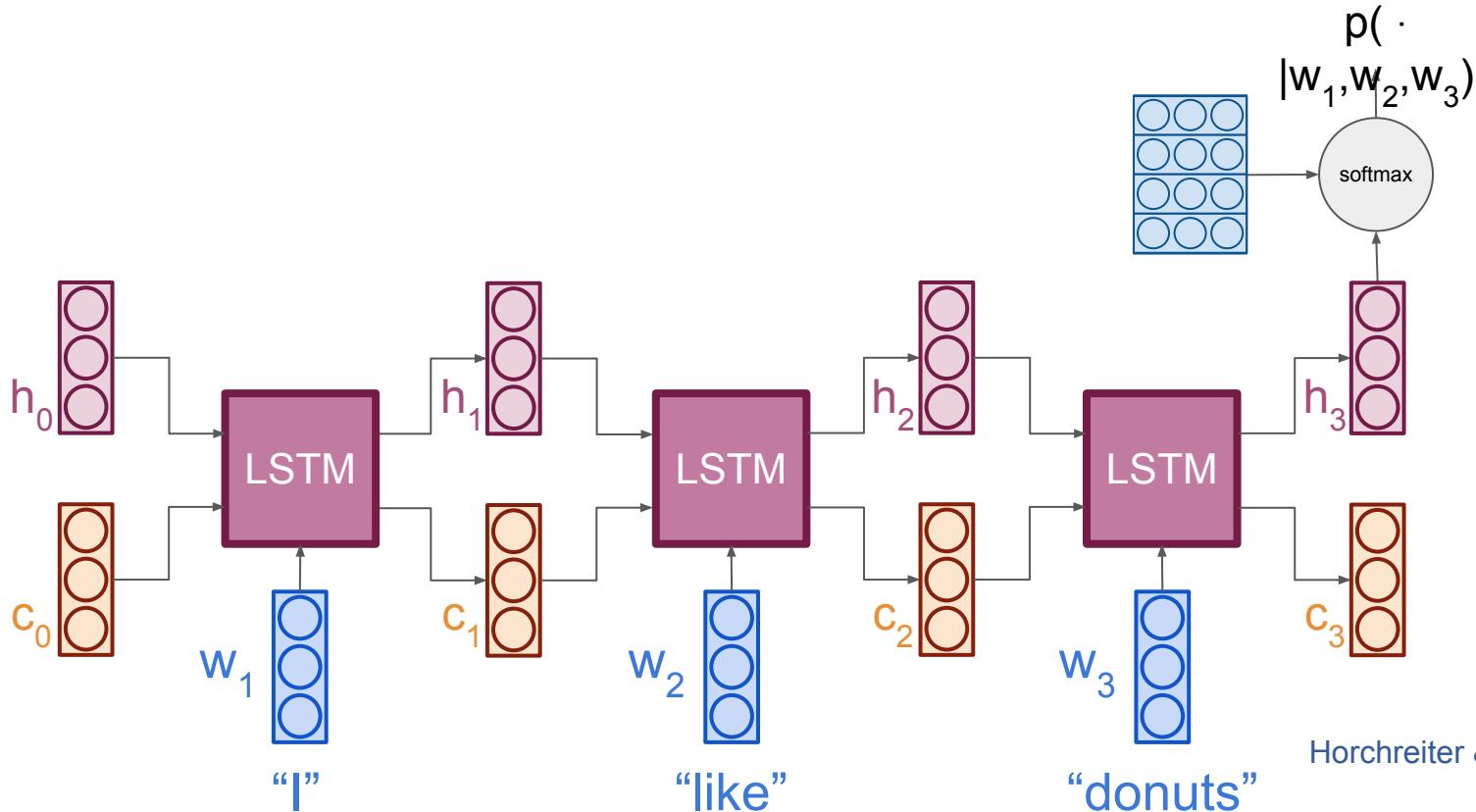


# Vanishing Gradients

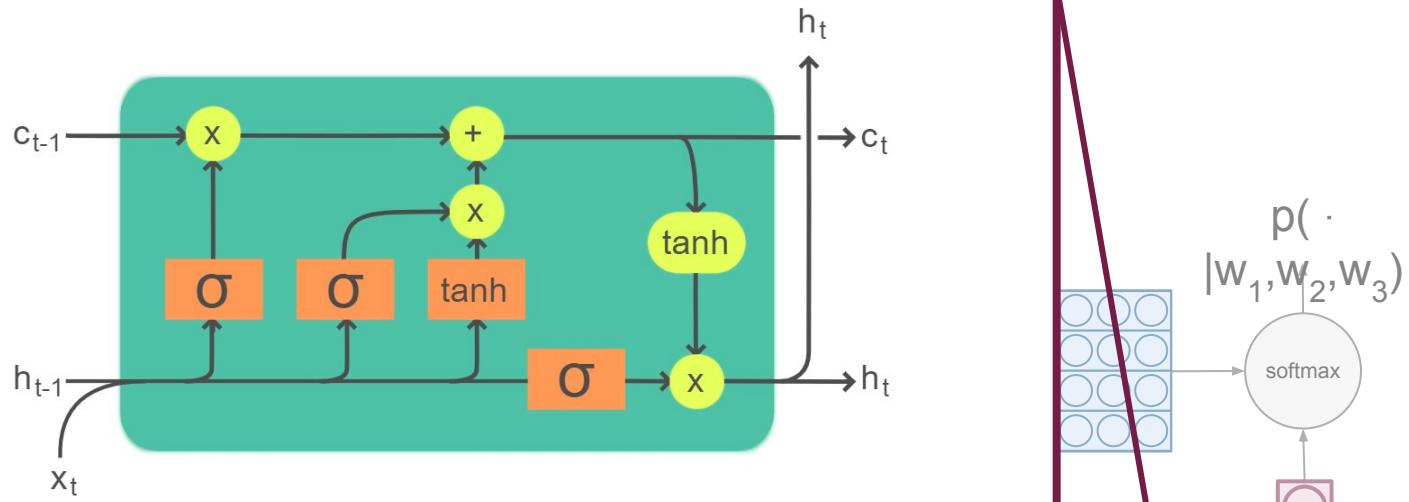
Explosion/vanishing after several steps



# Long Short Term Memory Networks



Hochreiter & Schmidhuber (1997)  
Gers et al. (2000)



## Legend:

Layer      Componentwise      Copy      Concatenate



STM



$w_1$



“I”



$w_2$



“like”



$w_3$



“donuts”

# Handling Large Vocabularies

- The softmax layer is a major bottleneck in LMs

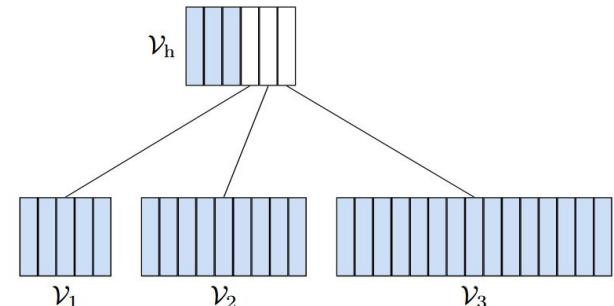
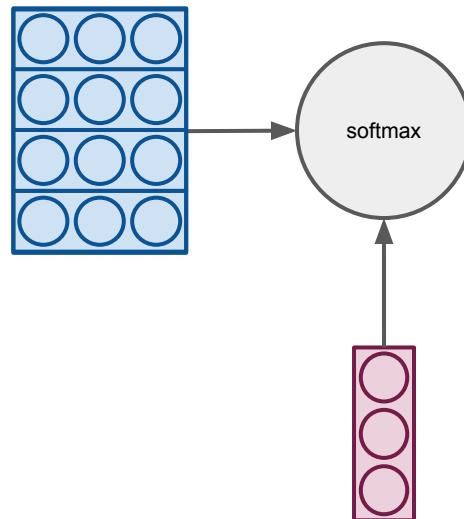


Figure 3. Our hierarchical model is organized as (i) a first level that includes both the most frequent words and vectors representing clusters, and (ii) clusters on the second level that are associated with rare words, the largest ones being associated with the less frequent words. The sizes are determined so as to minimize our computational model on GPU.

# Handling Large Vocabularies

- The softmax layer is a major bottleneck in LMs

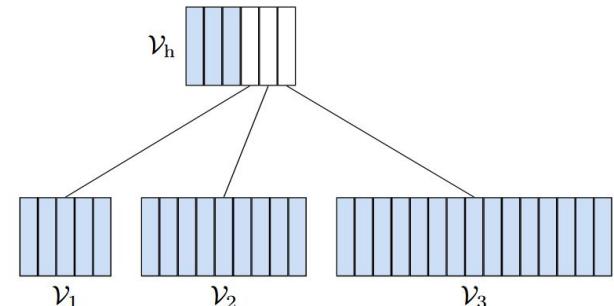
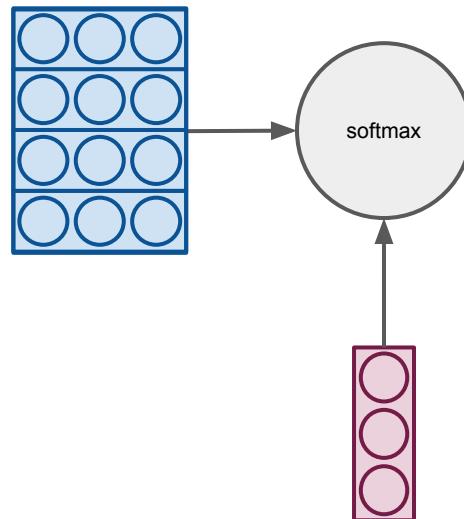


Figure 3. Our hierarchical model is organized as (i) a first level that includes both the most frequent words and vectors representing clusters, and (ii) clusters on the second level that are associated with rare words, the largest ones being associated with the less frequent words. The sizes are determined so as to minimize our computational model on GPU.

## [Optional if time permits] Exposure Bias

- Discrepancy between training and sampling
  - Distribution shift from true prefixes to sampled prefixes
- Scheduled sampling?

# Recurrent Models in a Nutshell

- Better language models than n-grams

# Recurrent Models in a Nutshell

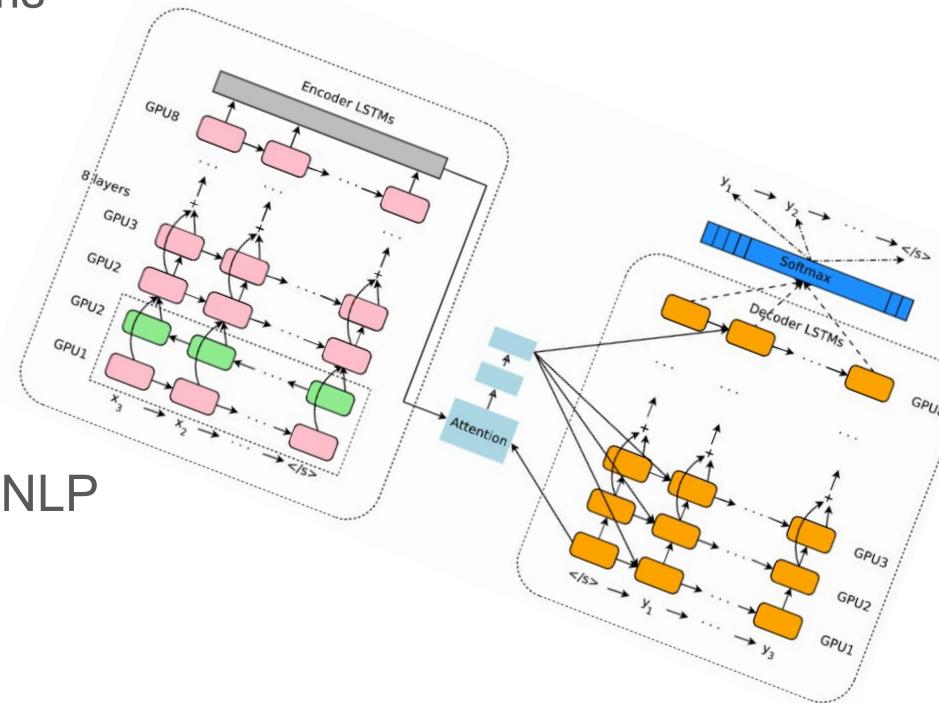
- Better language models than n-grams
- Highly tuned
  - Fast implementations
  - Good default hyper-parameters

TensorFlow > API > TensorFlow Core v2.8.0 > Python

**tf.compat.v1.keras.layers.CuDNNLSTM**

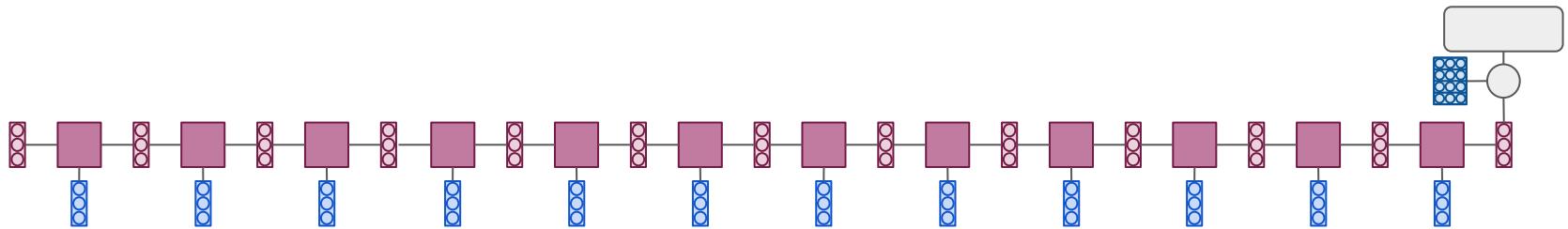
# Recurrent Models in a Nutshell

- Better language models than n-grams
- Highly tuned
  - Fast implementations
  - Good default hyper-parameters
- LSTM & variants were backbone of NLP at some point
  - Google translate ran on LSTM stack



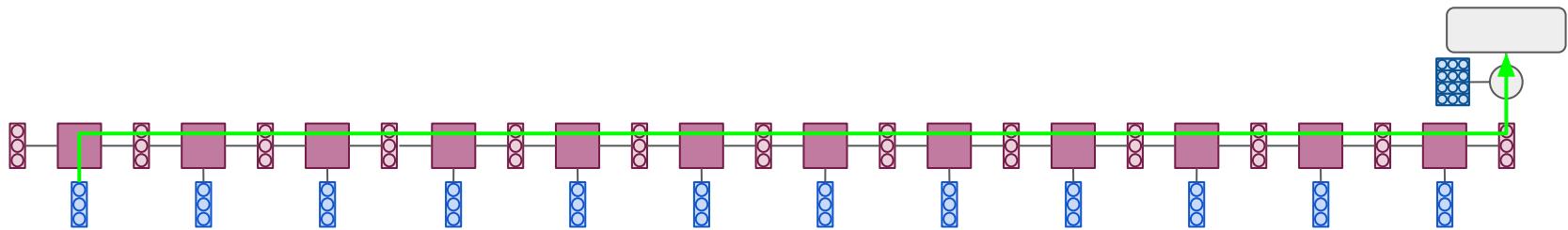
# Downsides of Recurrent Models

- How much context do they really use?
  - Theoretically unbounded



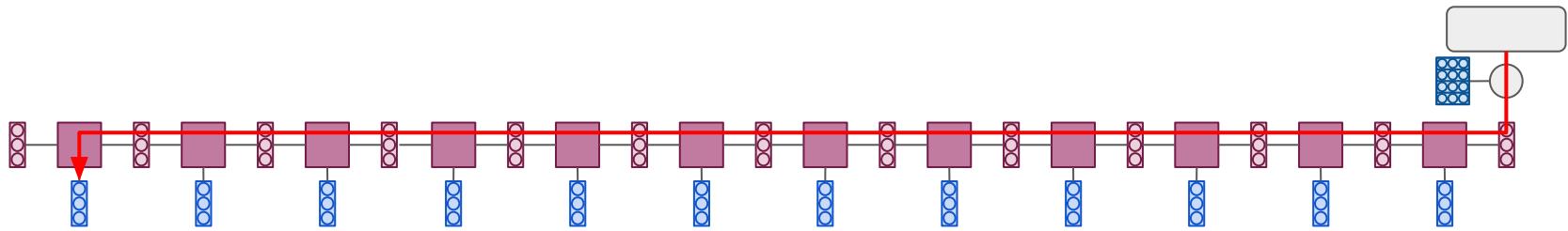
# Downsides of Recurrent Models

- How much context do they really use?
  - Theoretically unbounded



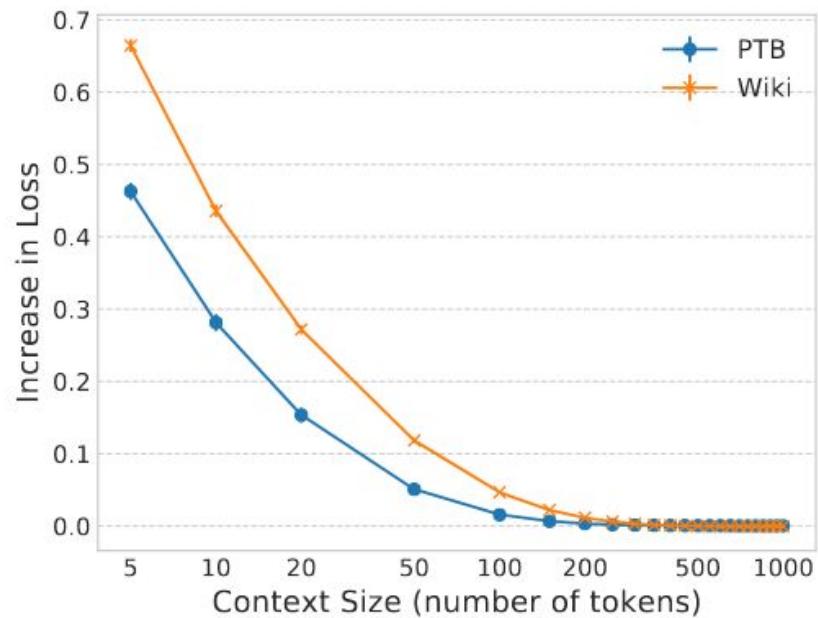
# Downsides of Recurrent Models

- How much context do they really use?
  - Theoretically unbounded



# Downsides of Recurrent Models

- How much context do they really use?
  - Theoretically unbounded
  - In practice: ~200 tokens



From Khandelwal et al. (2018)

# Downsides of Recurrent Models

- How much context do they really use?
  - Theoretically unbounded
  - In practice: ~200 tokens
- Scalability issues
  - Need to encode each token sequentially
  - Hard to parallelize during training
  - Difficult to scale to >8 layers (vanishing gradients, etc..)

