Representation learning and language modelling (2/2)

Benoît Sagot

MVA – Speech and Language processing – Class #2 – 22nd January 2026

Neural language models

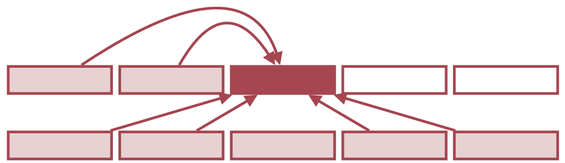
• *Draw a futuristic representation of a network of interconnected neurons without a central neuron.*
DALL-E 3 (via ChatGPT, GPT-4, 04/04/2024)

Language models

- A language model is a **probability distribution over sequences of words**
  - It quantifies the fact that the sequence *Ceci est une phrase correcte.* is more probable that the sequence *a Le llkdhfeee correcte est est*
  - This can be used to
    - **Find the most probable next word** given an input sequence
      - Example: *En raison de la grève, Pierre est arrivé en retard.* *tardant*
    - **Find the most probable word to fill a gap** in a sequence
      - Example: *En raison de la grève, Pierre est arrivé en retard.* *pluie* *ation*

Predictive language models

- **Prediction-based approach**: Given a position and a context, a model can be trained to predict which word would best fit said position
  - It will associate a weight (or a probability) to each possible word
  - Context restricted to the left context (autocompletion)
    → **generative language model**
  - Full context
    → **masked language model**
  - The language model can be **conditional**
    - Most conditional language models are generative



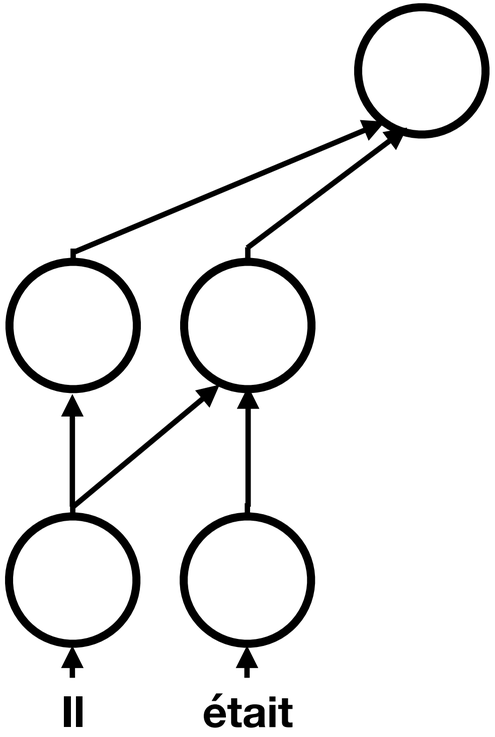
Towards neural approaches

- **Underlying rationale** (Manning 2024):
  - Language is the symbolic system *par excellence*; we should study and make use of its symbolic structure
  - This does not show that the main processor of these symbols, the human brain, is implemented as a physical symbol system
  - We need not design NLP systems as physical symbol systems
  - The brain is more like a neural network model
  - Artificial neural network models scale better and can capture the world represented by symbols
- Yet **artificial neural networks are only very distantly inspired by the human brain**
  - Interface between research on neural language models and on neuroscience = active research field

Generative neural langage models

Neural networks trained to predict the next word

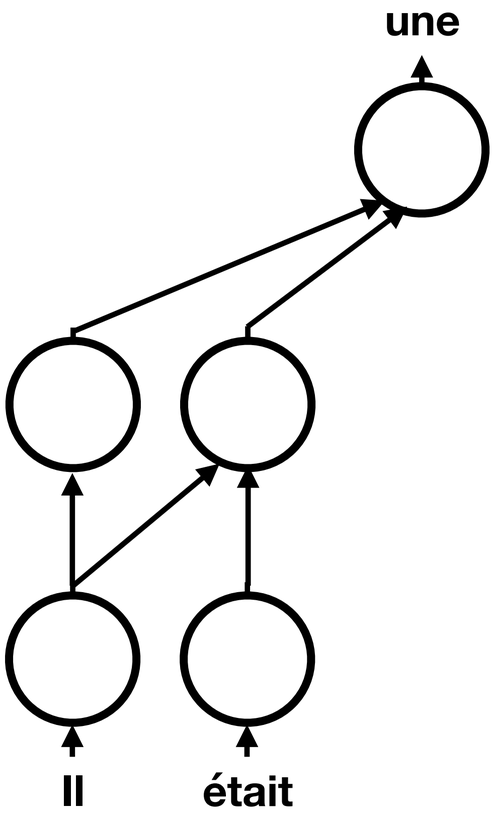
- Causal network



Generative neural langage models

Neural networks trained to predict the next word

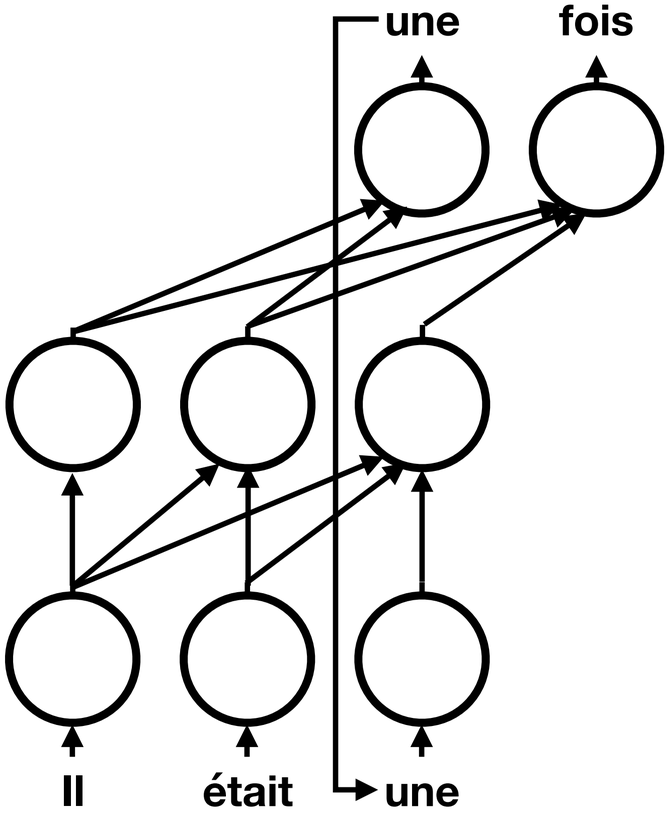
- Causal network
  - We predict the next word…



Generative neural langage models

Neural networks trained to predict the next word

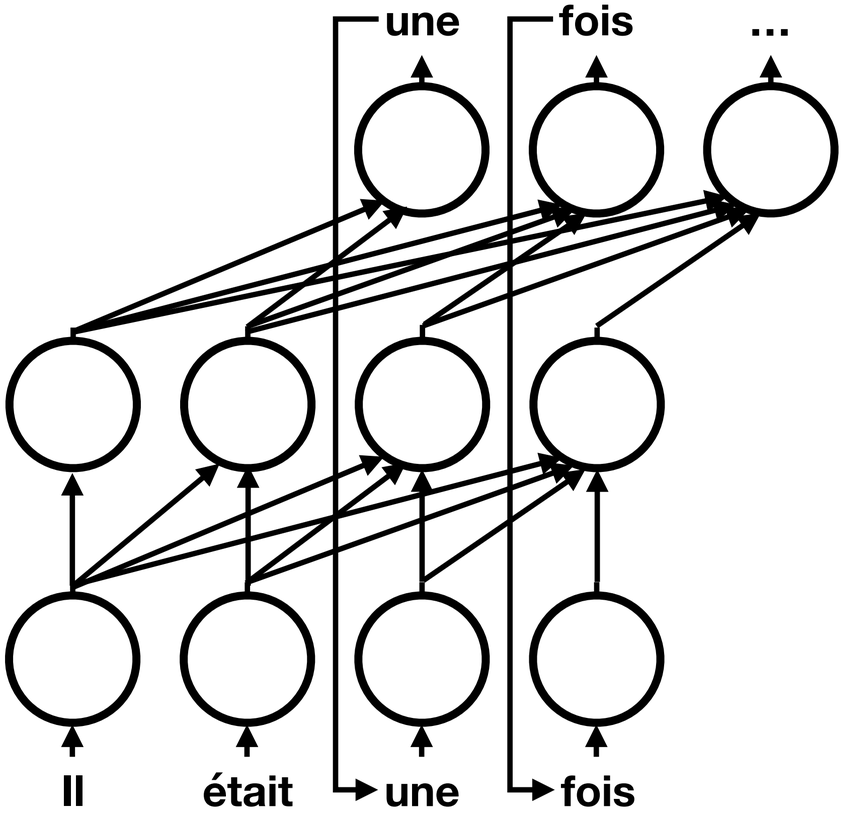
- Causal network
  - We predict the next word…
  - …then the next word…



Generative neural langage models

Neural networks trained to predict the next word

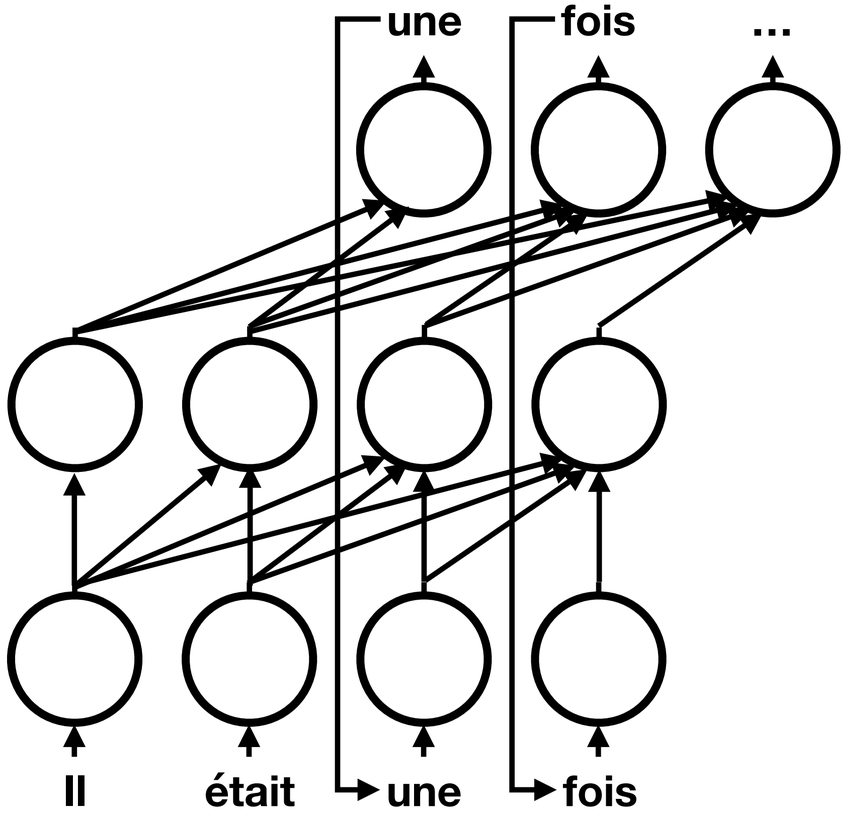
- Causal network
  - We predict the next word…
  - …then the next word…
  - …and so on



Generative neural langage models

Neural networks trained to predict the next word

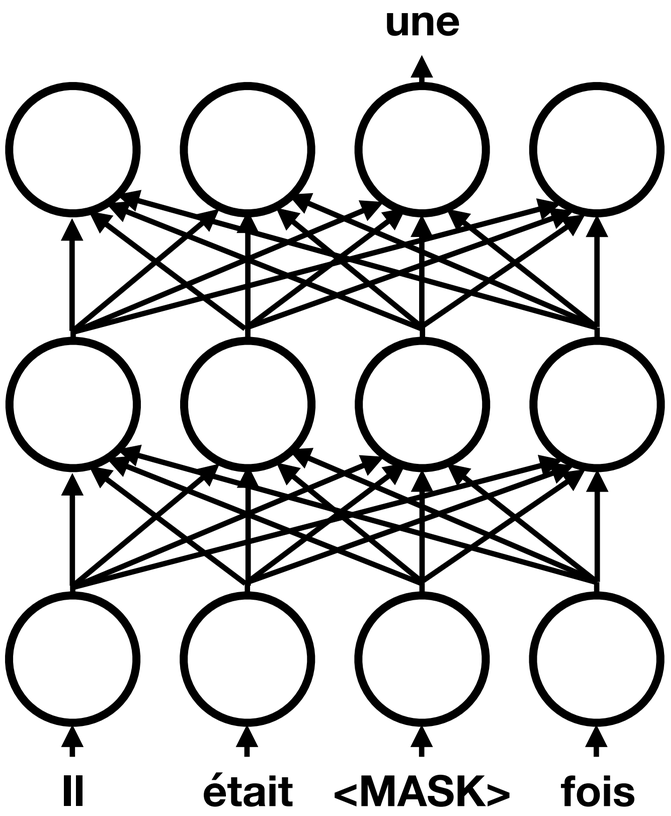
- Causal network
  - We predict the next word…
  - …then the next word…
  - …and so on
- We can predict full texts based on a ***prompt***



Neural masked language models

Neural networks trained to predict a masked word

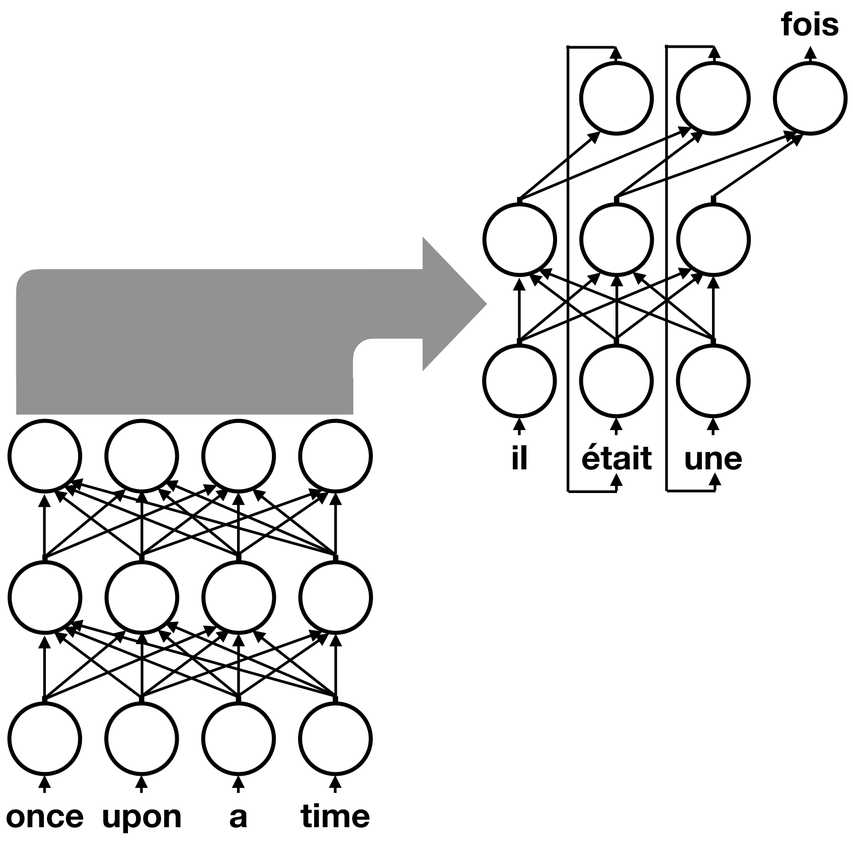
- Non causal network
  - The masked word is predicted using the full context

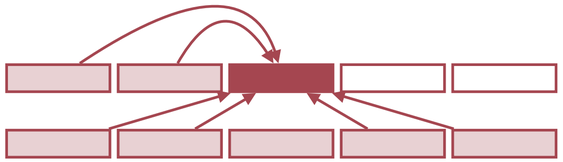


Sequence-to-sequence neural language models

Neural networks trained to conditionally predict the next word

- Encoder-decoder architecture
  - The conditioning sequence is encoded
  - We predict the next word iteratively depending on previous words and on the conditioning sequence





Generative language models

- **A generative language model** is mostly used for what it was trained for: **generating the next word given an input** (the prompt) – often iteratively
  - Some tasks can be modelled as text generation tasks via prompting
    - E.g. by giving a prompt such as “Translate the following into English: <sentence>”
  - They can also be fine-tuned

GPT, GPT-2,3,4

generative

Masked language models

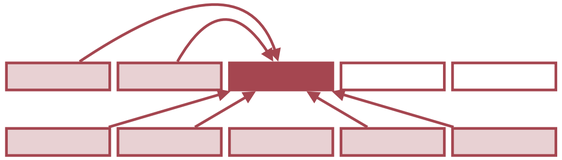
- **Language models based on a masked language modelling (MLM) architecture (e.g. BERT) produce better representations** – they see the full context, not only the left context
  - The language model can be used to do what it was trained for: filling a gap
  - The representations they produce can be used as contextual embeddings: this is the embedding provider scenario
  - A language model can also be used to initialise part of a task-specific architecture, part of which is isomorphic to the language model architecture: this is the fine-tuning scenario, i.e. an instance of transfer learning

ELMo, BERT, RoBERTa

MLM

Sequence-to-sequence language models

- **A sequence-to-sequence language model is a generative language model conditioned on a given input**
  - The input can be a “altered” version of the expected output (e.g. masked, transformed, written in another language, and more)
  - Such models are generally fine-tuned to perform conditioned text generation tasks, including but not limited to text transformation tasks. Examples:
    - Text transformation: machine translation, text simplification, etc.
    - Other conditioned text generation tasks: question answering



BART, T5

seq2seq

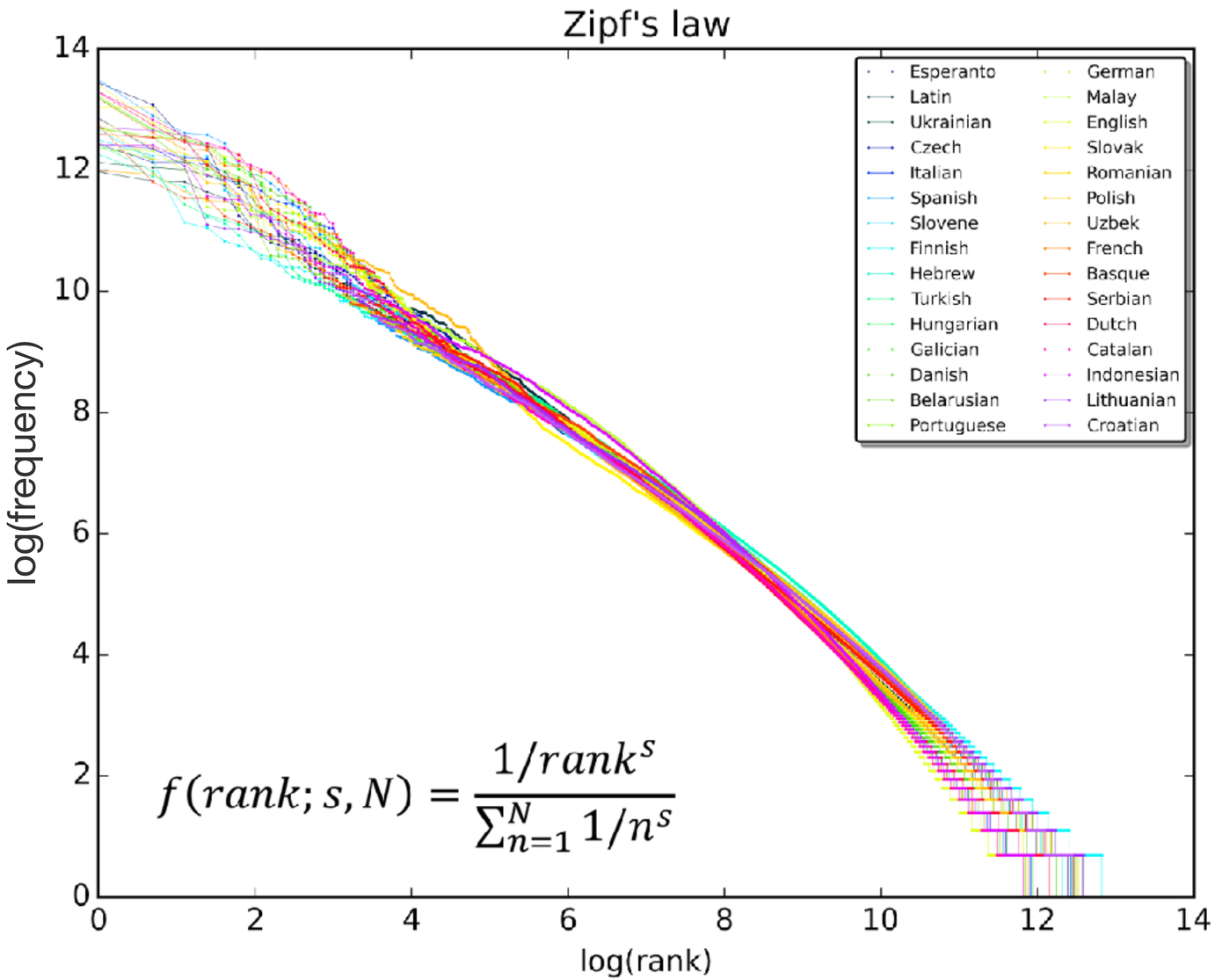
Word vectors

• *Draw me Natural Language Processing, steampunk style*
• *I would like something that would give a bigger importance to the notion of textual data.*
DALL-E 3 (via ChatGPT, GPT-4, 25/11/2023)

Lexical sparsity

A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias

(source: Wikipedia; data: dumps from Oct 2015)



What word vectors do we need?