# Attention and the Transformer Architecture

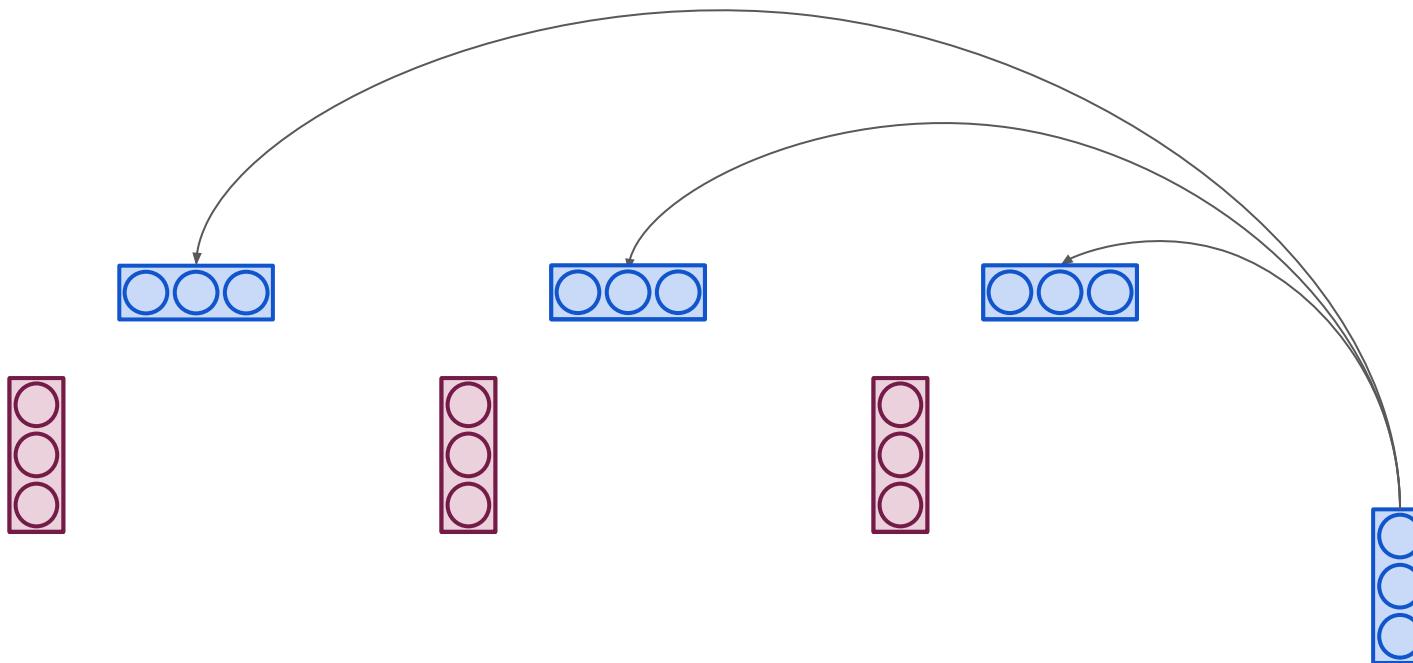
# Attention

- Adaptive weighted average of vectors



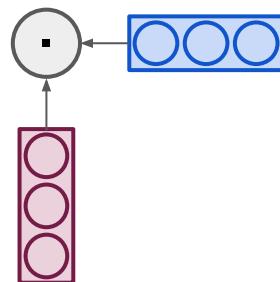
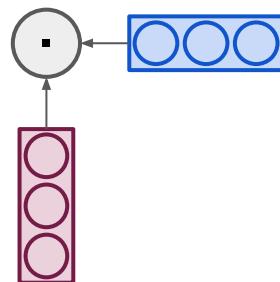
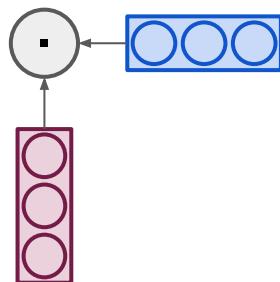
# Attention

- Adaptive weighted average of vectors



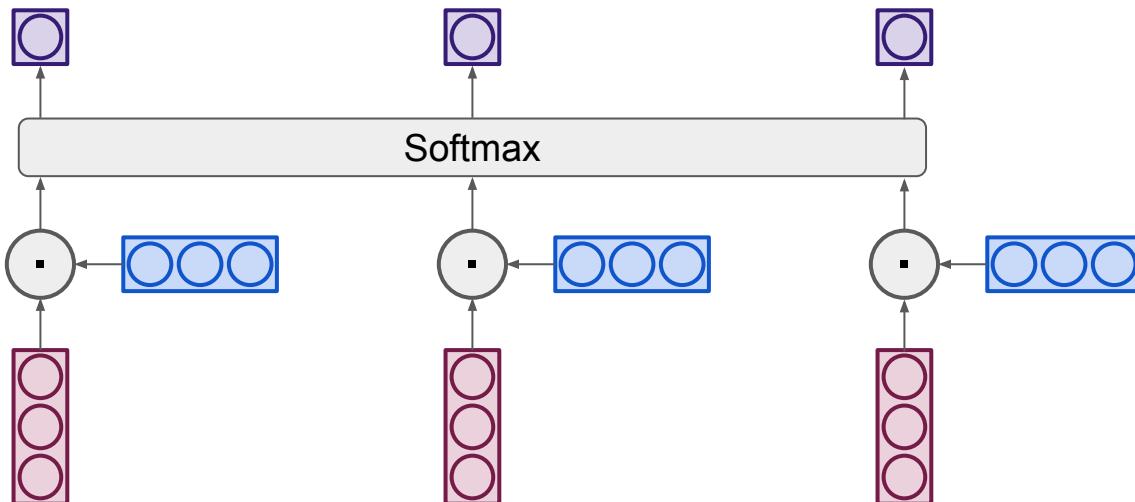
# Attention

- Adaptive weighted average of vectors



# Attention

- Adaptive weighted average of vectors

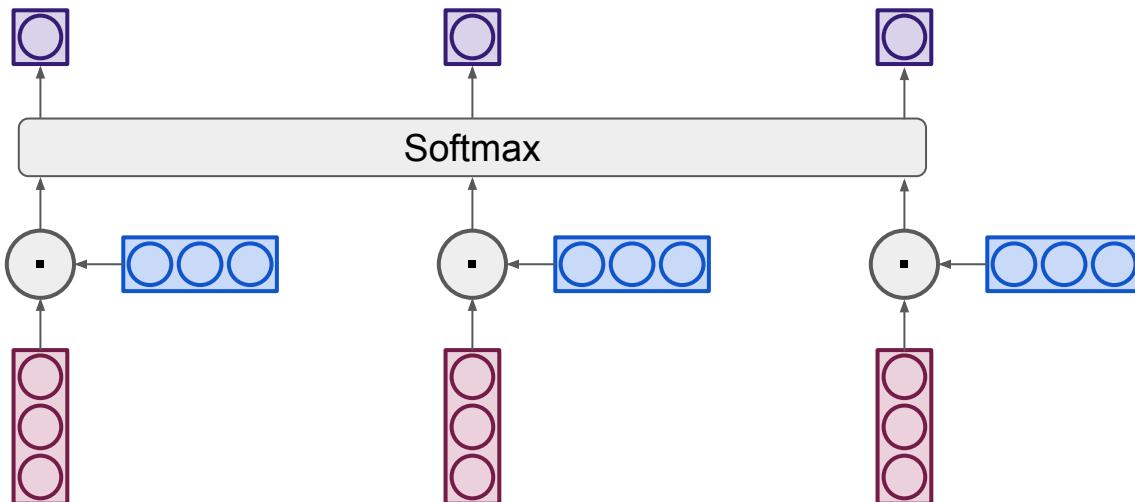


# Attention



# Attention

- Adaptive weighted average of vectors



# Attention

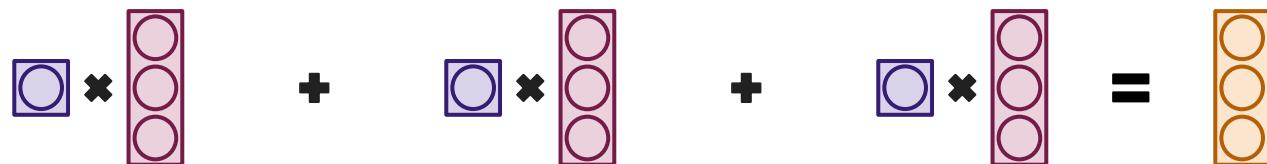


# Attention

$$\text{Input} \times \text{Weights} + \text{Input} \times \text{Weights} + \text{Input} \times \text{Weights}$$

The diagram illustrates the computation of attention weights. It shows three input vectors (blue squares) being multiplied (indicated by an asterisk) with three weight vectors (red rectangles). The results are then summed (indicated by plus signs).

# Attention

$$\begin{matrix} \text{purple square} \\ \times \\ \text{pink rectangle} \end{matrix} + \begin{matrix} \text{purple square} \\ \times \\ \text{pink rectangle} \end{matrix} + \begin{matrix} \text{purple square} \\ \times \\ \text{pink rectangle} \end{matrix} = \begin{matrix} \text{orange rectangle} \end{matrix}$$


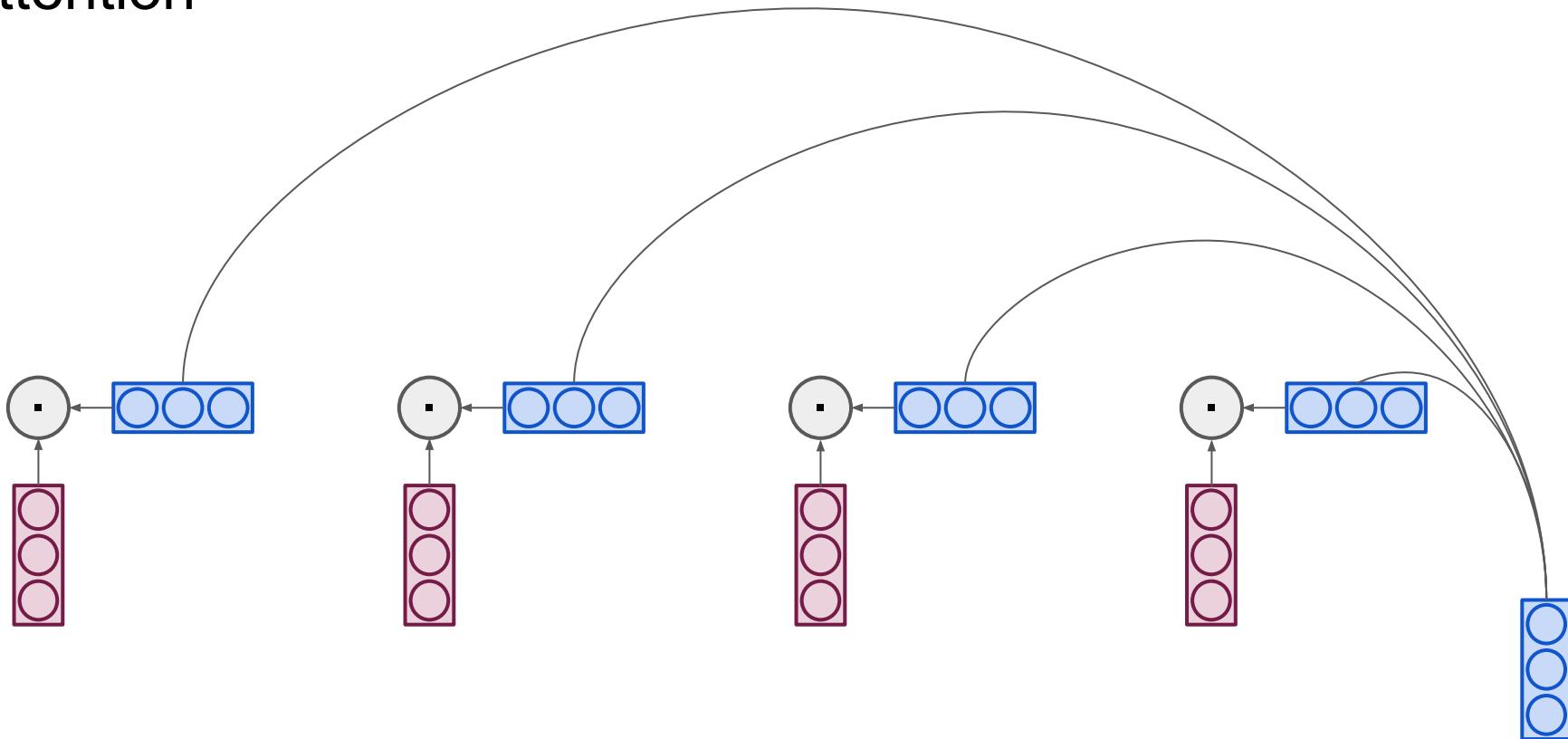
# Attention



# Attention



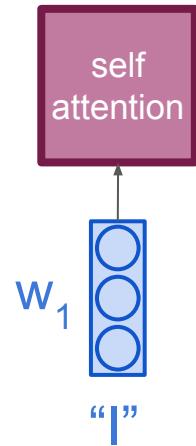
# Attention



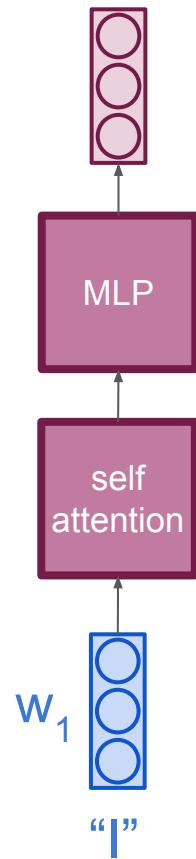
# Transformers



# Transformers



# Transformers

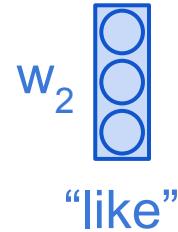


# Transformers



$w_1$

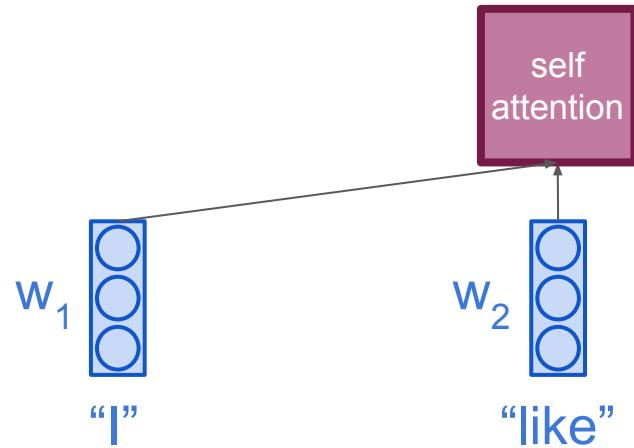
"I"