- Question answering:
  - Question: *How **tall** is Mt. Everest?*
    Candidate answer: *The official **height** of Mount Everest is 8848m*
- Plagiarism detection:
  - *Mainframes **are primarily** referred to as **large** computers with **rapid**, advanced processing capabilities that **can execute and** perform tasks **equivalent to many** Personal Computers (PCs)*
  - *Mainframes **usually are** referred to as large computers with **fast**, advanced processing capabilities that **could** execute and perform **by itself** tasks **that may require a lot of** Personal Computers (PCs)*
- We need to be able to capture **word similarity** in our representations
  - Thanks to its context of occurrences in a corpus
  - … but ideally, with different representations for different occurrences, to fully accommodate the fact that a same word (token) can have multiple meanings depending on the context

Representing words

- Each word, in each context it appears in, conveys a number of features
  - E.g. morphological, syntactic, semantic, etc.
- These features can be represented in 2 main ways:
  - Explicitly:
    cat = [ANIMAL: +, HUMAN: –, SIZE: average…]
  - Cf. lexicons/dictionaries, including lexical resources widely used in NLP
- Implicitly, using vector representations:
  - Such approaches were notably put forward in early neural approaches to NLP (Rumelhart, Hinton & Williams 1986)
  - Other works have proposed representing words (meanings) as vectors (Victorri & Fuchs 1992)

Photo BS

1-hot word vector representations

- Simplest vector representation:
  - Choose a vocabulary of size *n*
  - Represent the i-th word of the vocabulary as a vector of zeros except for the i-th coordinate, which is set to 1
  - **"1-hot" representation**
  - Main drawback: two words with similar features and behaviours have nothing more in common than completely different words
    - The scalar product is 0 in both cases

1	a
2	à
3	abaca
4	abacas
5	abacule
6	abacules
7	abaissa
8	abaissable
9	abaissables
10	abaissai
11	abaissaient
12	abaissais
…	
29 000	zythum
30 000	zythums

	1	2	3	4	5	6	7	8	…	29 000	30 000
abaissa	0	0	0	0	0	0	1	0	…	0	0

1-hot word vector representations

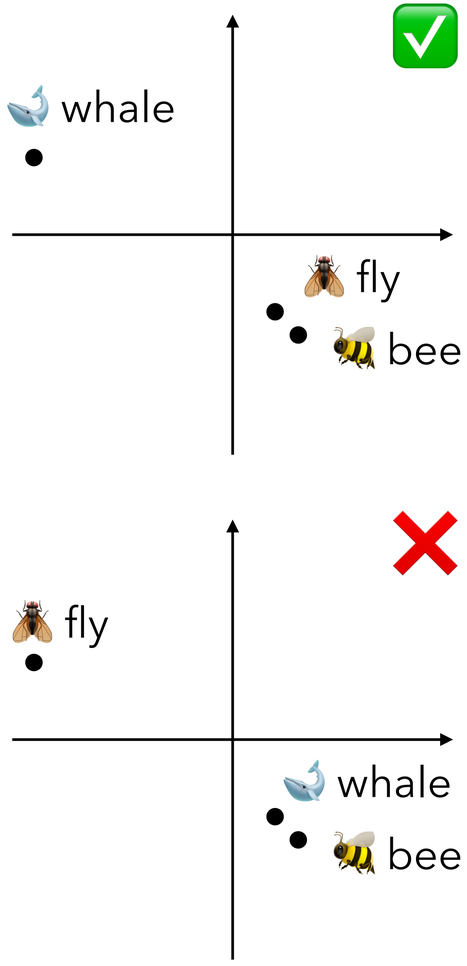
- Simplest vector representation:
  - Choose a vocabulary of size *n*
  - Represent the i-th word of the vocabulary as a vector of zeros except for the i-th coordinate, which is set to 1
  - **"1-hot" representation**
  - Main drawback: two words with similar features and behaviours have nothing more in common than completely different words
    - The scalar product is 0 in both cases

1	a
2	à
3	abaca
4	abacas
5	abacule
6	abacules
7	**abaissa**
8	abaissable
9	abaissables
10	abaissai
11	abaissaient
12	abaissais
…	
29 000	zythum
30 000	zythums

	1	2	3	4	5	6	**7**	8	…	29 000	30 000
abaissa	0	0	0	0	0	0	**1**	0	…	0	0

1-hot word vector representations

- Simplest vector representation:
  - Choose a vocabulary of size *n*
  - Represent the i-th word of the vocabulary as a vector of zeros except for the i-th coordinate, which is set to 1
  - **"1-hot" representation**
  - Main drawback: two words with similar features and behaviours have nothing more in common than completely different words
    - The scalar product is 0 in both cases
- If **two similar words could have similar representations**, models using such representations could be able to generalise



The distributional hypothesis

- Exemple – modified by Lin (1998), modified version of (Nida, 1975, p. 167):
  What is tesgüino?
  A bottle of tesgüino is on the table.
  Everybody likes tesgüino.
  Tesgüino makes you drunk.
  We make tesgüino out of corn.
  - From context words humans can guess tesgüino means an alcoholic beverage akin to beer
- Distributional hypothesis: **Two words are similar if they have similar word contexts**
  - Harris (1954):
    - *oculist and eye-doctor … occur in almost the same environments*
    - *If A and B have almost identical environments we say that they are synonyms.*
  - Firth (1957): *You shall know a word by the company it keeps!*

Illustration extraite d'un article du Sonora Star

Word vector representations

- **Sparse word vector representations**
  - Mutual-information weighted word cooccurrence matrices
- **Dense word vector representations, a.k.a non-contextual word embeddings**
  - Singular value decomposition (and Latent Semantic Analysis)
  - Neural-network-based representations (skip-grams, CBOW)
- **Dense word occurrence vector representations, a.k.a. contextual word embeddings**
  - Neural language model-based representations
- ***Shared intuition***
  - The meaning of a word is modelled by "embedding" it in a vector space
  - The meaning can be **lexical** (averaged over all occurrences, as in a dictionary) or **contextual**

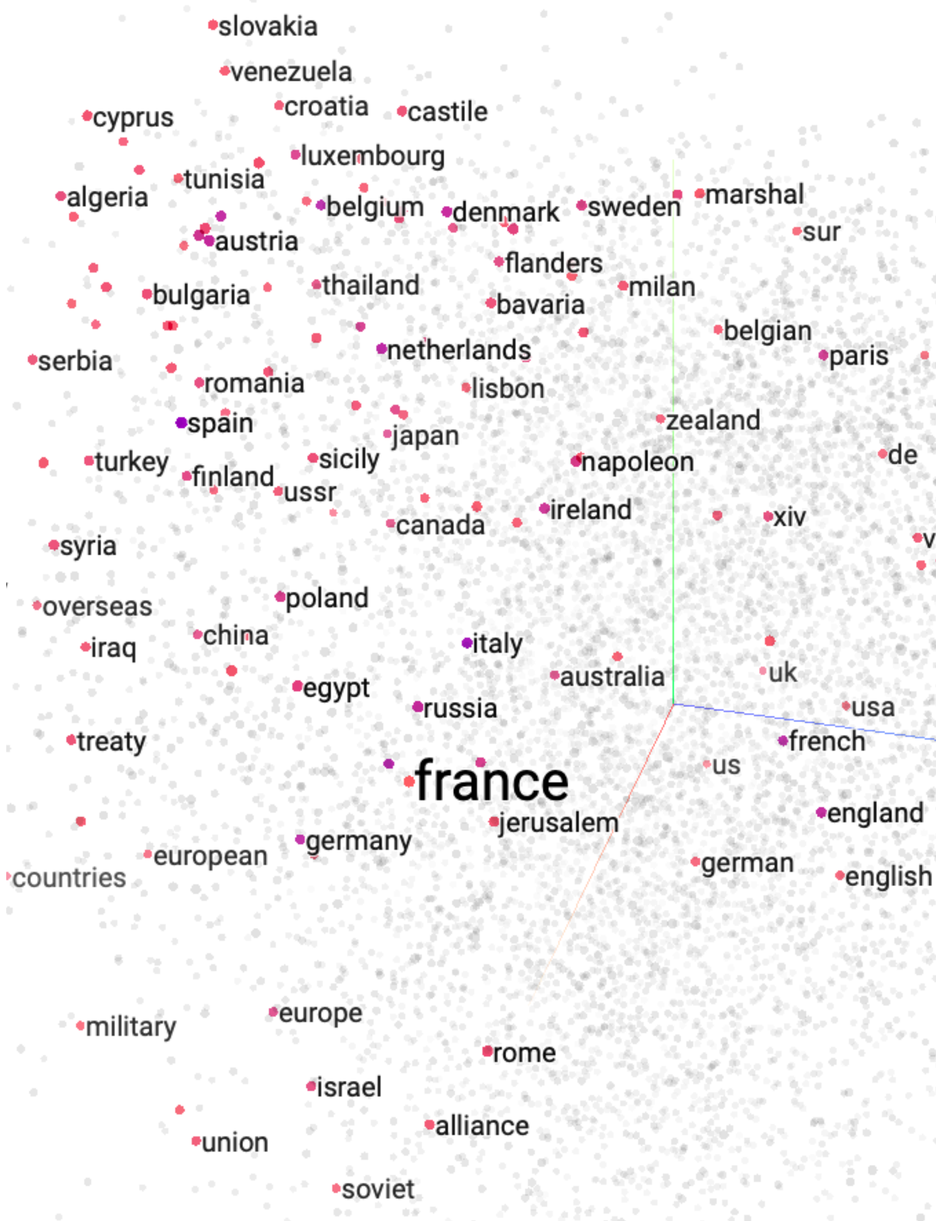


Illustration generated using the Embedding Projector

Cooccurrence matrices

- Cooccurrence = appearing in the same environment
  - document
  - immediate context
- Cooccurrence matrix = frequency counts of *(word, environment)* pairs
- Two main categories of cooccurrence matrix
  - Term-document matrix: how often word *i* occurs in document *j*
  - Term-term (word-word, word-context) matrix: how often word *i* occurs in the immediate vicinity of word *j*

Term-document matrix

- Each cell: count of word $w \in V$ in a document $d \in D$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

- Each cell: count of word $w \in V$ in a document $d \in D$
  - Each document is a count vector in $\mathbb{N}^{|V|}$ (a column)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

- Each cell: count of word $w \in V$ in a document $d \in D$
  - Each document is a count vector in $\mathbb{N}^{|V|}$ (a column)
  - Each word is a count vector in $\mathbb{N}^{|D|}$ (a row)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in term-document matrices

- Two documents are similar if their vectors are similar (Salton *et al.* 1975)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in term-document matrices

- Two documents are similar if their vectors are similar (Salton *et al.* 1975)
- Two words are similar if their vectors are similar (Deerwester *et al.* 1990)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in word-word matrices

- Instead of entire documents, use **smaller contexts** (paragraph, window of ± 4 words)
- A word is now defined by a vector over **counts of context words**
- Instead of each vector being of length $|D|$
  - each vector is now of length $|V|$, the word-word matrix is $|V|^2$
- Example with a 7-word context

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…	…
apricot	0	0	0	1	0	1		
pineapple	0	0	0	1	0	1		
digital	0	2	1	0	1	0		
information	0	1	6	0	4	0		

Word-word matrix

- We showed only 4x6, but the real matrix is 500,000 x 500,000
  - Very sparse (most values are 0)
  - That is acceptable, since there are lots of efficient algorithms for sparse matrices.
- Similarity is measured using the **cosine distance** between two vectors
- The **size of context windows** depends on your goals
  - The **shorter** the windows, the more **syntactic** the representation
    1-3 ~ syntactic similarity
  - The **longer** the windows, the more **semantic** the representation
    4-10 ~ semantic/topical similarity

Problems with raw counts

- Raw word frequency is not a great measure of association between words
  - It is very skewed
    - For example "the" and "of" are very frequent, but maybe not the most discriminative
- We would be more interested in a measure that would give more importance to context words that are **more informative** about the target word
  - Pointwise Mutual Information (PMI): do words *x* and *y* cooccur more than if they were independent? (Church and Hanks 1990)

$$\mathrm{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

(ranges from -∞ to +∞)

  - But this metric has issues

Positive Pointwise Mutual information (PPMI)

- Negative PMI values are problematic
  - They correspond to words co-occurring **less** than we expect by chance
  - Unreliable without a really huge corpus
    - Imagine two rare words w_1 and w_2 (say, frequency = 10^{-6}): it is hard to compare $P(w_1,w_2)$ with 10^{-12} and know whether the difference is statistically significative…
  - It is unclear whether we have intuitions about unrelatedness
    - We are interested in similarity (relatedness)
- We replace negative PMI values by 0 => PPMI (Bullinaria and Levy 2007)

$$\mathrm{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

Sparse vs. dense vectors

- PPMI vectors are **sparse**
  - Up to 500,000 dimensions, if not more (depends on the language)
  - Most values are 0
  - Cosine distance not optimal on such vectors, no information is shared between similar words
  - Machine learning systems using such vectors have a large number of weights to train
- How can we get **dense**, low-dimensionality vectors?
  - Many techniques have been proposed. 2 examples:
    - Classical: **singular value decomposition (SVD)** of (P)PMI matrices (Deerwester *et al.* 1990)
      - State-of-the-art until the 2010s
    - Since the mid-2010s: side-effect of predictive **neural networks**

Does it work better than sparse vectors?

- In short, yes!
  - Lower dimensions represent information that is not important, not necessarily significant: **denoising**
  - Removing lower dimensions results in **generalisations**, including capturing **higher-order cooccurrence**
  - Fewer dimensions = models easier to learn (**fewer weights**)

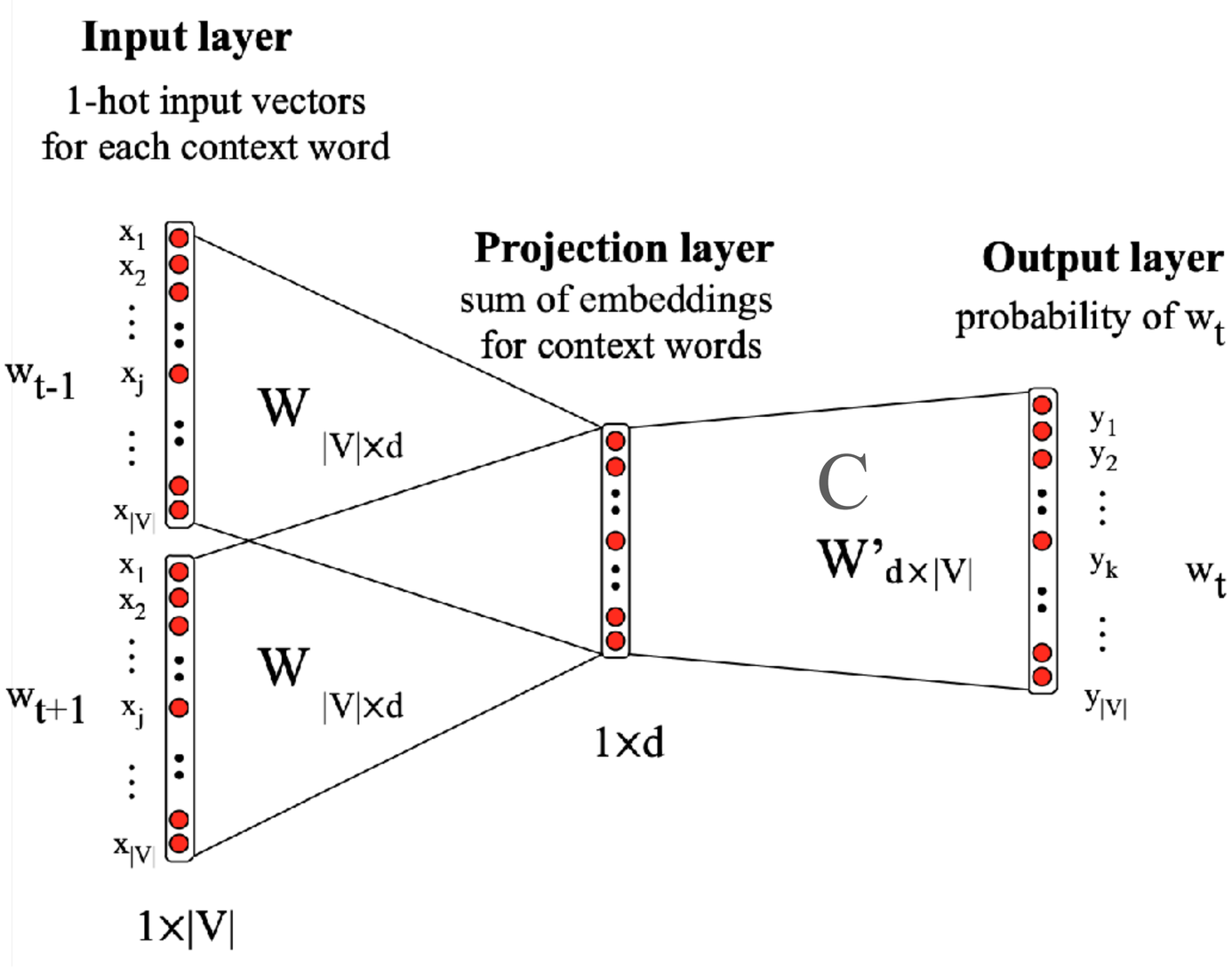
Prediction-based embeddings

- Different approach for implementing the same intuition
  - Underlying principle is still the distributional hypothesis
  - Instead of starting with **word counts to quantify the notion of distribution**, we **learn to predict words based on distributional properties of their contexts**
  - We do so **using a neural approach**
- Underlying idea: we train a neural network to perform a given word-based task and extract **intermediate representations** as word representations (word embeddings)
- Approach popularised by the **word2vec** package (Mikolov et al. 2013a,b)
  - Easy and fast to train
  - Freely available, pre-trained embeddings