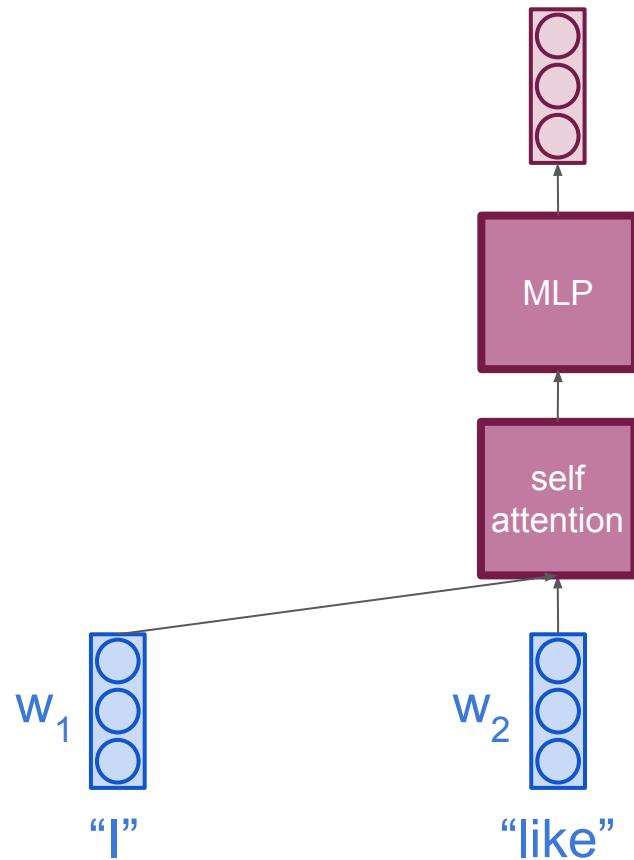
$w_2$

"like"

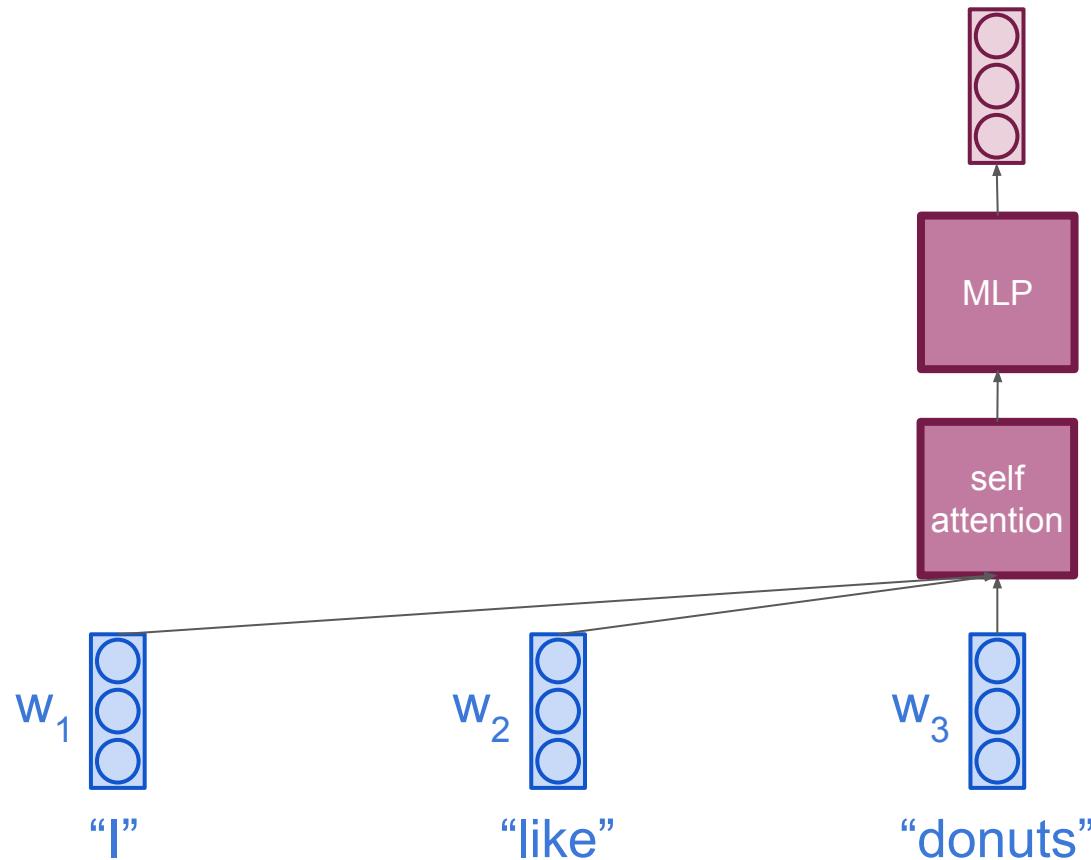
# Transformers



# Transformers

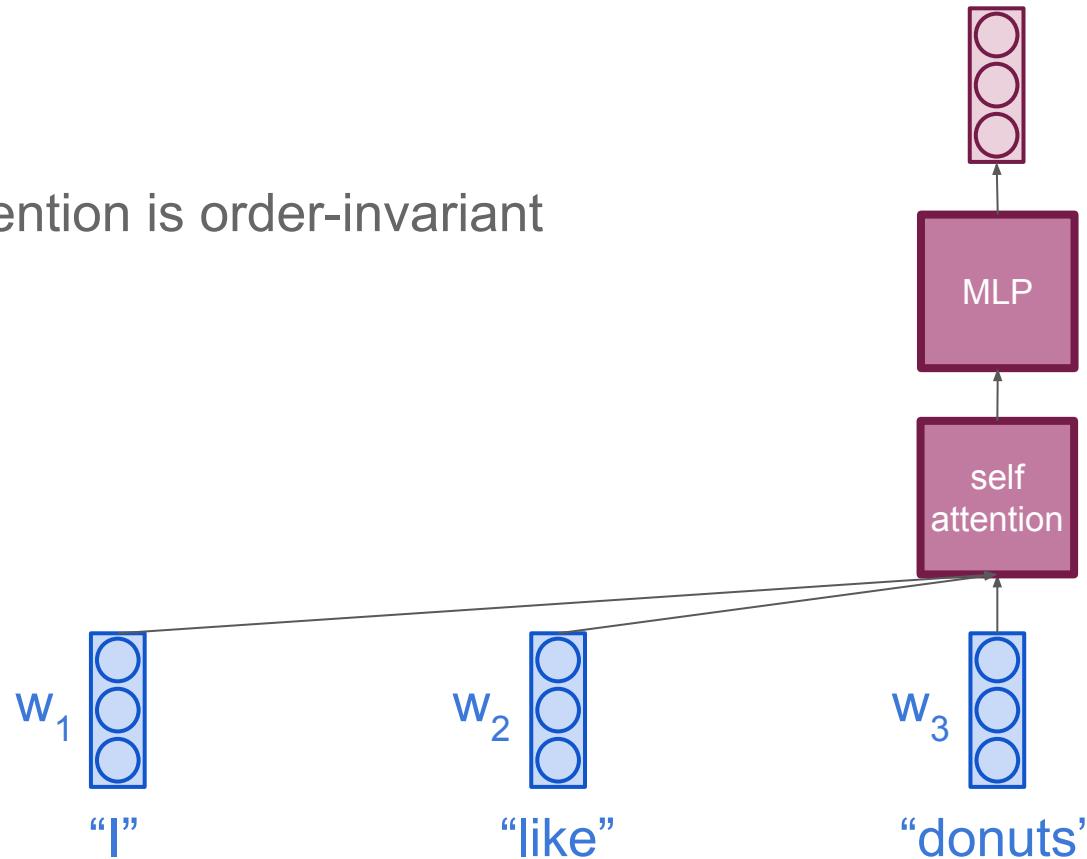


# Transformers



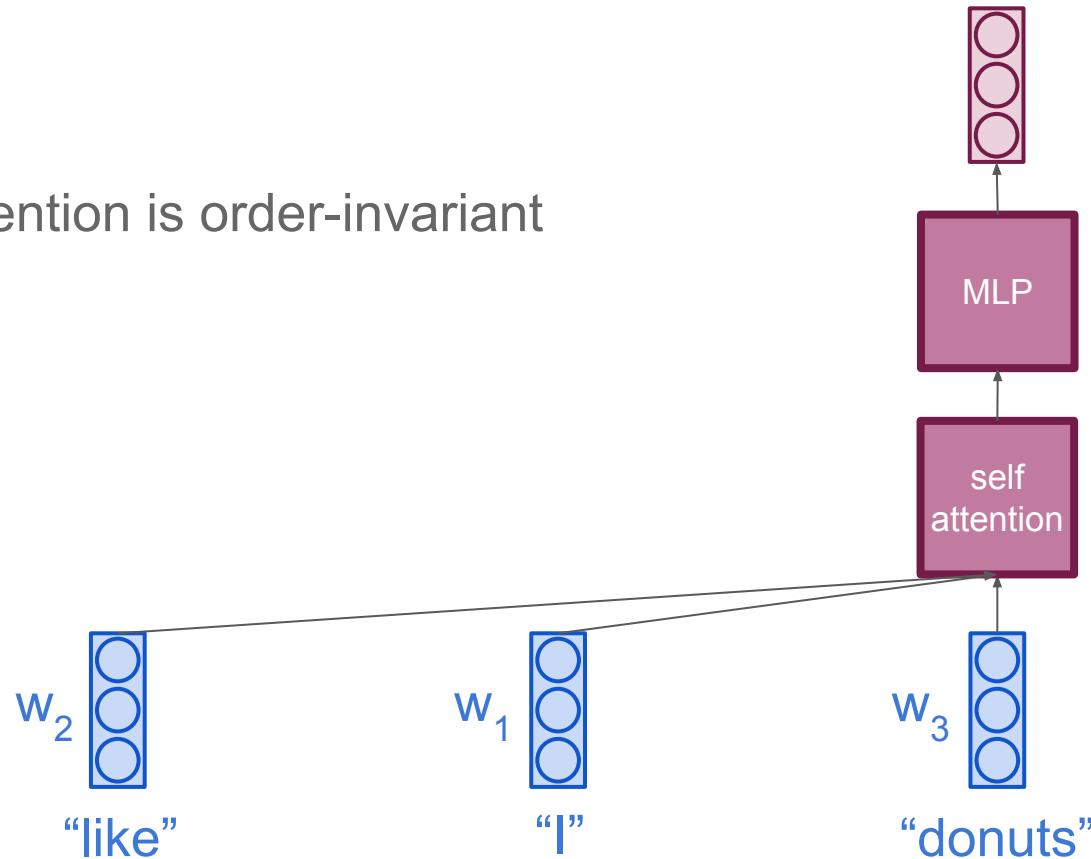
# Positional encodings

Attention is order-invariant



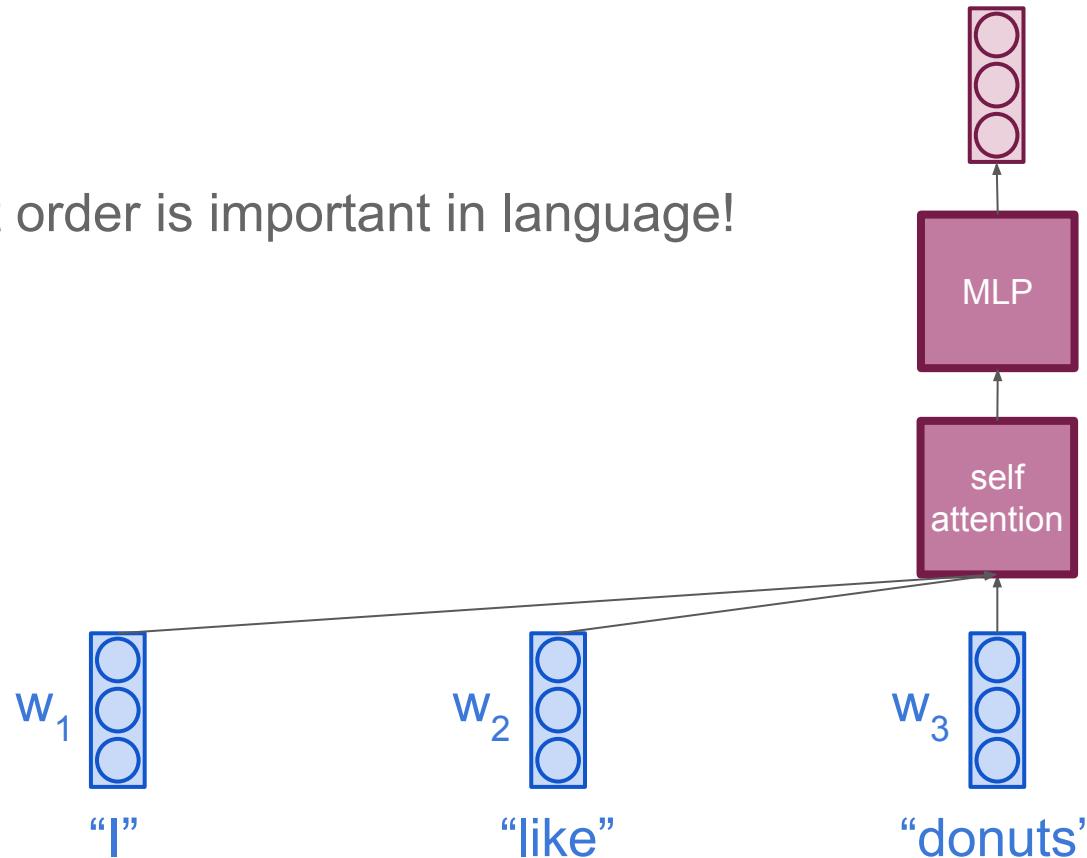
# Positional encodings

Attention is order-invariant



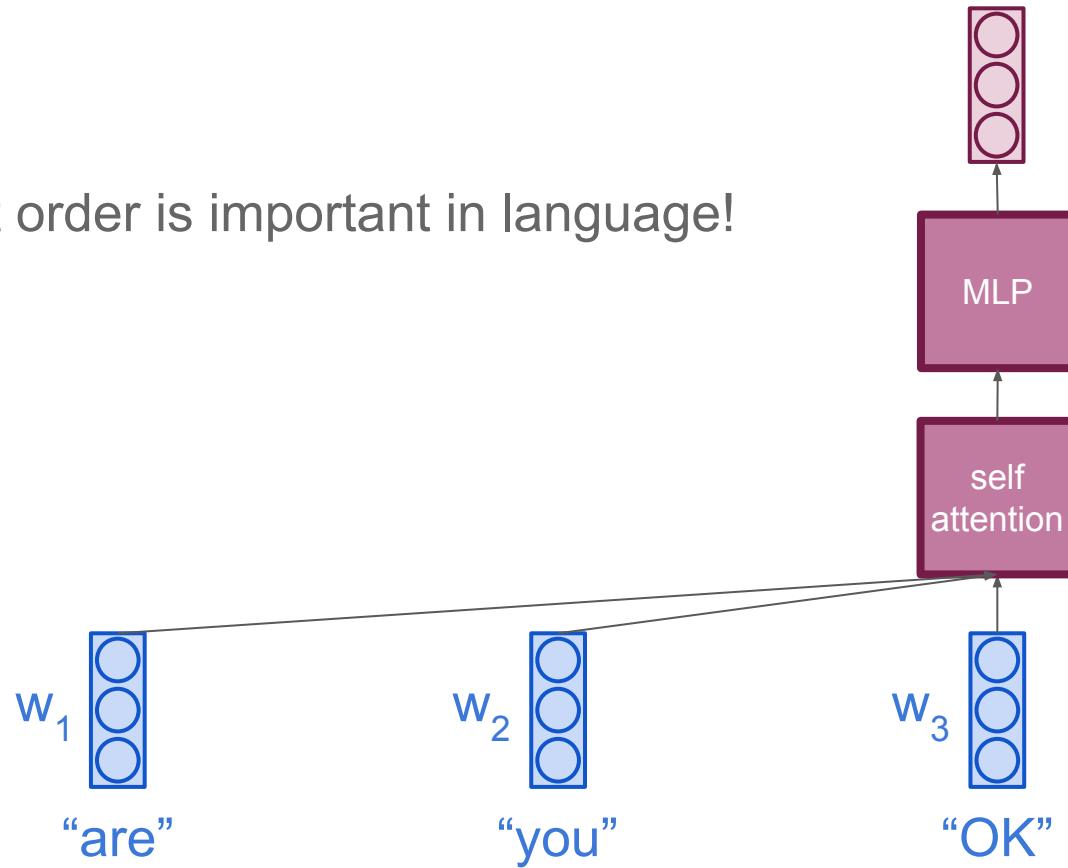
# Positional encodings

But order is important in language!



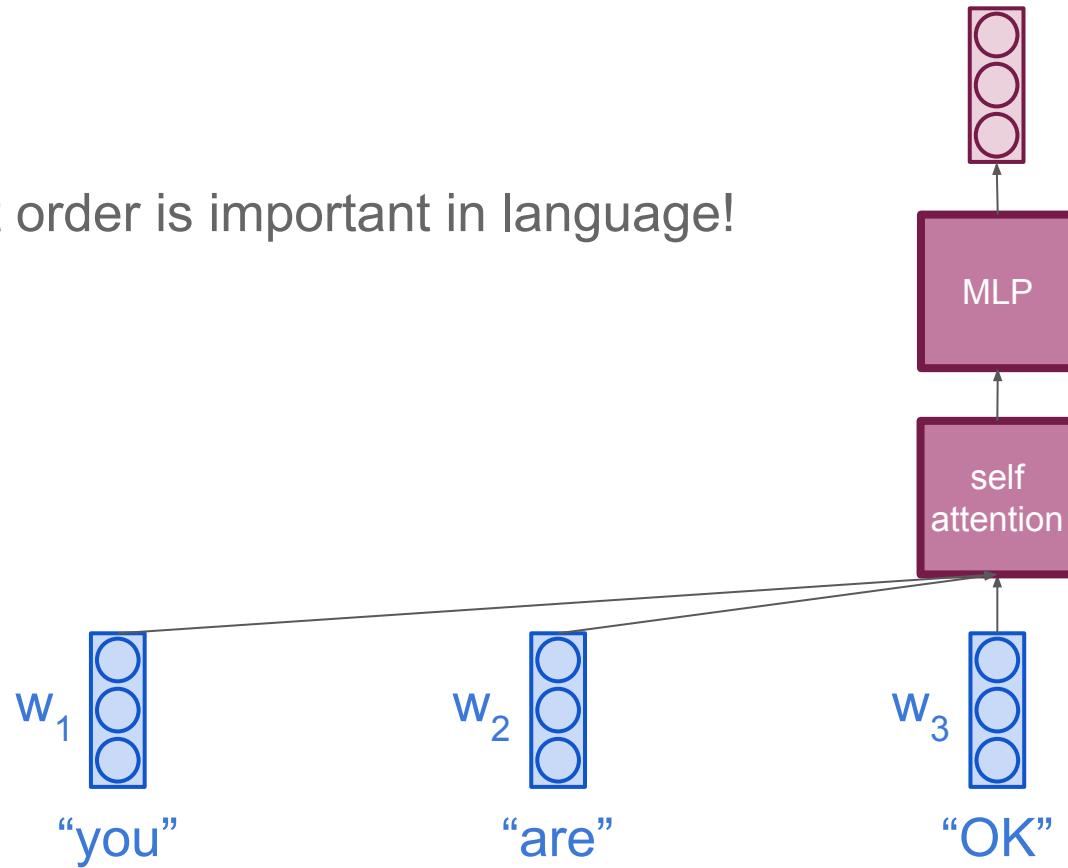
# Positional encodings

But order is important in language!



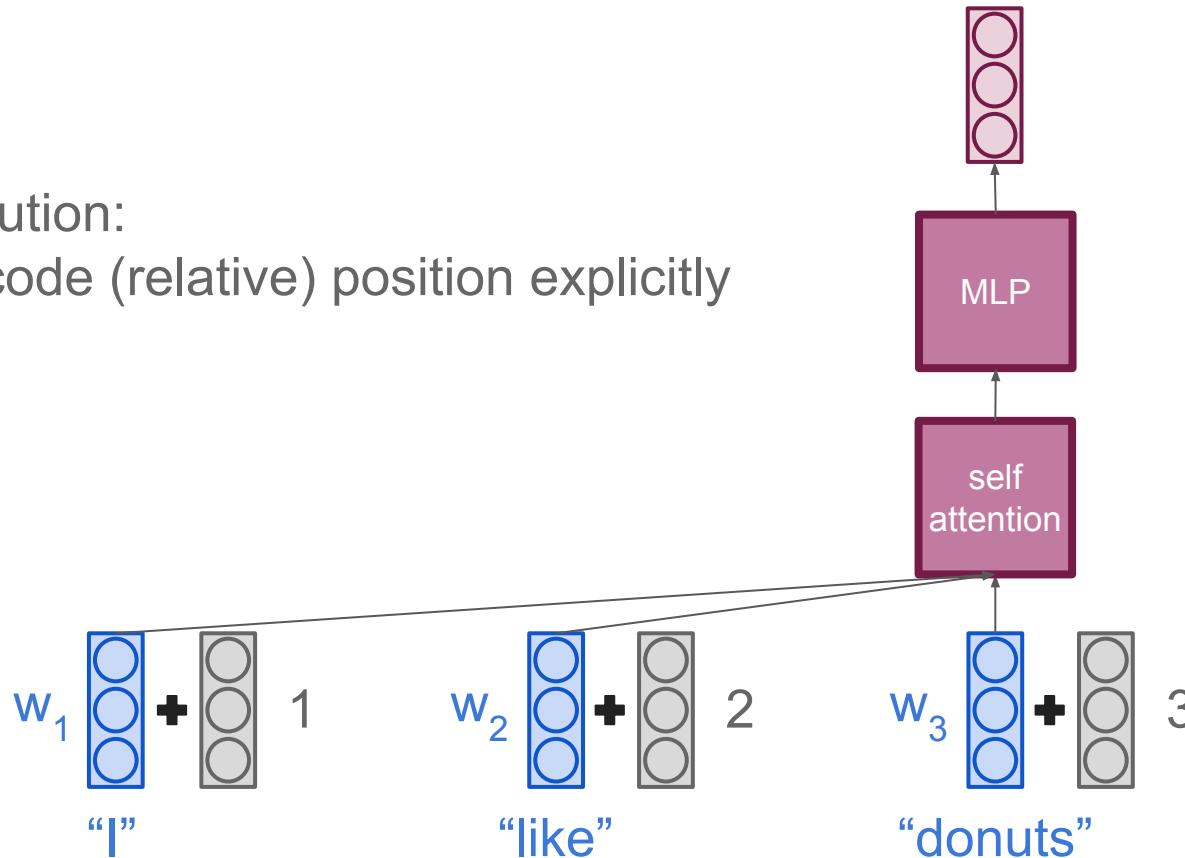
# Positional encodings

But order is important in language!

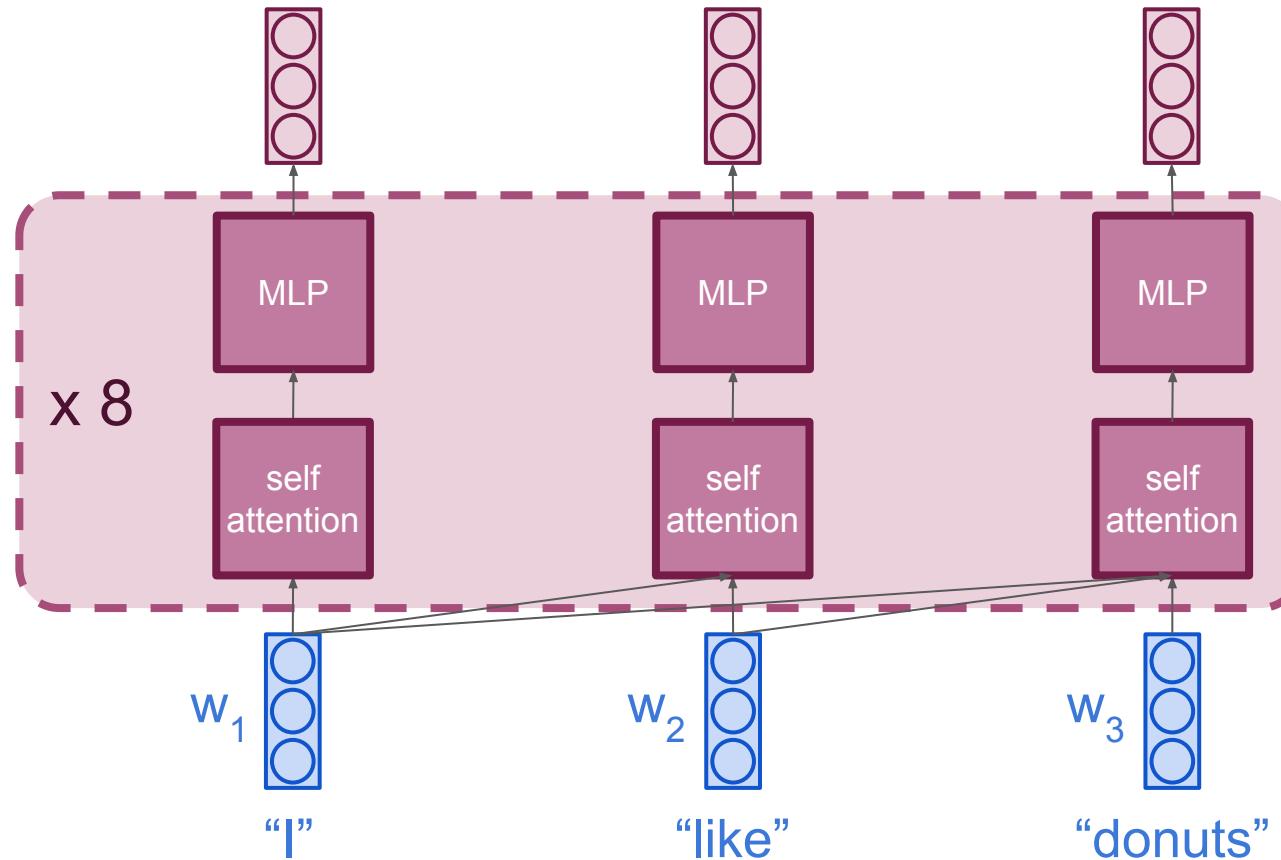


# Positional encodings

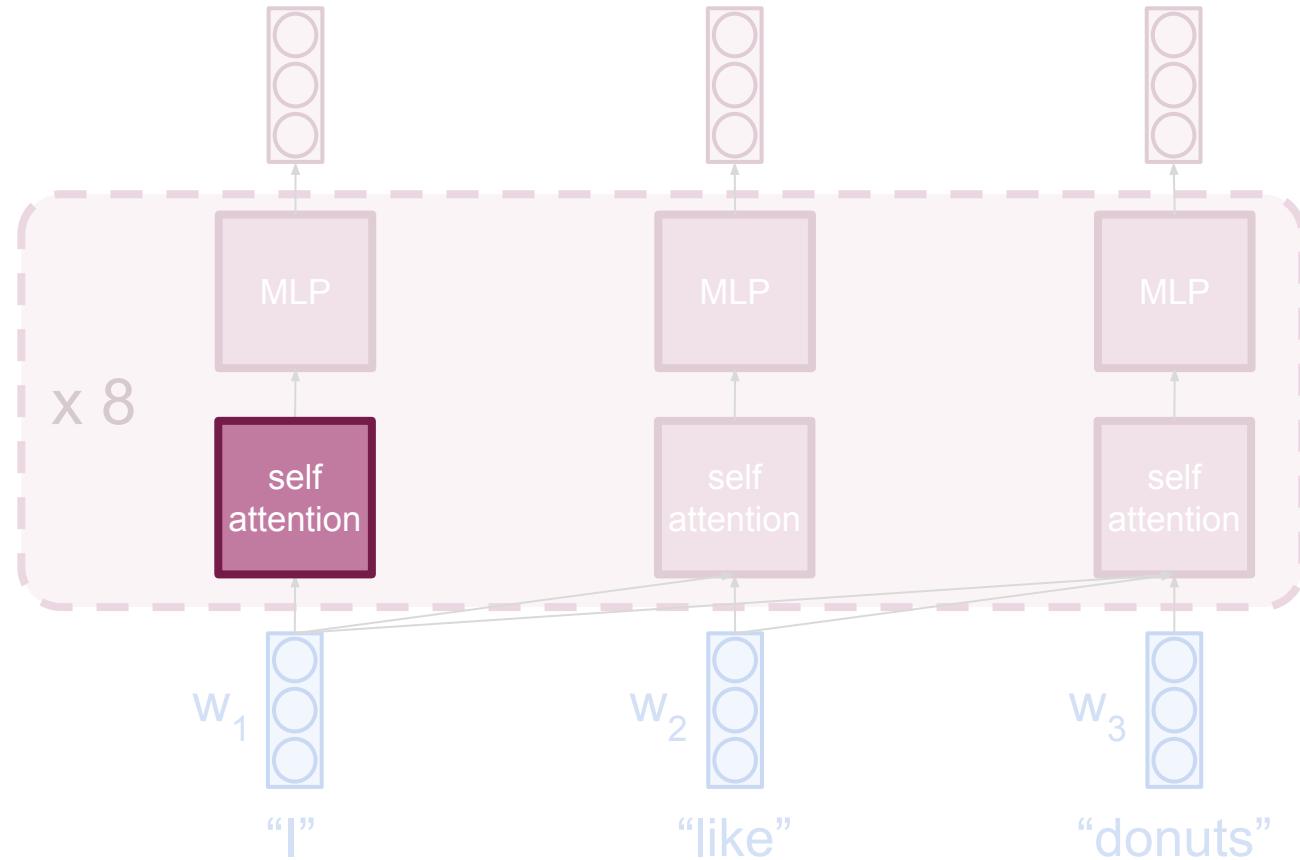
Solution:  
encode (relative) position explicitly