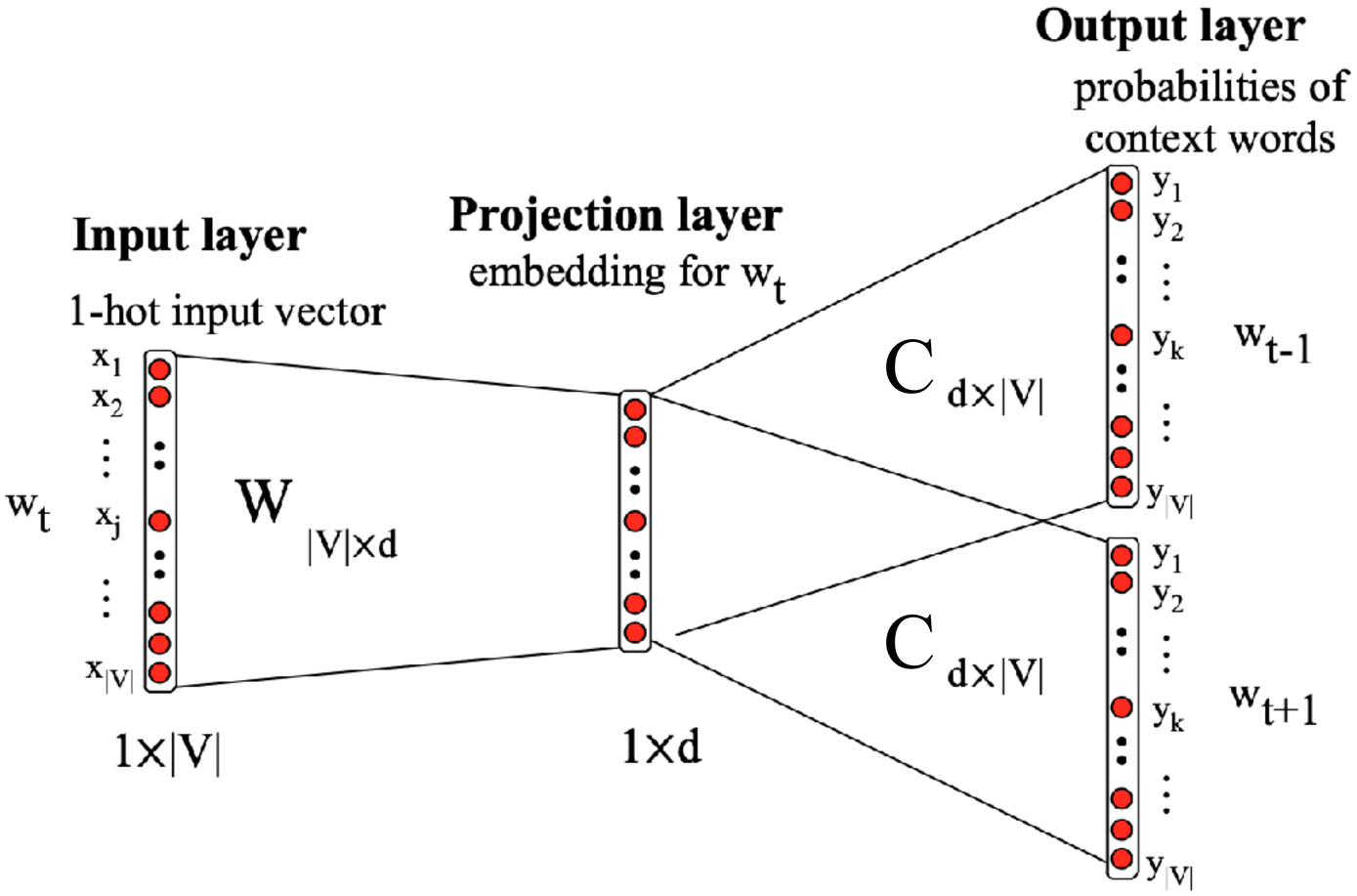
word2vec

- Neural network with one hidden layer > This hidden layer will provide the embeddings
- Input and output layers use **1-hot vector representations**
  - A word $w_i \in V$ is represented by a $|V|$-dimensional 1-hot vector
  - A k word context is represented by a $|V|$-dimensional k-hot vector
  - In the output layer, each value is interpreted as a probability via a SoftMax operation
- Two types of prediction in word2vec:
  - Given a context surrounding a position, predict the word that should fill this position: **CBoW** (continuous bag of words)
  - Given a word, predict its neighbours: **skip-gram**

CBoW

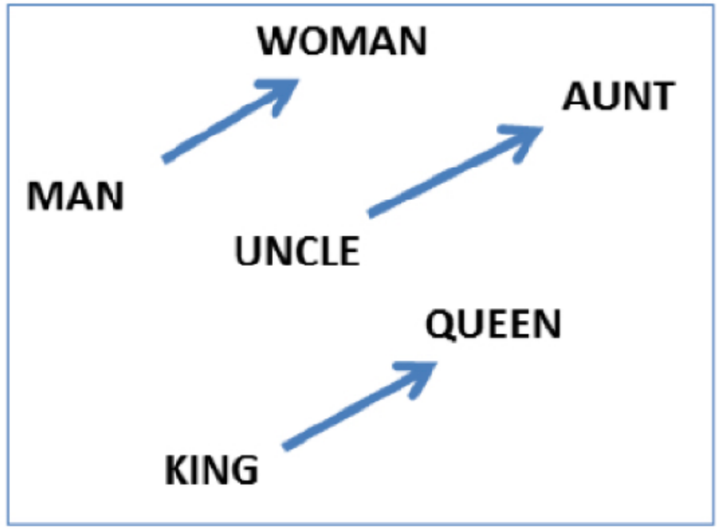

Input layer
1-hot input vectors
for each context word
Projection layer
sum of embeddings
for context words
Output layer
probability of w_t
w_{t-1}
w_{t+1}
w_t
x_1 x_2 x_j $x_{|V|}$
y_1 y_2 y_k $y_{|V|}$
W $|V|\times d$
W $|V|\times d$
C
W' $d\times|V|$
$1\times d$
$1\times|V|$

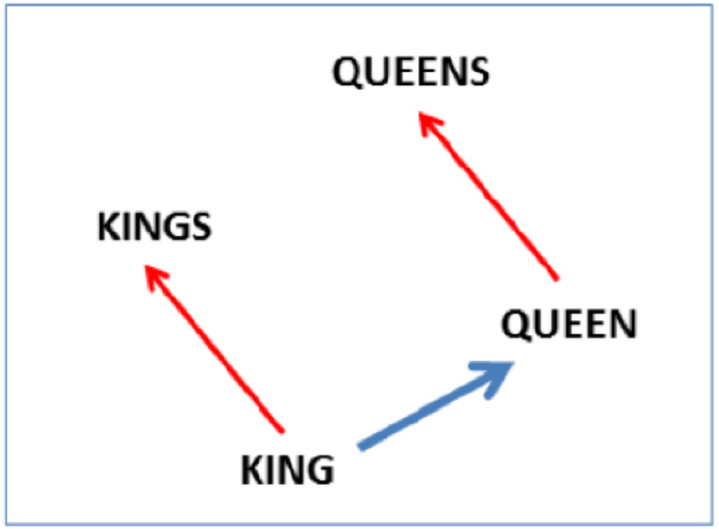
Skip-grams


Output layer
probabilities of
context words
Input layer
1-hot input vector
Projection layer
embedding for w_t
w_t
x_1
x_2
x_j
$x_{|V|}$
$W_{|V|\times d}$
$1\times|V|$
$1\times d$
$C_{d\times|V|}$
$C_{d\times|V|}$
y_1
y_2
y_k
$y_{|V|}$
w_{t-1}
w_{t+1}

Analogy: the structure of word similarity

- vector(*'king'*) – vector(*'man'*) + vector(*'woman'*) ≈ vector(*'queen'*) – if vector('king') is left aside
- vector(*'Paris'*) – vector(*'France'*) + vector(*'Italy'*) ≈ vector(*'Rome'*)





Word2vec vs. PPMI+SVD

- The full skip-gram algorithm in word2vec includes several clever refinements, in particular **negative sampling** – hence the acronym **SGNS**
- It was shown by (Levy & Goldberg 2014) that the core of the algorithm, when embedding dimensionality increases, tends towards factorising the word-word PPMI matrix
- Yet, word2vec vectors are better than the previous state-of-the-art. Why?
    - In fact, the clever refinements make the difference – they can be adapted to the SVD approach, with very good results! (Levy *et al.* 2015)

Overall, there does not seem to be a consistent significant advantage to one approach over the other, thus refuting the claim that prediction-based methods are superior to count-based approaches.

The contradictory results in (Baroni et al., 2014) stem from creating `word2vec` embeddings with somewhat pre-tuned hyperparameters (recommended by `word2vec`), and comparing them to "vanilla" PPMI and SVD representations. In particular, shifted PMI (negative sampling) and context distribution smoothing (`cds` = 0.75, equation (3) in Section 3.2) were turned on for SGNS, but not for PPMI and SVD. An additional difference is Baroni et al.'s setting of `eig=1`, which significantly deteriorates SVD's performance (see Section 6.1).

Quote from (Levy *et al.* 2015)

Non-contextual embeddings

- These approaches have made it possible to successfully use neural networks to perform NLP tasks.
- Their main limitation is that they are **non-contextual**
  - The French word ‘avocat’ (meaning ‘avocado’ or ‘lawyer’ depending on the context) is represented by the same vector regardless of its context

Pierre est un excellent avocat

Pierre mange un excellent avocat

Portrait of Frédéric Thomas (avocat, writer, journalist). Musée Carnavalet

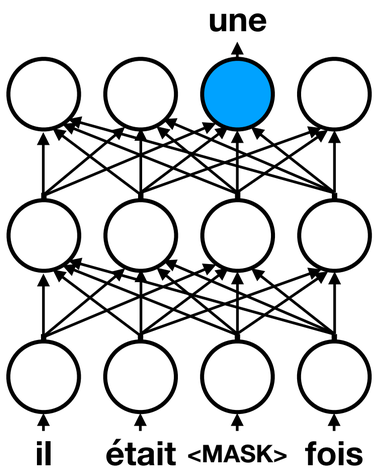
Source: Wiktionary

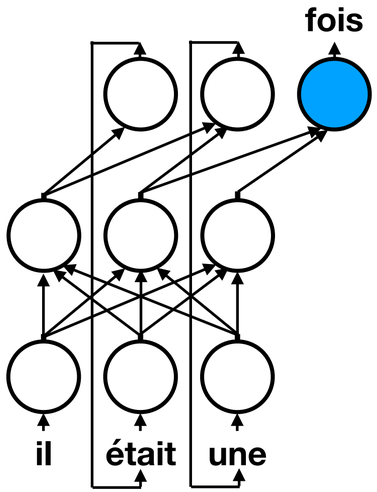
- *Draw a hybrid of a lawyer and an avocado*
- *Add a very light steampunk touch to it, while keeping it ultrarealistic.*

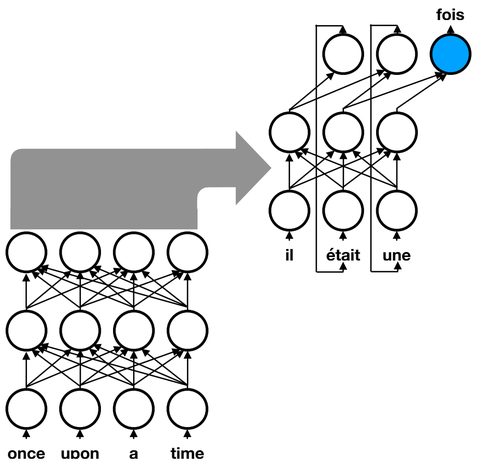
DALL-E 3 (via ChatGPT, GPT-4, 04/02/2024)

Language models and word embeddings

- How can we produce word embeddings (vectors representing words) that depend on the context?
- We need a system that represents words as vectors
  - And that uses close vectors for similar (word, context) pairs
- This is exactly what a **neural language model** does
  - A neural language model is a predictive language model that generates a probability distribution over the predefined vocabulary
  - The vector representing the word predicted by the model can be seen as a **contextual word embedding**







Recurrent language models

• *Draw me Natural Language Processing, steampunk style*
• *I would like something that would give a bigger importance to the notion of textual data.*
DALL-E 3 (via ChatGPT, GPT-4, 25/11/2023)