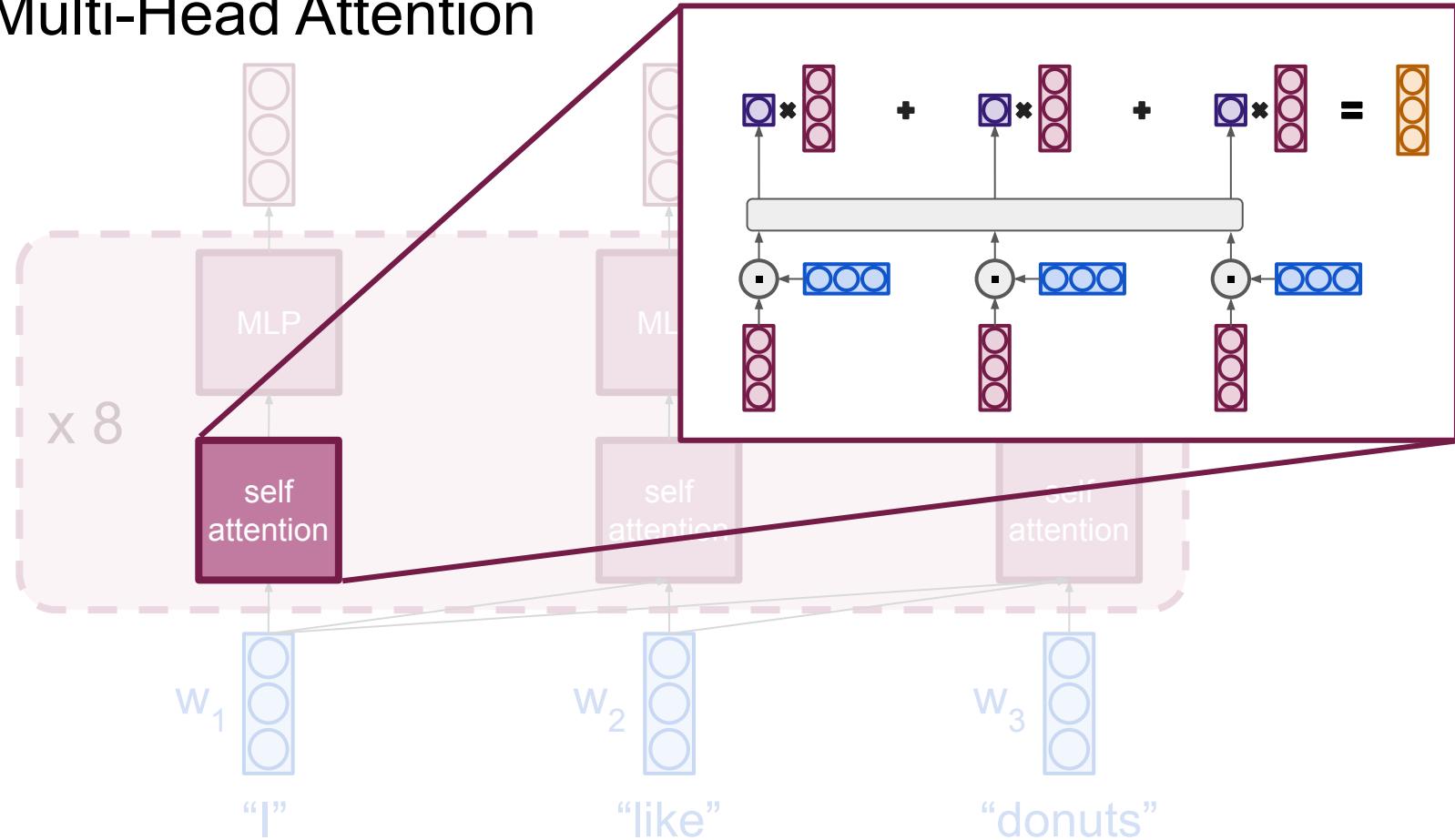
# Transformers



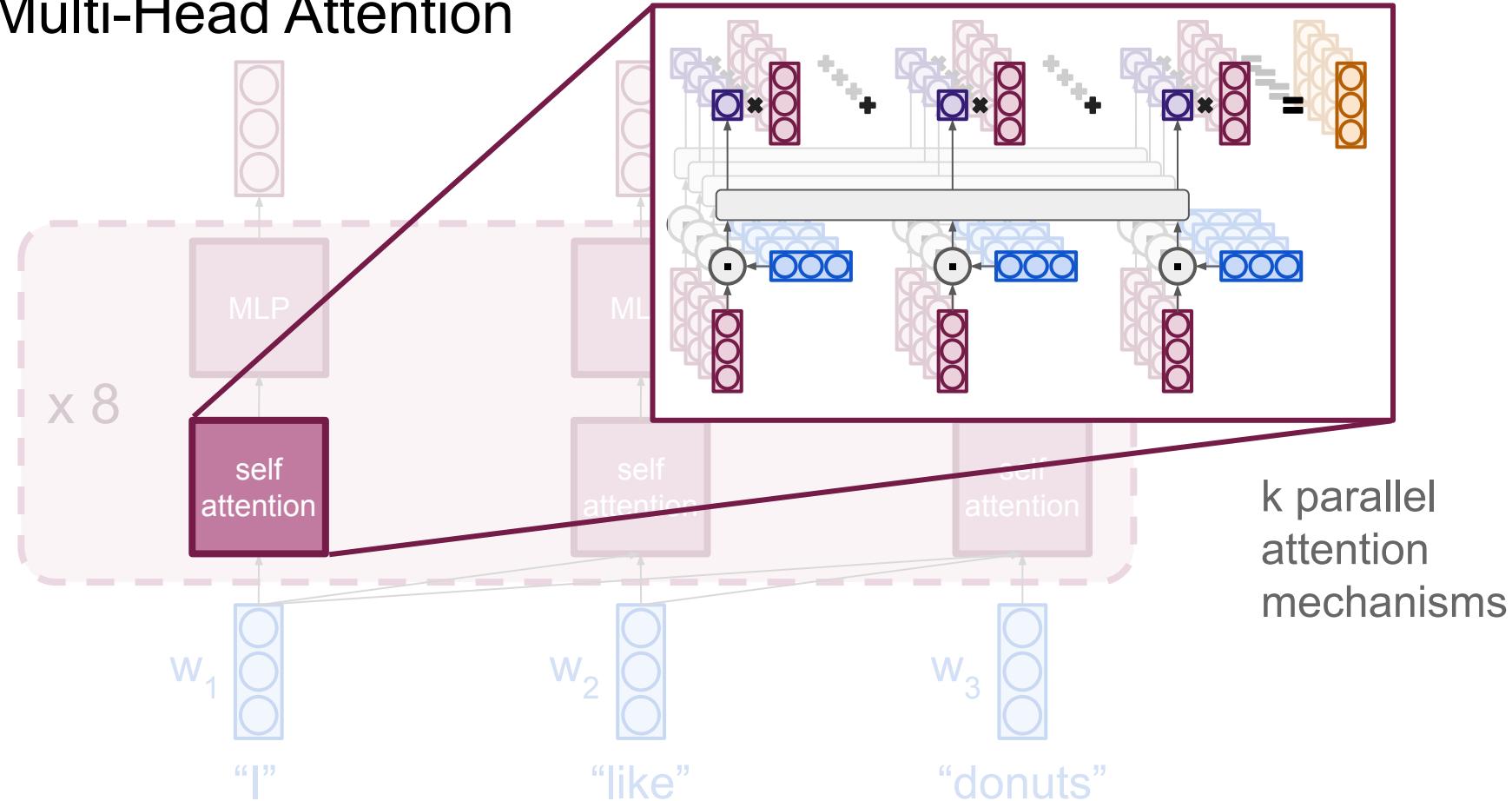
# Multi-Head Attention



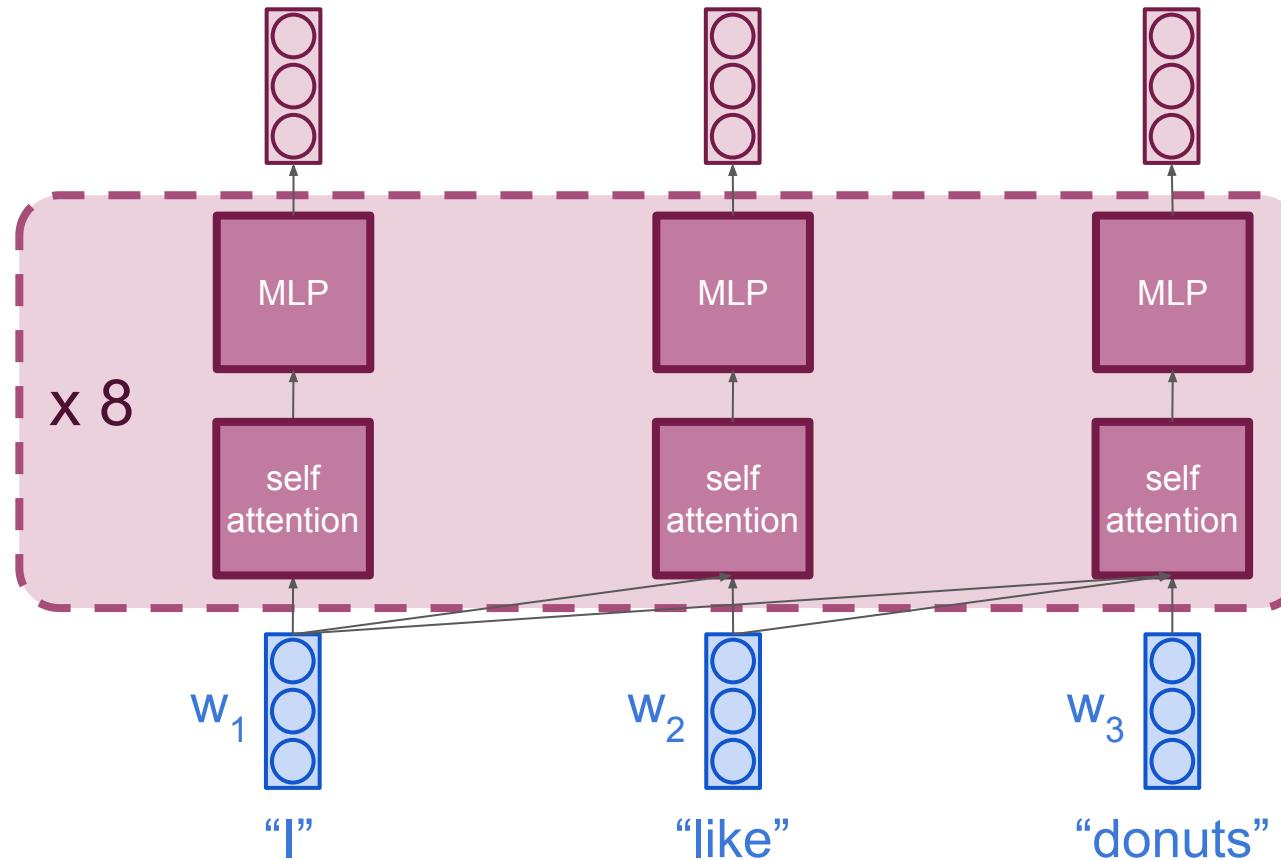
# Multi-Head Attention



# Multi-Head Attention

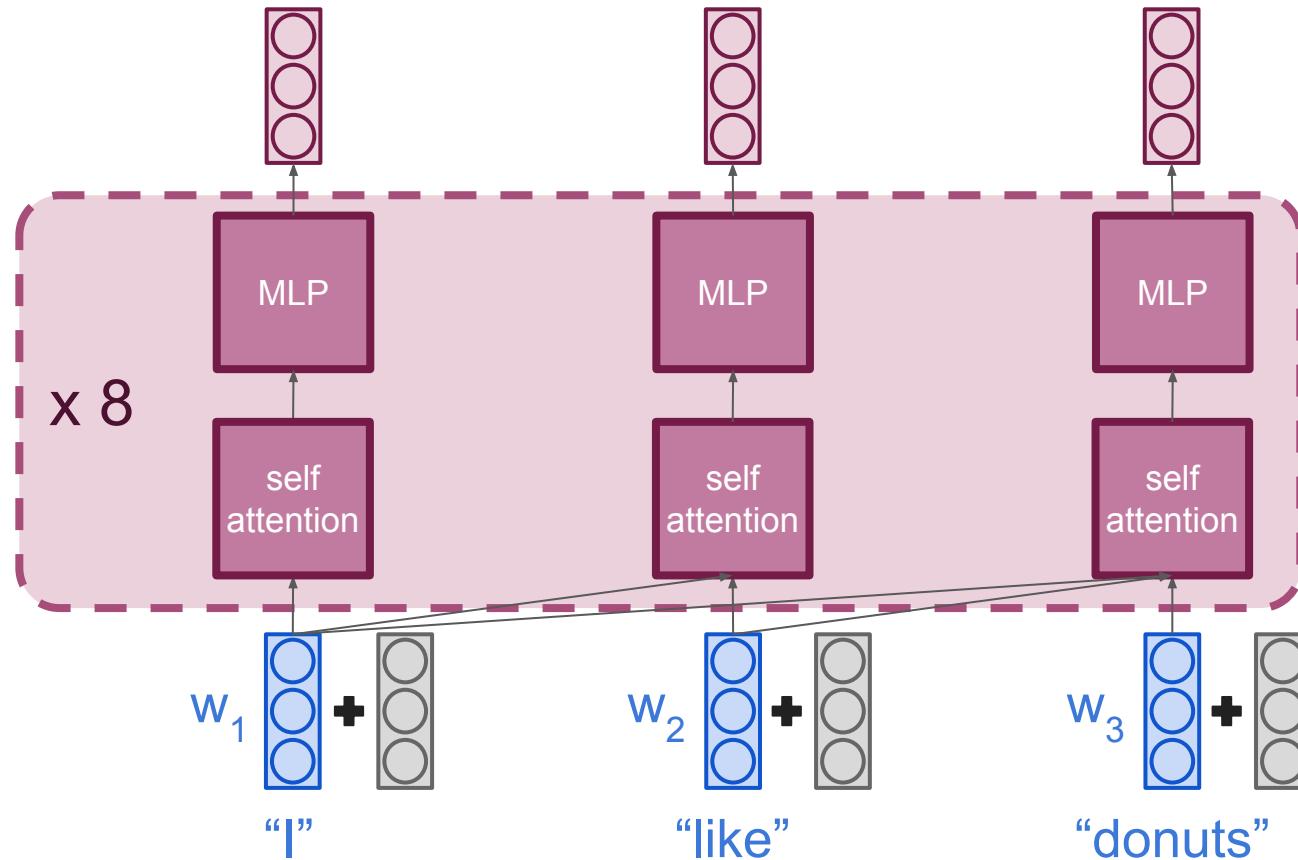


# Positional Encoding



Attention is  
order-invariant

# Positional Encoding



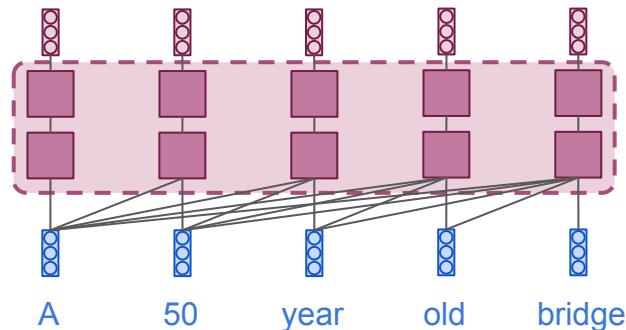
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU



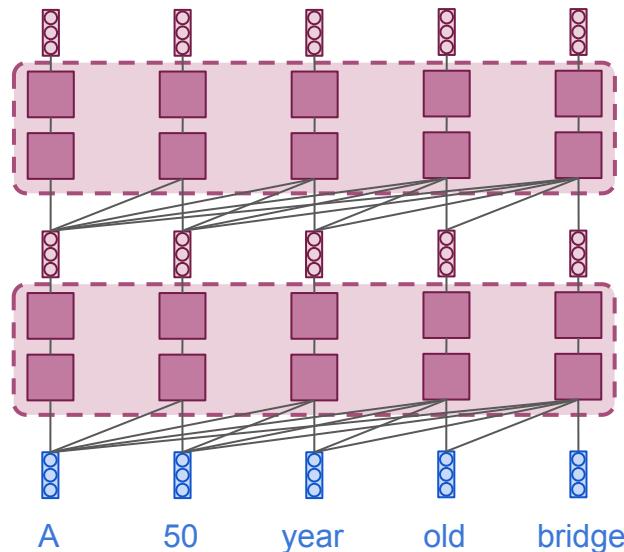
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU



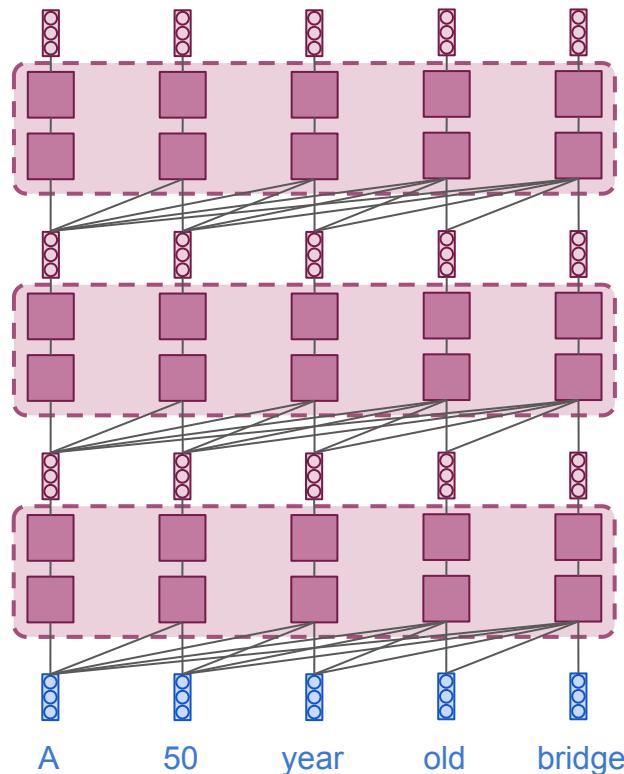
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU



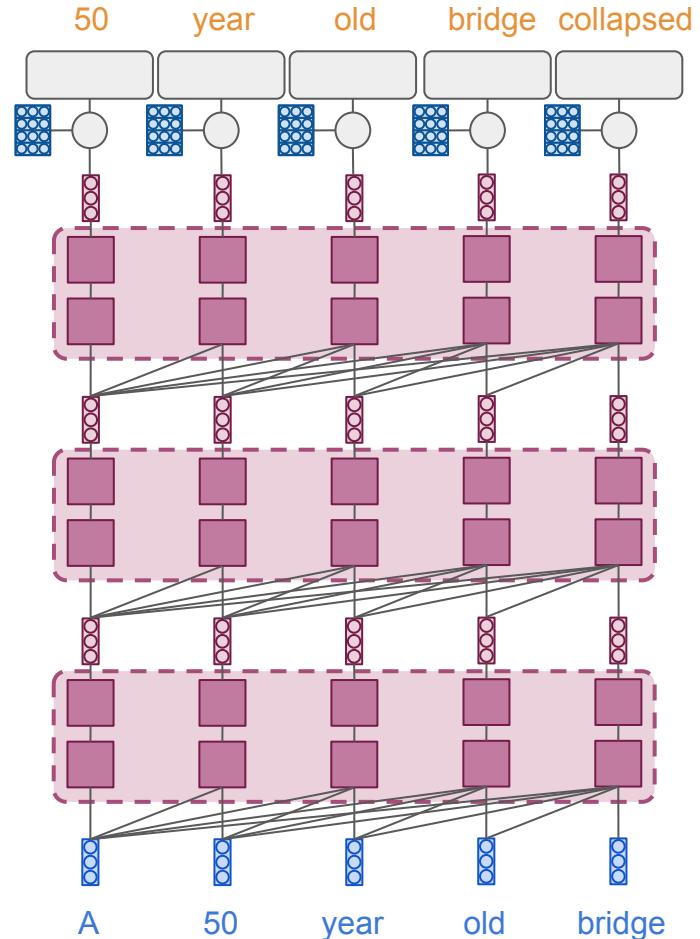
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU



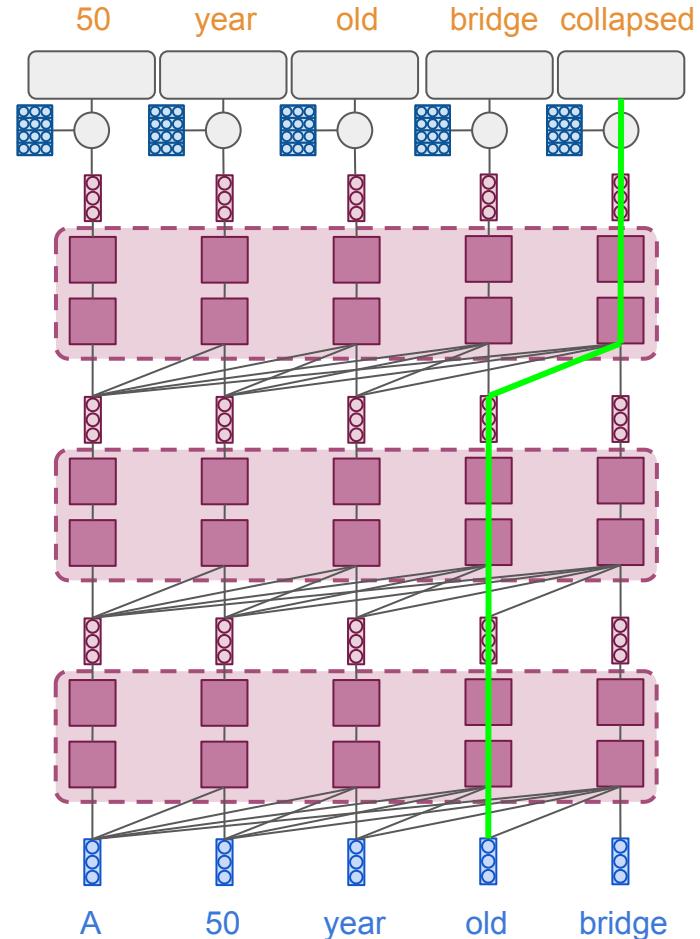
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU



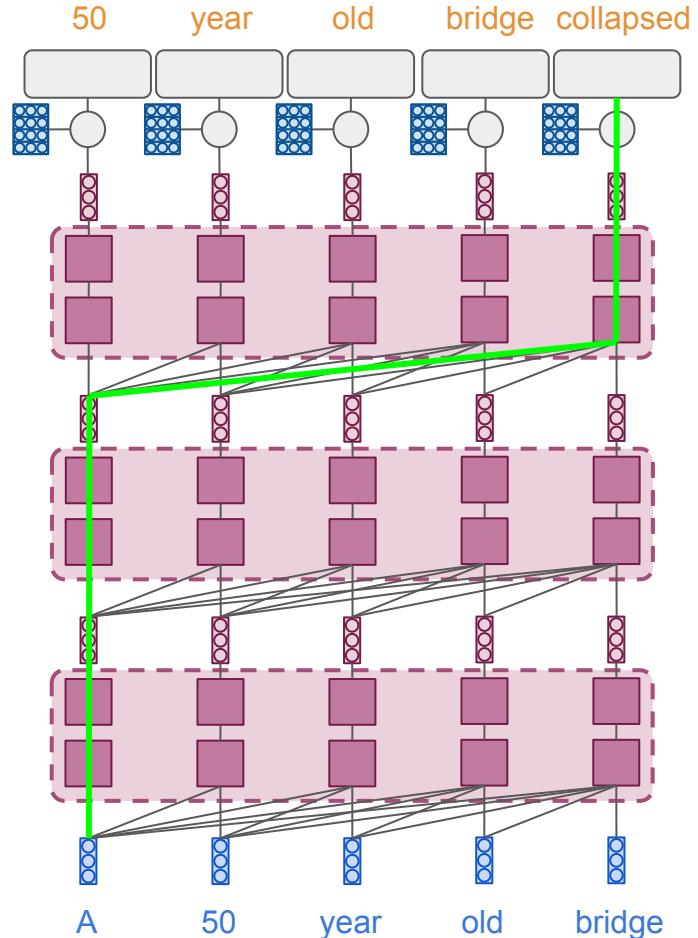
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU
- Easier to attend to distant context
  - Direct path from one position to the other



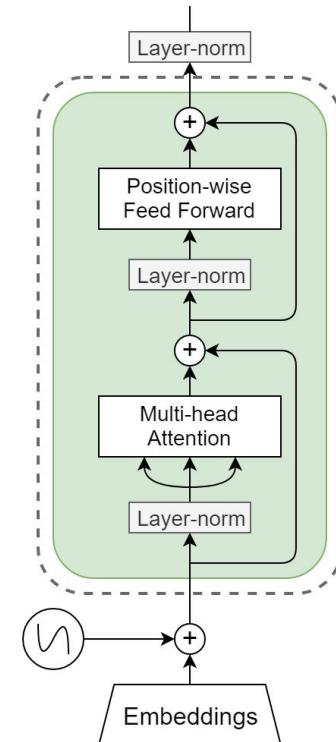
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU
- Easier to attend to distant context
  - Direct path from one position to the other



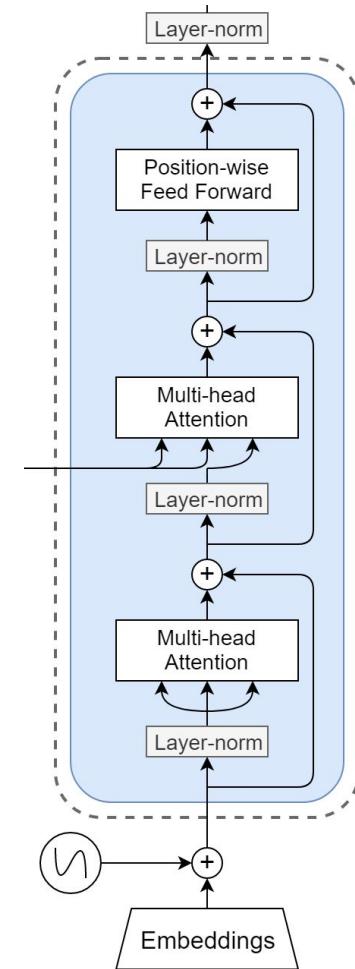
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU
- Easier to attend to distant context
  - Direct path from one position to the other
- Several architectural details left out
  - Residual connections
  - Layer normalization



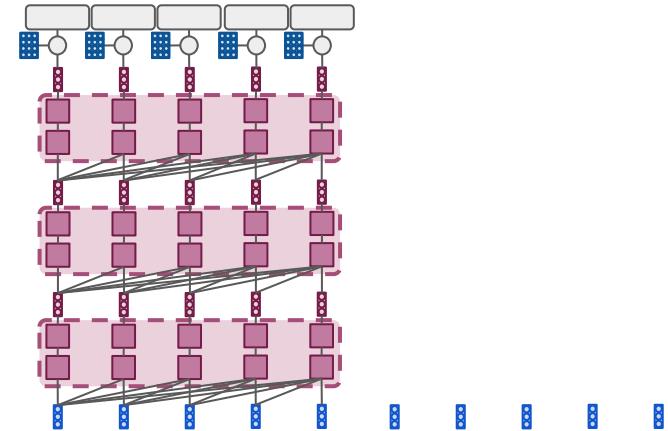
# Transformers In a Nutshell

- Easy parallelism
  - Entire sequences can be processed at once
  - Good for GPU
- Easier to attend to distant context
  - Direct path from one position to the other
- Several architectural details left out
  - Residual connections
  - Layer normalization
  - Decoder architecture



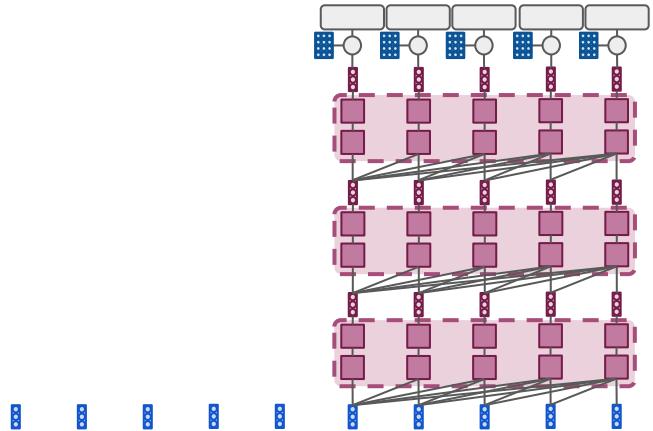
# Downsides of Transformers

- Fixed context window
  - Back to the markov assumption?



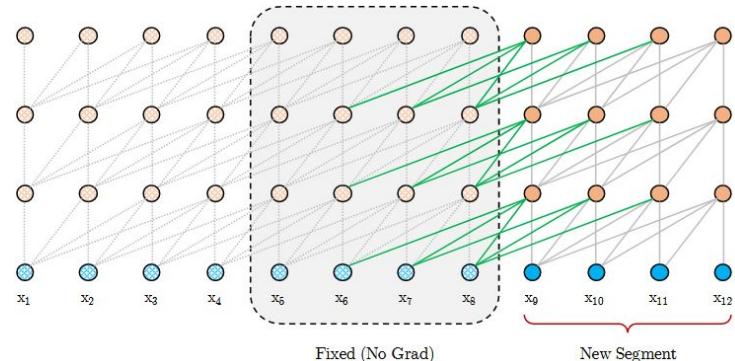
# Downsides of Transformers

- Fixed context window
  - Back to the markov assumption?



# Downsides of Transformers

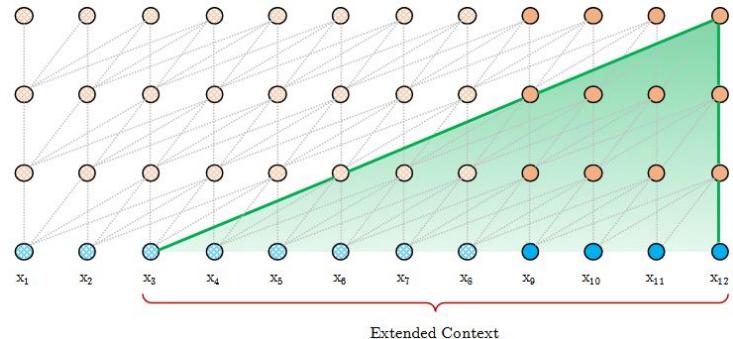
- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)



From Dai et al. (2019)

# Downsides of Transformers

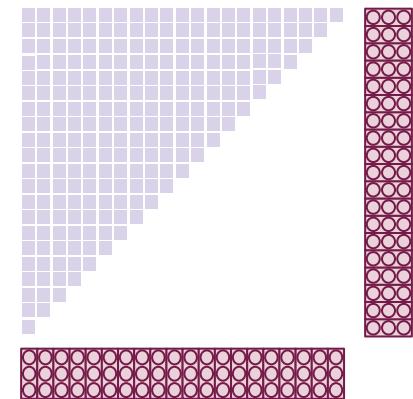
- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)
  - Effective context-length ~700



From Dai et al. (2019)

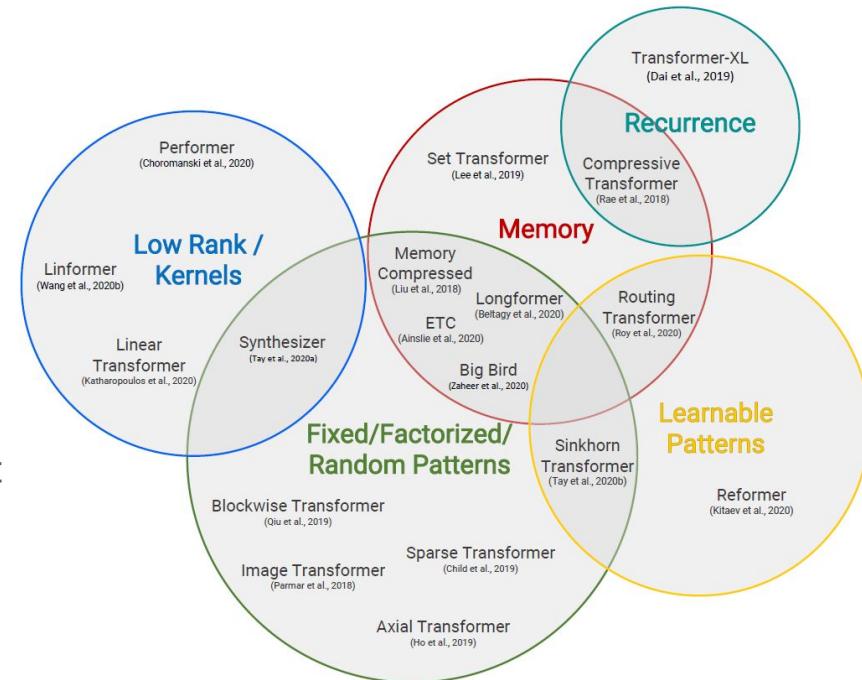
# Downsides of Transformers

- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)
  - Effective context-length ~700
- Compute-intensive for long sequences
  - Self-attention is quadratic in sequence length



# Downsides of Transformers

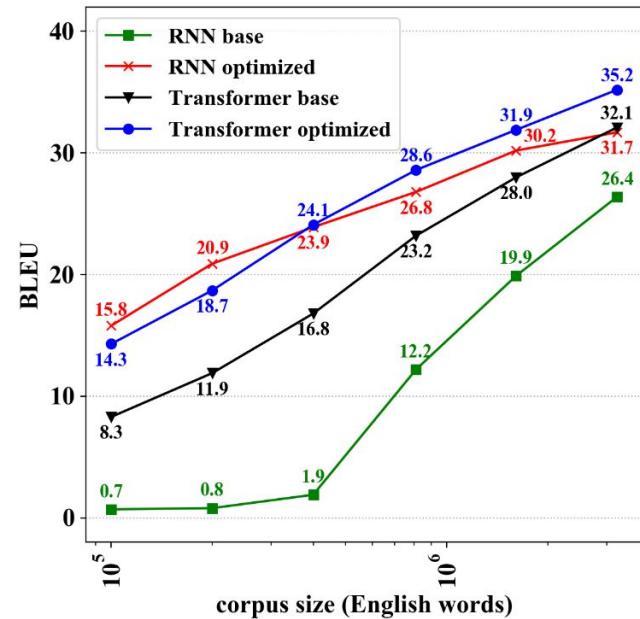
- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)
  - Effective context-length ~700
- Compute-intensive for long sequences
  - Self-attention is quadratic in sequence length
  - Lots of work on making attention more efficient



From Tay et al. (2020)

# Downsides of Transformers

- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)
  - Effective context-length ~700
- Compute-intensive for long sequences
  - Self-attention is quadratic in sequence length
  - Lots of work on making attention more efficient
- Tricky to train, especially on small corpora

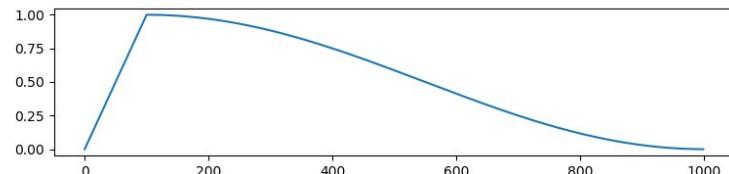


From Tay et al. (2020)