RNN

generative

Recurrent architectures for generative LMs

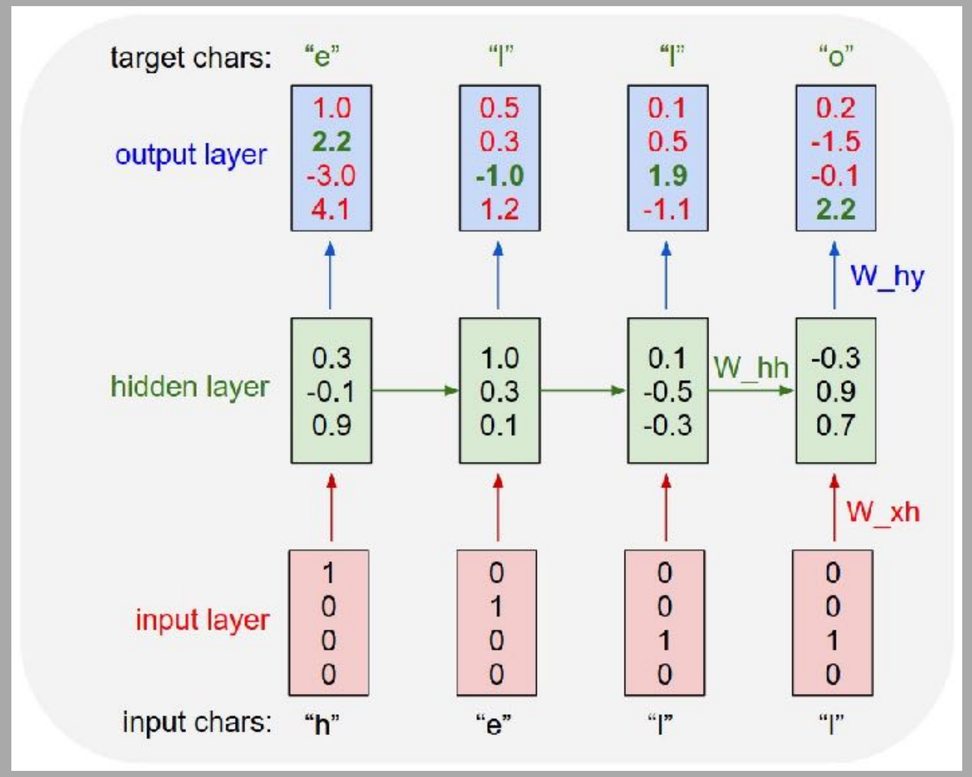
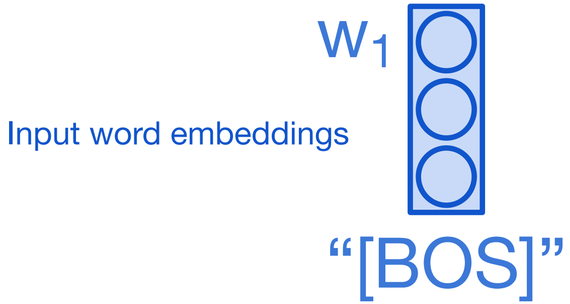


Figure from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Networks

RNN

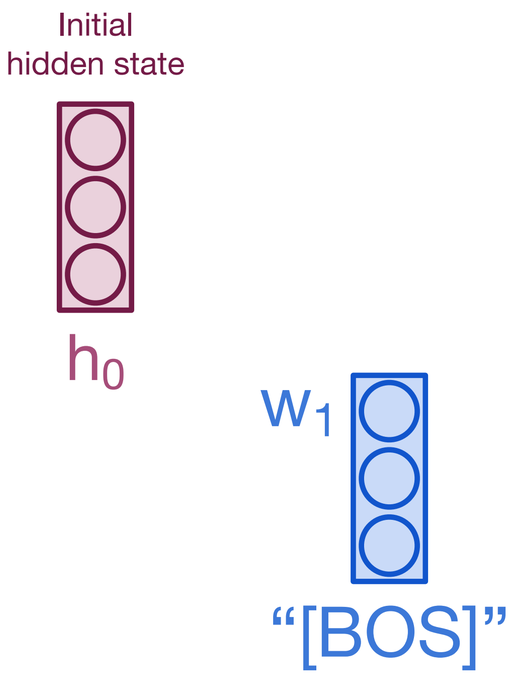
generative



Recurrent Neural Networks

RNN

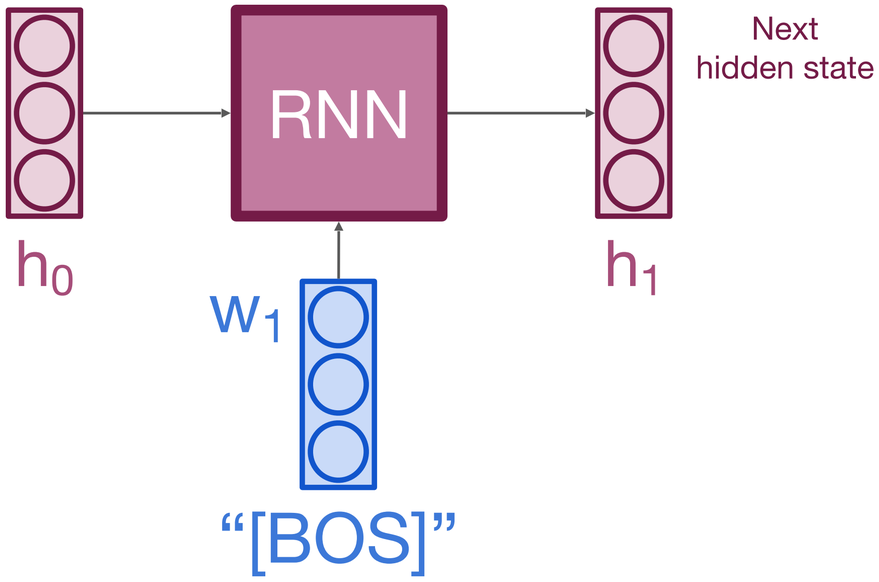
generative



Recurrent Neural Networks

RNN

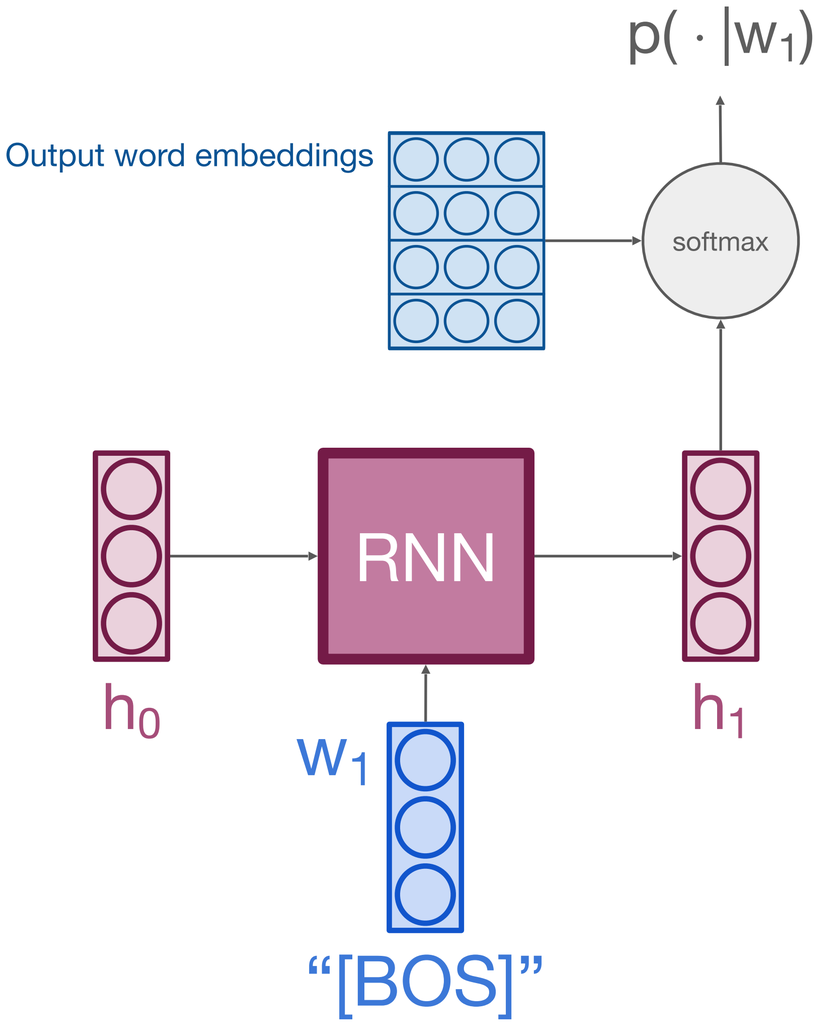
generative



Recurrent Neural Networks

RNN

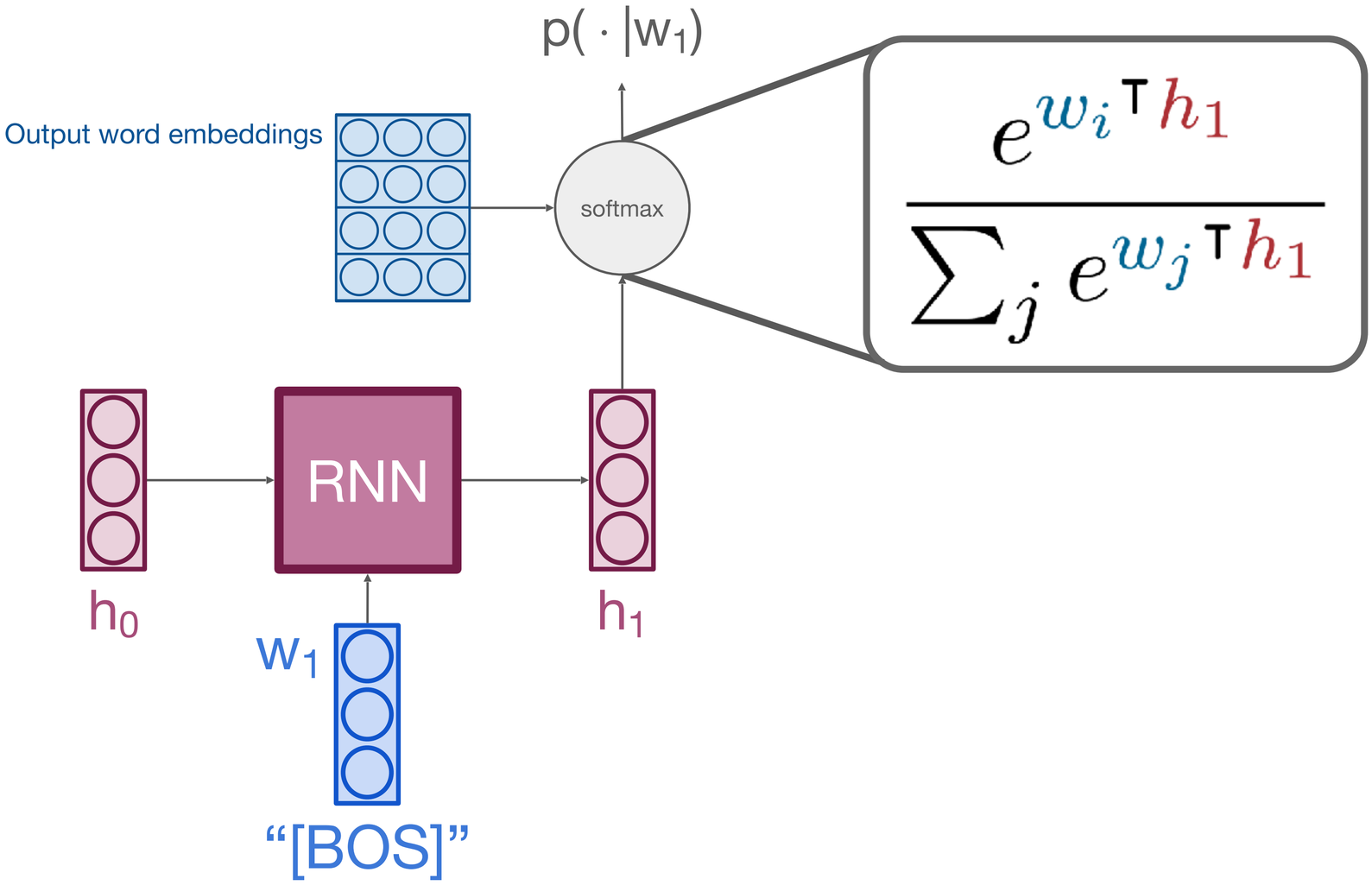
generative



Recurrent Neural Networks

RNN

generative

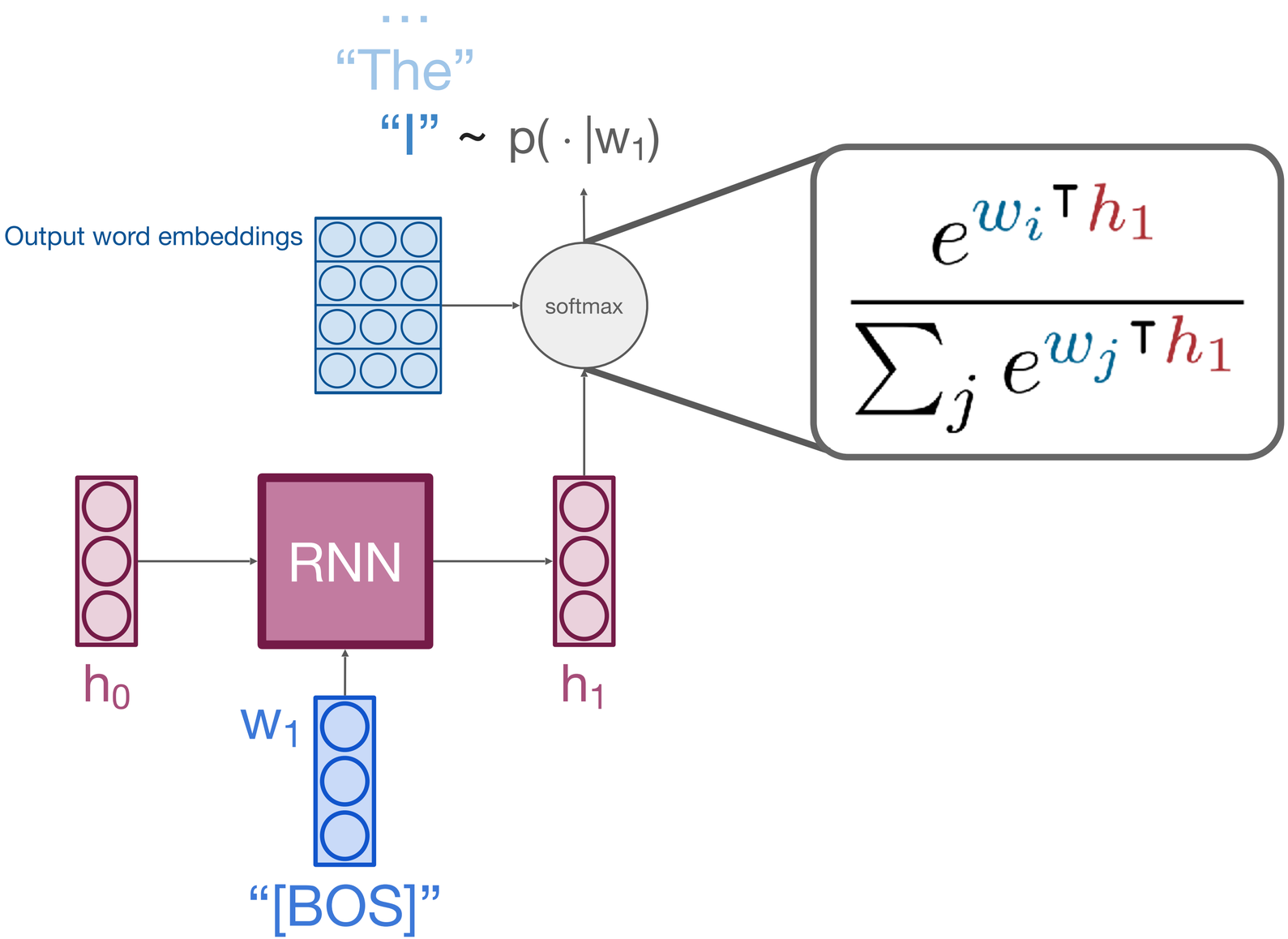


$$\frac{e^{w_i{}^\top h_1}}{\sum_j e^{w_j{}^\top h_1}}$$