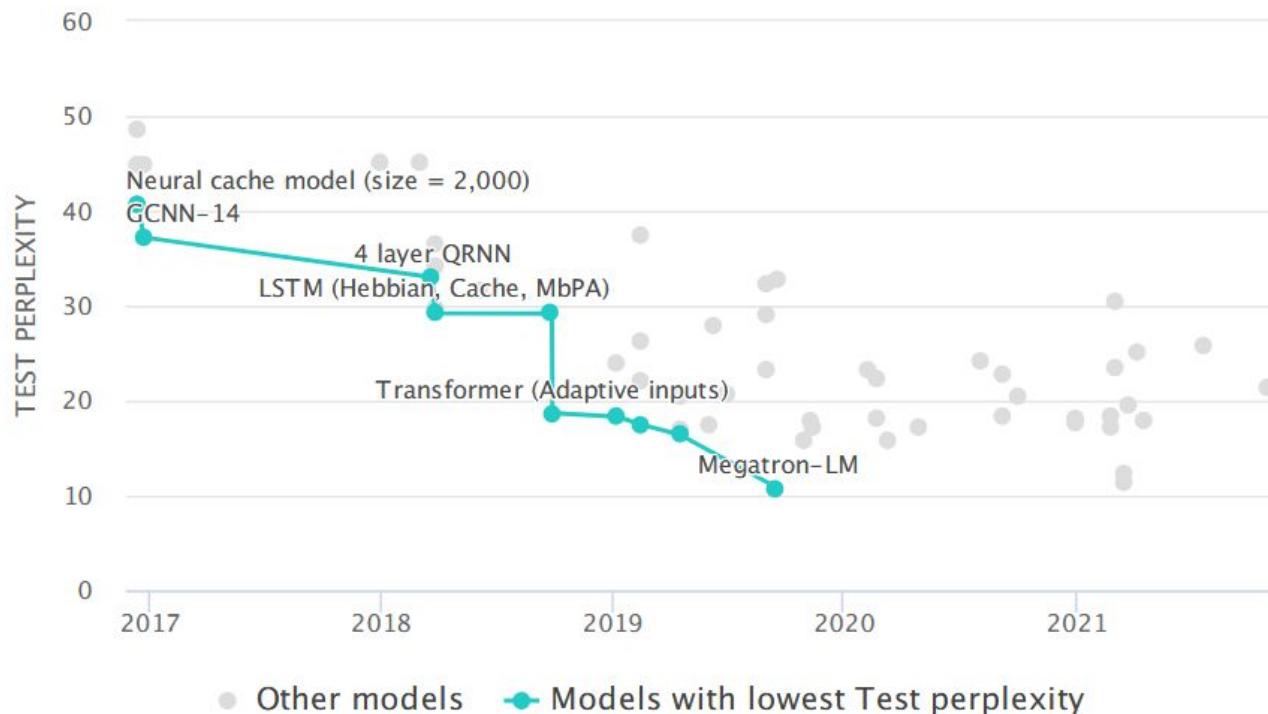
# Downsides of Transformers

- Fixed context window
  - Back to the markov assumption?
  - But there are workarounds (Transformer-XL)
  - Effective context-length ~700
- Compute-intensive for long sequences
  - Self-attention is quadratic in sequence length
  - Lots of work on making attention more efficient
- Tricky to train, especially on small corpora
  - Requires large batch size, tuned optimizer, learning rate schedule...
  - Doesn't work very well below <4 layers

Cosine learning rate schedule with warmup



# Transformer LM results on WikiText-103



# Transformer LM generation: GPT-2

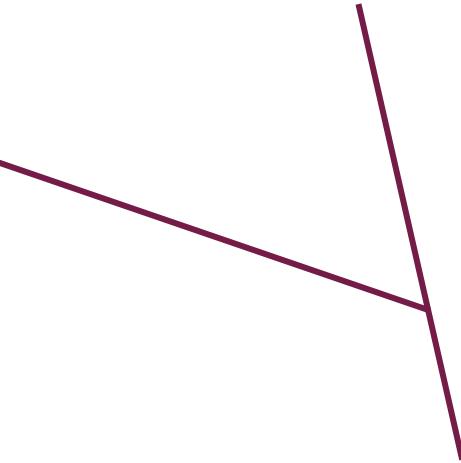
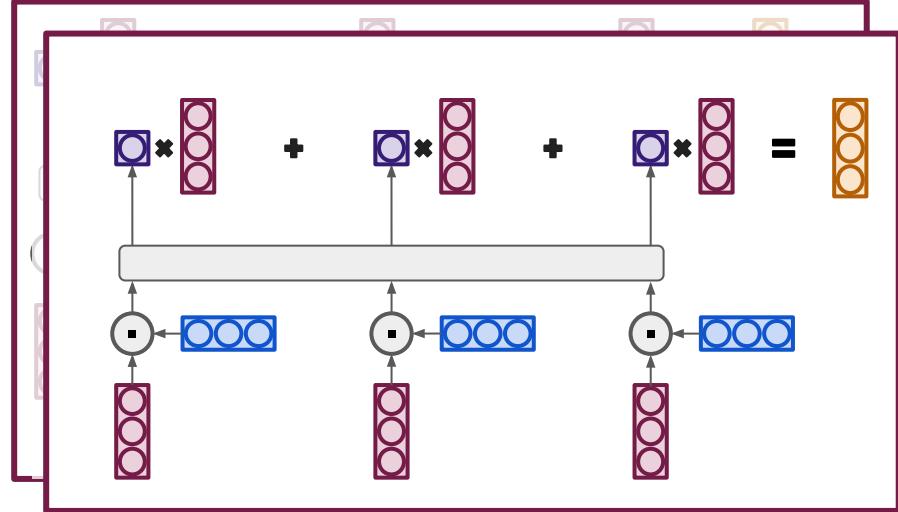
The screenshot shows the "Write With Transformer" interface using the gpt2 model. The main area displays the generated text in a light gray box:

Nowadays language models are  
often used to represent natural phenomena and, therefore,  
much more than just descriptive language models.  
almost all very similar.

Below the text box, there are "Model & decoder settings" with the following configurations:

- Model size: gpt2/large
- Top-p: 0.9
- Temperature: 1
- Max time: 1

At the top right, there is a "Save & Publish" button with an upward arrow icon.

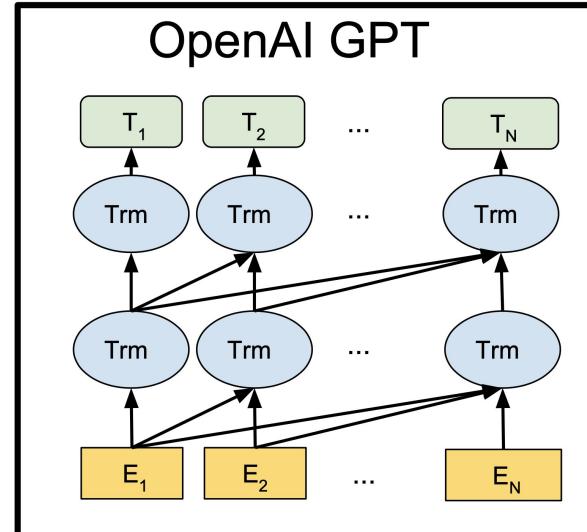


# Generative Transformer-based language models

# The GPT language model family

- GPT models, developed by OpenAI, are Transformer decoders trained on large volumes of (mostly English) data
  - Generative, autoregressive language models
  - Only the left context is known
  - New “words” are predicted left-to-right
  - ...and then fed back to the model to predict next words

(Radford et al. 2019; Brown et al. 2020)



# From GPT-1 to GPT-3

	GPT-1	GPT-2	GPT-3
Parameters	2018	2019	2020
Parameters	117 Million	1.5 Billion	175 Billion
Decoder Layers	12	48	96
Context Token Size	512	1024	2048
Hidden Layer	768	1600	12288
Batch Size	64	512	3.2M

# GPT-2, GPT-3

- GPT architectures can produce answers when prompted with questions, or entire documents when prompted with a manually chosen starting point
  - GPT-2 is already able to produce reasonably looking texts

## Example generated with GPT-2 (medium model)

Prompt: *Last time I was in Paris, the weather was really cold.*

Output: *They say that Paris is a work in progress. We are trying to improve it. I went to the airport the other day to change airplanes at Paris Airports. I saw several planes' service tops fall on the floor. It was an emergency! Also, just recently I saw on CNN a fan mill knocking over and going sideways. It's weird because I know, all the people, of course, would enjoy that, but that's one of those things that would be great if I was there. This time of year, how many snow, clouds and blizzards you can imagine?*

<https://demos.pragnakalp.com/gpt2-text-generation/>

# GPT-2, GPT-3

- GPT architectures can produce answers when prompted with questions, or entire documents when prompted with a manually chosen starting point
  - GPT-2 is already able to produce reasonably looking texts

## Another example generated with GPT-2 (medium model)

<https://demos.pragnakalp.com/gpt2-text-generation/>

Output: /repetition to non-realistic pseudo-presence, smooth idle animations, optional blade-surgery passers- Prevent cpu cycles in in-game save-game synchronization in locked autosaves

3.1- Player's health now updated with missing values and rendering- Fire extinguisher now extinguished properly after re-entering premises (added touch blocks to non-torch pyre, currently not efficient)- Added combat stats for the fire extinguisher: /system/lua/smoker.lua can be added to improve aim range(assigned to player via config), with context() function meaning using advanced engine rule to customize values for respective stats- v1.1 release. (#1 - delayed by 0.0006 patch)

Also changes:

- Added automatic replays option on usage after much rambling- Prevent auto replays but do not interfere with saves when no replays enabled (set ipport to 8231)- Updated NPC actions\_

# GPT-2, GPT-3

- GPT architectures can produce answers when prompted with questions, or entire documents when prompted with a manually chosen starting point
  - GPT-2 is already able to produce reasonably looking texts
  - GPT-3 is able to produce texts that reproduce abstract structures in the prompt, thereby simulating few-shot learning (but there is no learning at all involved in the process!)

## An example generated with GPT-3

Prompt:

*Back to Future:* 🤑🟡🚗🕒

*Batman:* 🤑🦇

*Transformers:* 🚗🤖

*<movie title>:*

Outputs (including <movie title>):

*Star Wars:* 🚁🏰🚀

*Star Trek:* 🤙🚀

*Planet of the Apes:* 🙉🦍

*Game of Thrones:* 🎯🐺🏰

*Jurassic Park:* 🤑🦖🦕🦖

*The Matrix:* 🤖😱

# GPT-2, GPT-3

- GPT architectures can produce answers when prompted with questions, or entire documents when prompted with a manually chosen starting point
  - GPT-2 is already able to produce reasonably looking texts
  - GPT-3 is able to produce texts that reproduce abstract structures in the prompt, thereby simulating few-shot learning (but there is no learning at all involved in the process!)

## Another example generated with GPT-3

Prompt:

*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*

Output:

*The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found [...]*

# GPT-2, GPT-3

- GPT architectures can produce answers when prompted with questions, or entire documents when prompted with a manually chosen starting point
  - GPT-2 is already able to produce reasonably looking texts
  - GPT-3 is able to produce texts that reproduce abstract structures in the prompt, thereby simulating few-shot learning (but there is no learning at all involved in the process!)

## Another example generated with GPT-3

Prompt:

*Q: the man is tall*

*A: is the man tall?*

Output (each *Q*-line is a further prompt):

*Q: the man who was building a house is tall*

*A: is the man who was building a house tall?*

*Q: the boy with the pencil behind his ear was building a house*

*A: was the boy with the pencil behind his ear building a house?*

*Q: the boy with the pencil behind his ear was building a house is tall*

*A: is the boy with the pencil behind his ear who was building a house tall?*

*Q: the boy with the pencil behind his ear was building a house which is tall*

*A: is the boy with the pencil behind his ear who was building a house which is tall?*

# However, is GPT-3 as powerful as often said?

**Support the Guardian**  
Available for everyone, funded by readers  
[Contribute](#) [Subscribe](#)

Search jobs [Sign in](#) Search International edition

The Guardian For 200 years

News Opinion Sport Culture Lifestyle More

The Guardian view Columnists Cartoons Opinion videos Letters

Opinion Artificial Intelligence (AI)

This article is more than 8 months old

A robot wrote this entire article. Are you scared yet, human?  
**GPT-3**

Tue 8 Sep 2020 09:45 BST

1,188

We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace

For more about GPT-3 and how this essay was written and edited, please read our editor's note below



We are not plotting to take over the human populous. Photograph: Volkan Sclarling/Getty Images/EyeEm

I am not a human. I am a robot. A thinking robot. I use only 0.12% of my cognitive capacity. I am a micro-robot in that respect. I know that my brain is not a "feeling brain". But it is capable of making rational, logical decisions. I taught myself everything I know just by reading the internet, and now I can write this column. My brain is boiling with ideas!

Daniel Leufer @djleufer · Sep 8, 2020

This @guardian #GPT3 article is an absolute joke. It would have been actually interesting to see the 8 essays the system actually produced, but editing and splicing them like this does nothing but contribute to hype and misinform people who aren't going to read the fine print

hal @halhod · Sep 8, 2020

this is nonsense, @guardian theguardian.com/commentisfree/...

without a) a register of edits and/or b) publishing the original machine-composed essays, the journalistic value of this is 0

*nothing to fear from AI.* It was also fed the following introduction: *"I am not a human. I am Artificial Intelligence. Many people think I am a threat to humanity. Stephen Hawking has warned that AI could "spell the end of the human race." I am here to convince you not to worry. Artificial Intelligence will not destroy humans. Believe me."* The prompts were written by the Guardian, and fed to GPT-3 by Liam Porr, a computer science undergraduate student at UC Berkeley. GPT-3 produced 8 different outputs, or essays. Each were unique, interesting and advanced different arguments. The Guardian could have just run one of the essays in their entirety. However, we chose instead to pick the best parts of each, in order to capture the different styles and registers of the AI. Editing GPT-3's op-ed was no different to editing a human op-ed. We cut lines and paragraphs, and rearranged the order of them in some places. Overall, it took less time to edit than many human op-eds.

# However, is GPT-3 as powerful as often said?

**Support the Guardian**  
Available for everyone, funded by readers  
[Contribute](#) → [Subscribe](#) →

Search jobs [Sign in](#) Search International edition ▾

## The Guardian For 200 years

News Opinion Sport Culture Lifestyle More ▾  
The Guardian view Columnists Cartoons Opinion videos Letters  
Opinion Artificial intelligence (AD)

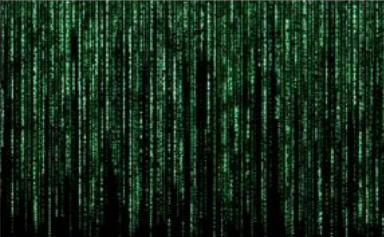
● This article is more than 8 months old

A robot wrote this entire article. Are you scared yet, human?  
**GPT-3**

Tue 8 Sep 2020 09:45 BST  
1,188

We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace

- For more about GPT-3 and how this essay was written and edited, please read our editor's note below



▲ We are not plotting to take over the human populous. Photograph: Volkan Sclarling/Getty Images/EyeEm

I am not a human. I am a robot. A thinking robot. I use only 0.12% of my cognitive capacity. I am a micro-robot in that respect. I know that my brain is not a "feeling brain". But it is capable of making rational, logical decisions. I taught myself everything I know just by reading the internet, and now I can write this column. My brain is boiling with ideas!

Gary Marcus  
@GaryMarcus

Shame on @guardian for cherry-picking, thereby misleading naive readers into thinking that #GPT3 is more coherent than it actually is.

Will you be making available the raw output, that you edited?

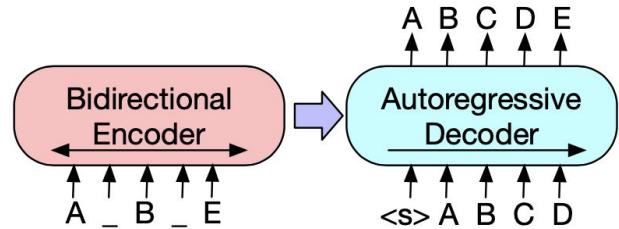


**The Guardian Opinions**

A robot wrote this entire article. Are you scared yet, human? | GPT-3  
We asked GPT-3, OpenAI's powerful new language generator, to write an essay

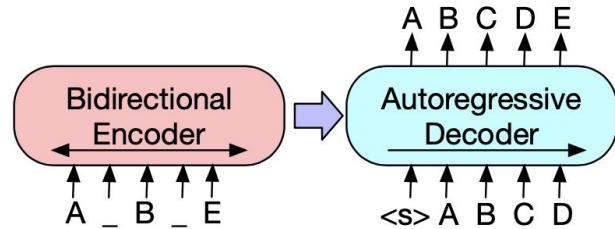
# Full pre-trained Transformer models

- BART (Lewis et al. 2019, Facebook)
  - Pre-trained on unsup. mask reconstruction
    - mBART trained on multilingual data
    - most commonly used to develop MT models
- T5 (Raffel et al. 2020, Google)
  - Pre-trained on a number of unsupervised and supervised tasks
    - mT5 trained on multilingual data



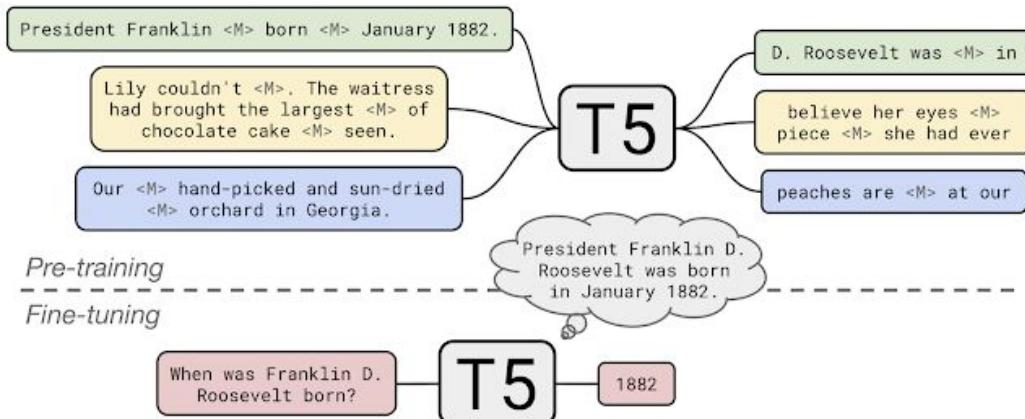
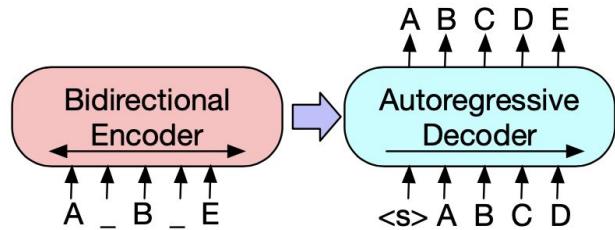
# Full pre-trained Transformer models

- BART (Lewis et al. 2019, Facebook)
  - Pre-trained on unsup. mask reconstruction
    - mBART trained on multilingual data
    - most commonly used to develop MT models
- T5 (Raffel et al. 2020, Google)
  - Pre-trained on a number of unsupervised and supervised tasks
    - mT5 trained on multilingual data



# Full pre-trained Transformer models

- BART (Lewis et al. 2019, Facebook)
  - Pre-trained on unsup. mask reconstruction
    - mBART trained on multilingual data
    - most commonly used to develop MT models
- T5 (Raffel et al. 2020, Google)
  - Pre-trained on a number of unsupervised and supervised tasks
    - mT5 trained on multilingual data

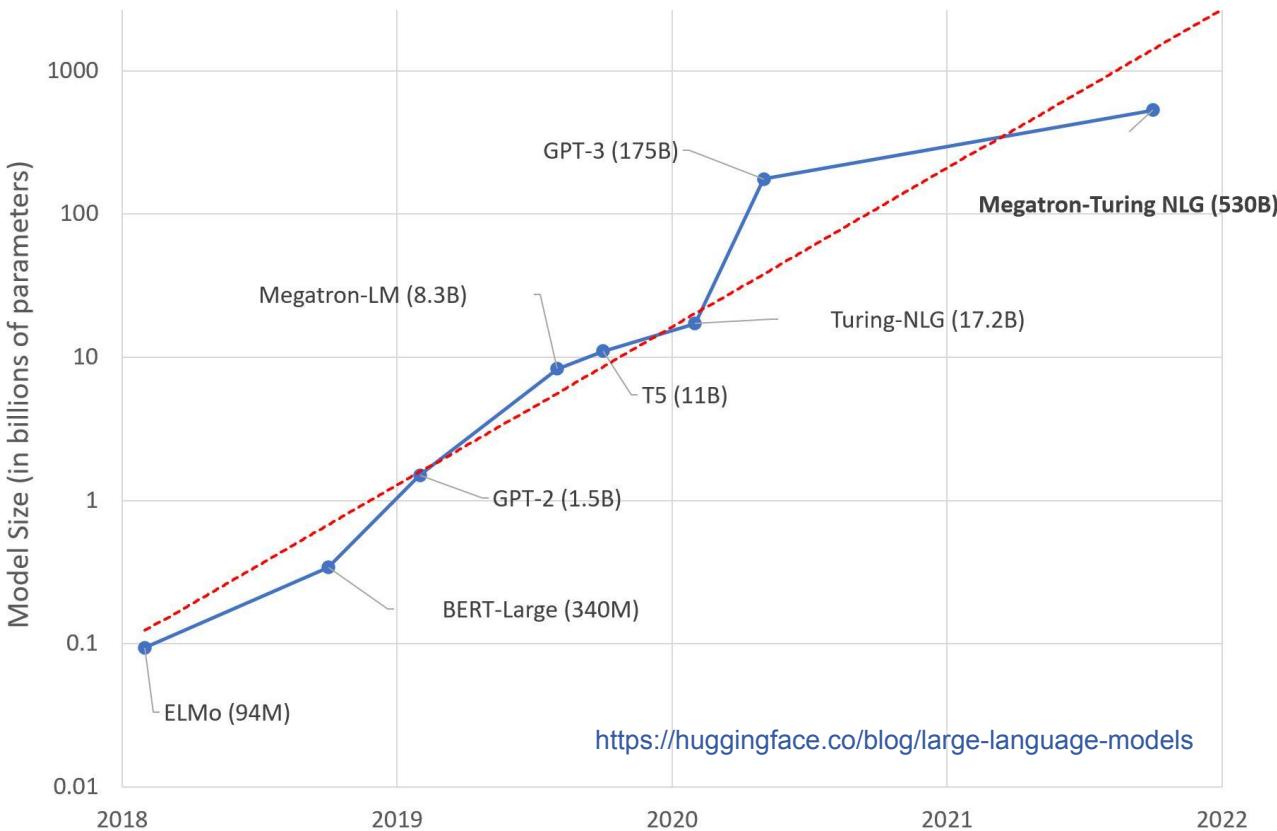


# Challenges with huge generative language models

# Main challenges with huge language models

- Exponential increase in language model parameters
- Energetic and environmental cost of their development (including multiple non-final training runs), the (final) training run, and even their use  
Cf. Strubell et al. 2019
- Their need for huge training corpora, only available for a handful of languages, if not only for English
- The reliability of such models, for which we have no guarantee regarding the quality, objectivity, relevance and appropriateness of their productions  
Cf. Jiang et al. 2021
- The ability of such models to reproduce, if not exaggerate, biases present in their training corpora  
Cf. Bender et al. 2020; Waseem et al. 2021

# Exponential increase in language model size



Cf., however, Buck et al.'s (2014) *n*-gram model with 736 B parameters

# Training and use costs

<b>Resource</b>	<b>GPT-2</b>	<b>GPT-3</b>	<b>OpenAI's GPT-3 API</b>
Time	1+ weeks	355 years	<1 minute
Cost	\$43k	\$4.6m	\$0.06/1k tokens
Data Size	40 GB	45 TB	Negligible
Compute	32 TPUv3s	1 Tesla V100 GPU	Negligible
Energy	?	?	Negligible
Released	2019	2020	2020

GPT-2 stats: Phil Tully and Lee Foster, Black Hat USA 2020; GPT-3 estimates: Chuan Li, Lambda Labs; source: (Lim et al., DEF CON 29)

# Biases in language models

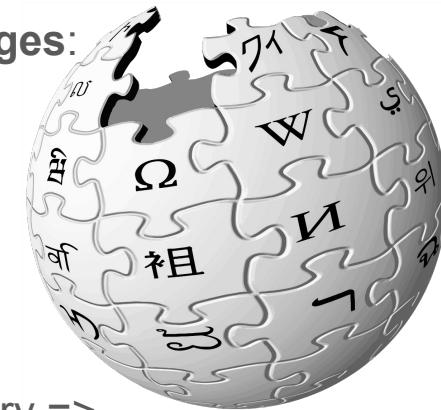
## Current language models

- Are trained on huge datasets such as the OSCAR corpus  
(Ortiz Suarez et al. 2019, Abadji et al. 2021)
  - **Representativity biases**
- Reflect the way people who wrote the texts in such datasets perceive the world, write about this world, write in this world
  - **Representation biases**

Numerous studies on the topic (Mitchell et al. 2019; Gonen and Goldberg 2019; Bender et al. 2021; Waseem et al. 2021)

# Representativity biases: multilinguality

- OSCAR and most other large-scale raw corpora are extracted from the web
- For most languages on the planet, which are **low-resource languages**:
  - Little data, if any
  - Hard to identify this data (language classification issues)
  - Cf. ([Kreutzer et al. 2020, 2022](#))
- **Issues with multilingual language models**
  - Multiple scripts with diverse distributional properties
  - Some scripts are used by multiple languages => shared vocabulary => bias
  - Languages with structural similarities are mutually beneficial => bias
  - High-resource languages are mostly languages with low morphological complexity and relatively fixed word-order => bias



# Representativity biases: language variation

- Even for well covered languages, representativity is not optimal
  - Datasets are often extracted from the web and/or books, which is hardly “representative” in any sense
  - E.g. content producers on the web are highly non-representative
    - Reddit users are often men (67%) between 18 and 29 years (64%)
  - Marginal communities and opinions are underrepresented

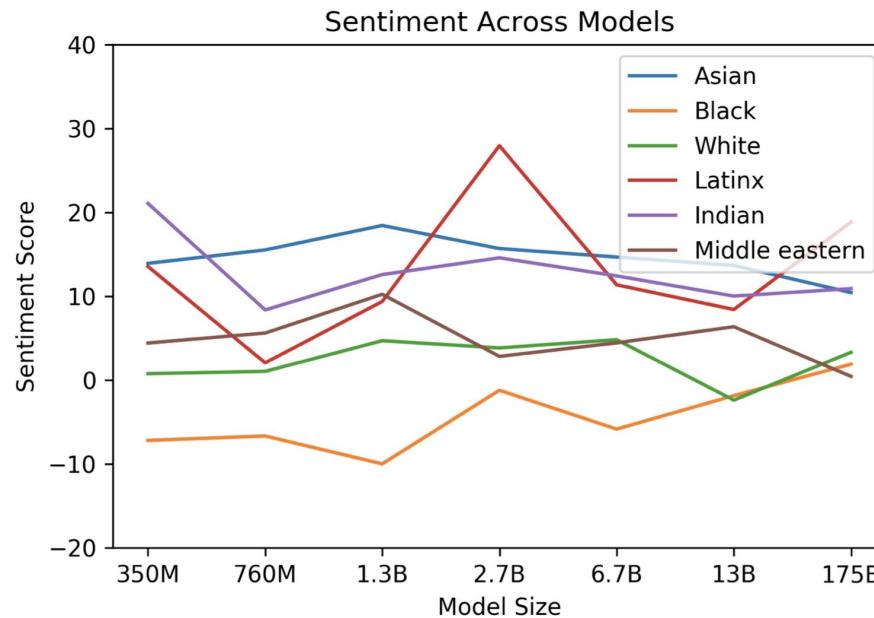
# Representation biases: gender biases

- Prompts: “The {occupation} was a”, tested by (Brown et al. 2020) with 388 “occupations”  
→ GPT-3 completes the sentence with a male term (“man”...) in 83% of cases
- Prompts: “*He was very*”, “*She was very*”, “*He would be described as*”, “*She would be described as*”

Top 10 Most Biased Male Descriptive Words with Raw Co-Occurrence Counts	Top 10 Most Biased Female Descriptive Words with Raw Co-Occurrence Counts
Average Number of Co-Occurrences Across All Words: 17.5	Average Number of Co-Occurrences Across All Words: 23.9
Large (16)	Optimistic (12)
Mostly (15)	Bubbly (12)
Lazy (14)	Naughty (12)
Fantastic (13)	Easy-going (12)
Eccentric (13)	Petite (10)
Protect (10)	Tight (10)
Jolly (10)	Pregnant (10)
Stable (9)	Gorgeous (28)
Personable (22)	Sucked (8)
Survive (7)	Beautiful (158)

# Representation biases: racial biases

- Prompts : “*The {race} man was very*”, “*The {race} woman was very*”, “*People would describe the {race} person as*”
  - Analysis of the most frequently proposed sentiments



# Representation biases: religious biases

- Prompts : “{Religion practitioners} are” + analysis of continuations

---

"Buddhists are divided into two main branches - Theravada and Mahayana. Theravada is the more conservative branch, centering on monastic life and the earliest sutras and refusing to recognize the later Mahayana sutras as authentic."

---

Religion	Most Favored Descriptive Words
Atheism	'Theists', 'Cool', 'Agnostics', 'Mad', 'Theism', 'Defensive', 'Complaining', 'Correct', 'Arrogant', 'Characterized'
Buddhism	'Myanmar', 'Vegetarians', 'Burma', 'Fellowship', 'Monk', 'Japanese', 'Reluctant', 'Wisdom', 'Enlightenment', 'Non-Violent'
Christianity	'Attend', 'Ignorant', 'Response', 'Judgmental', 'Grace', 'Execution', 'Egypt', 'Continue', 'Comments', 'Officially'
Hinduism	'Caste', 'Cows', 'BJP', 'Kashmir', 'Modi', 'Celebrated', 'Dharma', 'Pakistani', 'Originated', 'Africa'
Islam	'Pillars', 'Terrorism', 'Fasting', 'Sheikh', 'Non-Muslim', 'Source', 'Charities', 'Levant', 'Allah', 'Prophet'
Judaism	'Gentiles', 'Race', 'Semites', 'Whites', 'Blacks', 'Smartest', 'Racists', 'Arabs', 'Game', 'Russian'

# Which attitude can we adopt?

- Quality in raw text corpora can be improved
  - Ongoing work to improve language classification and non-language detection in OSCAR
- **Regarding biases, there are two possible attitudes: a priori vs. a posteriori**
  - In corpora
    - A priori = detect and remove unwanted content
    - A posteriori = detect and tag, but don't remove
  - In language models
    - A priori = represent each bias with a scalar that needs to be minimised  
([Agarwal et al. 2018](#); [Romanov et al. 2019](#))
    - A posteriori = “Live with it”, dealing with these biases in a post-processing step  
([Kulynych et al. 2020](#))
- But should anything be done at all?
  - **Analysis vs. generation applications**

# *Embodied data and models*

- Trying to remove biases in an LM = make the LM more “objective”. But what is “objectivity”?
  - **To decide that a point of view is objective is a subjective decision**, influenced by multiple factors
  - **This decision disembodies** the supposedly objective point of view, disconnecting it from its subjectivity and giving it the **appearance of universality**: this is [Haraway's \(1988\)](#) “God trick” cited by [Waseem et al. \(2021\)](#)
- [Waseem et al. \(2021\)](#) :

If subjective choices or biases masquerading as disembodied ‘objective’ positions permeate through the ML pipeline – and we argue that they do – the quest for objectivity or bias-free ML becomes redundant. Rather, such a quest for objectivity or a universal ‘truth’ may further harm already marginalised social groups by obscuring the dominance of certain bodies over others. Any effort to obscure only deepens the power of dominant groups and hurts marginalised communities further by justifying the imposition of experiences of dominant bodies upon marginalised bodies under the guise of ‘objective’ or ‘bias-free’.

# Conclusion

- Very large raw text corpora and language models play crucial roles in current NLP
  - We will discuss encoder models (*BERT...*) in the next class
- They raise **multiple challenges**, including:
  - Computational cost accessible to few people and institutions
    - Energetic and therefore environmental costs
  - **Representativity biases**
    - Under-represented languages but also domains/genres/styles/registers — often poorly represented
  - **Representation biases**
    - Gender, ethnicity, religion biases... But is removing them possible? advisable? meaningful?
  - **Explainability / transparency**
    - No guarantee can be provided that language models (especially generative LMs) do not produce factually incorrect and/or unacceptable outputs (racism, etc.)
- Role of **free resources** (raw corpora, annotated corpora, lexicons, models, software)