Recurrent Neural Networks

RNN

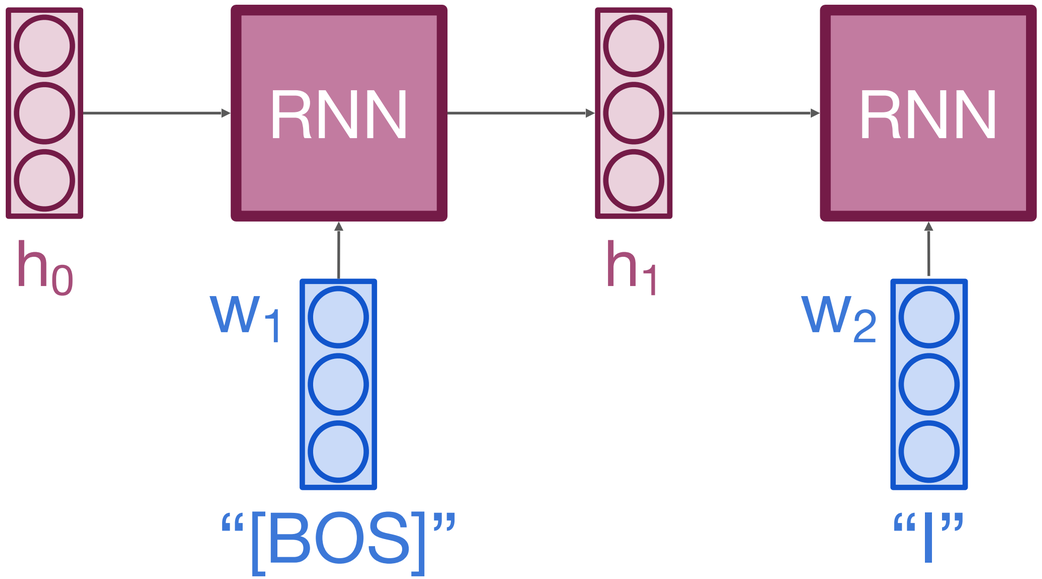
generative



Recurrent Neural Networks

RNN

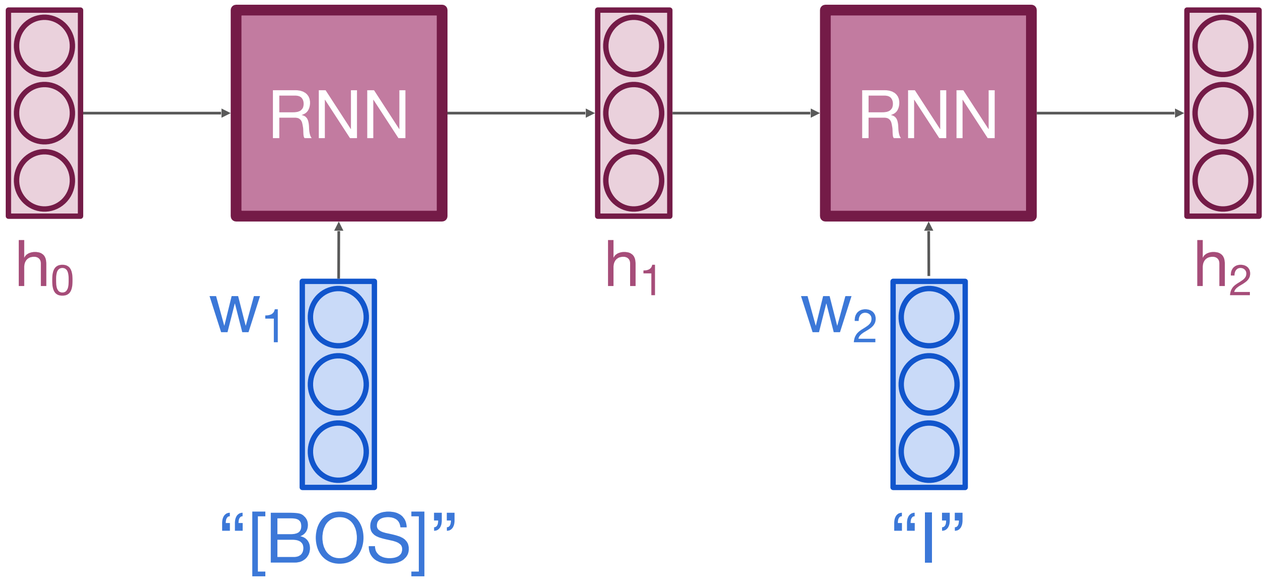
generative



Recurrent Neural Networks

RNN

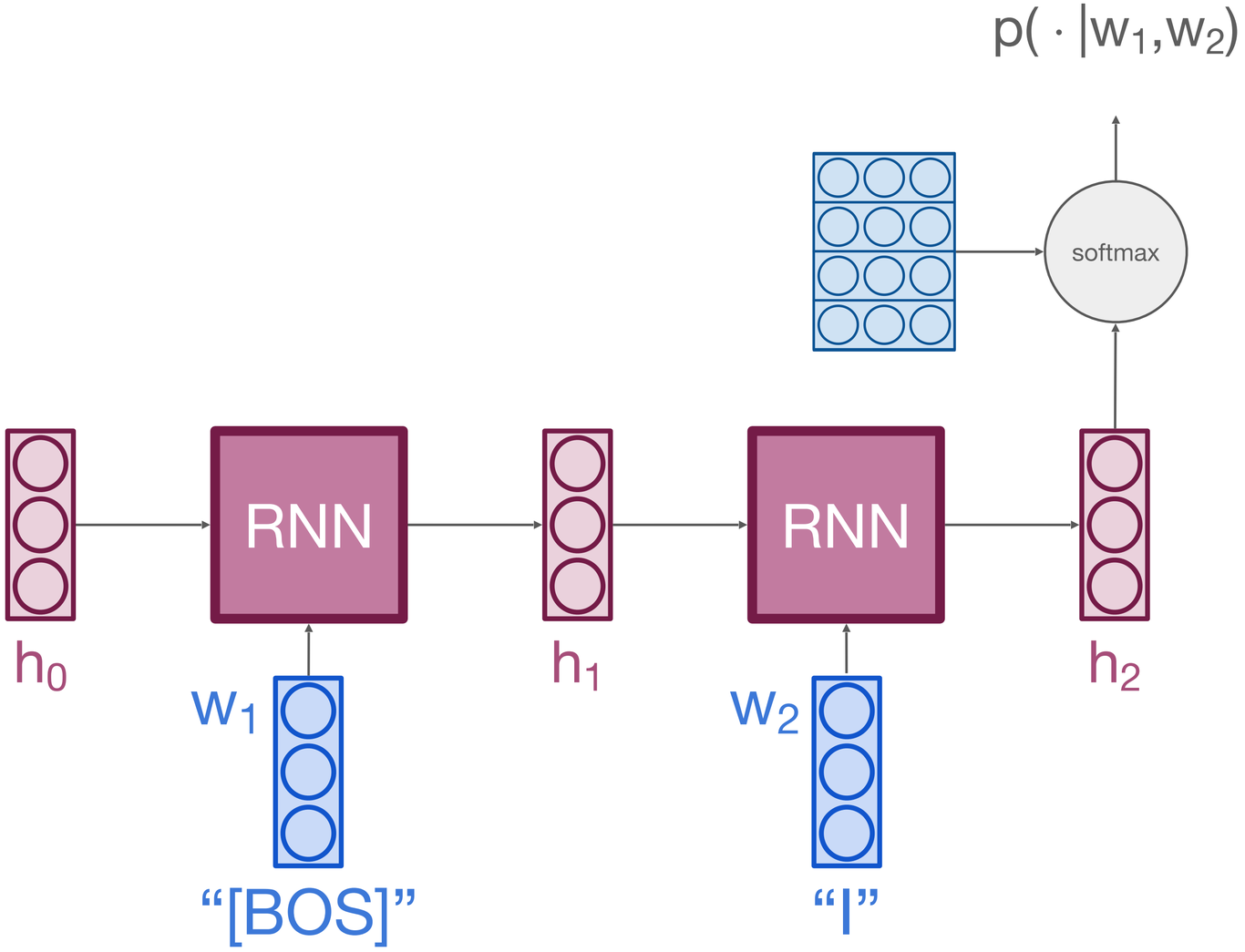
generative



Recurrent Neural Networks

RNN

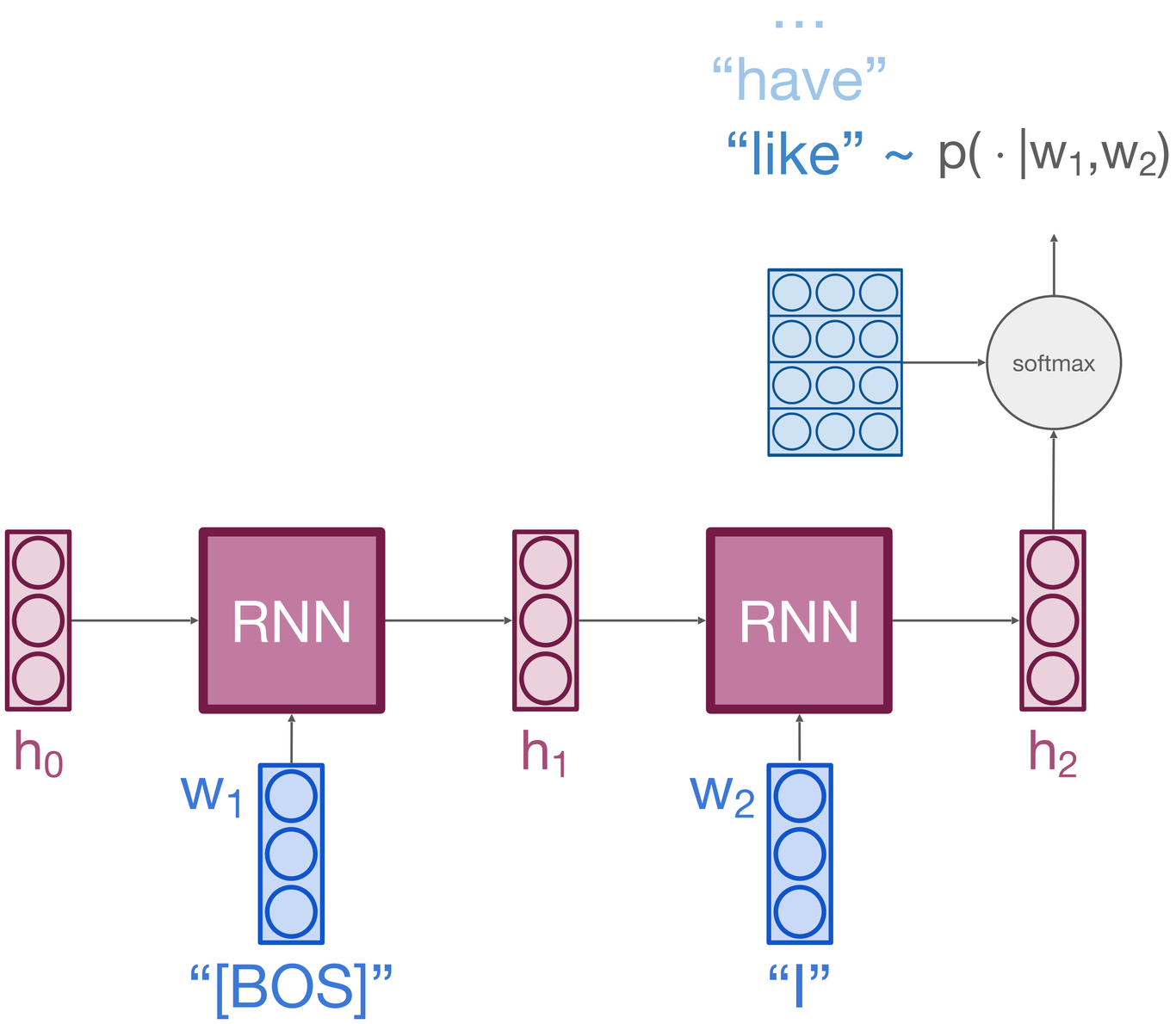
generative



Recurrent Neural Networks

RNN

generative



Recurrent Neural Networks

RNN

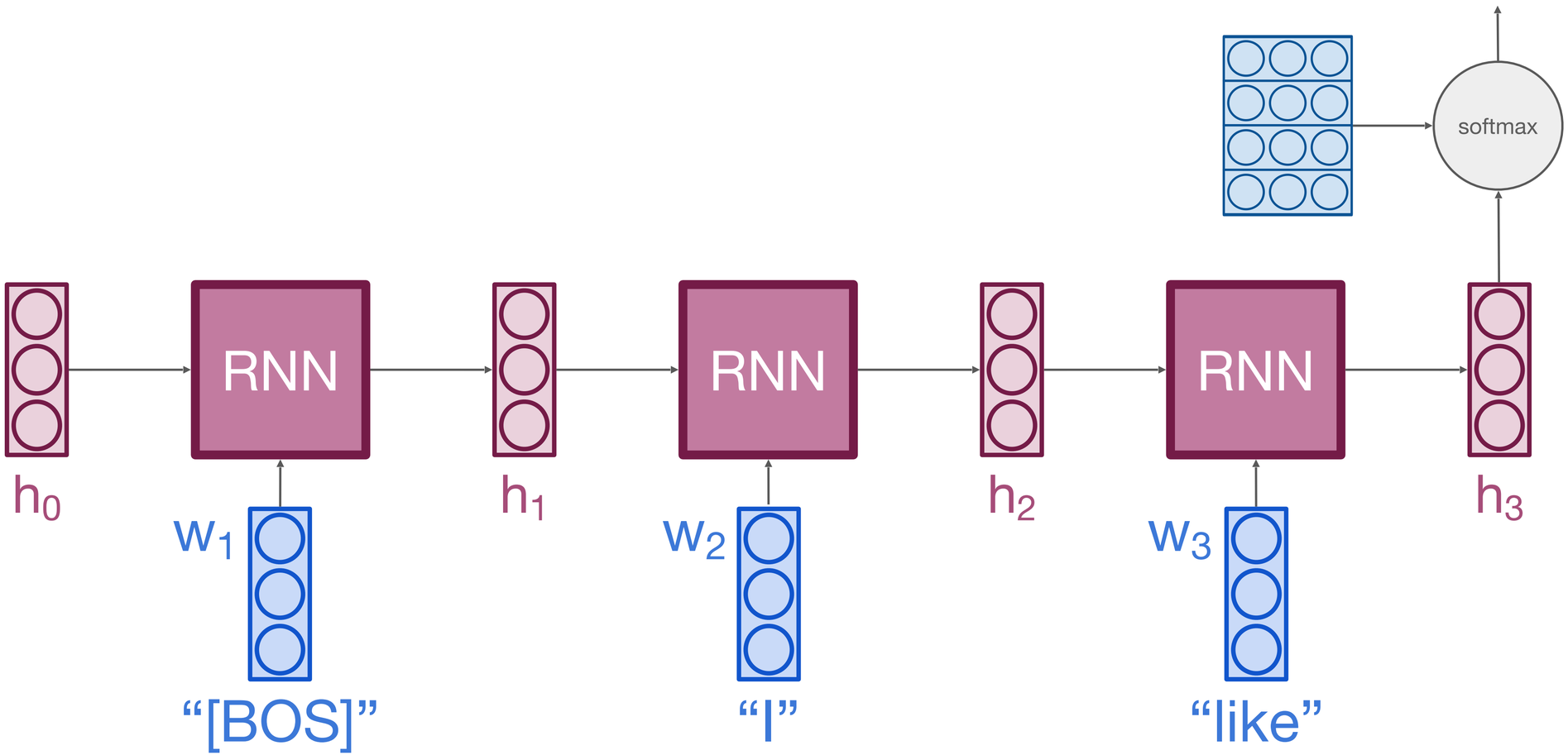
generative

…

"running"

"dogs"

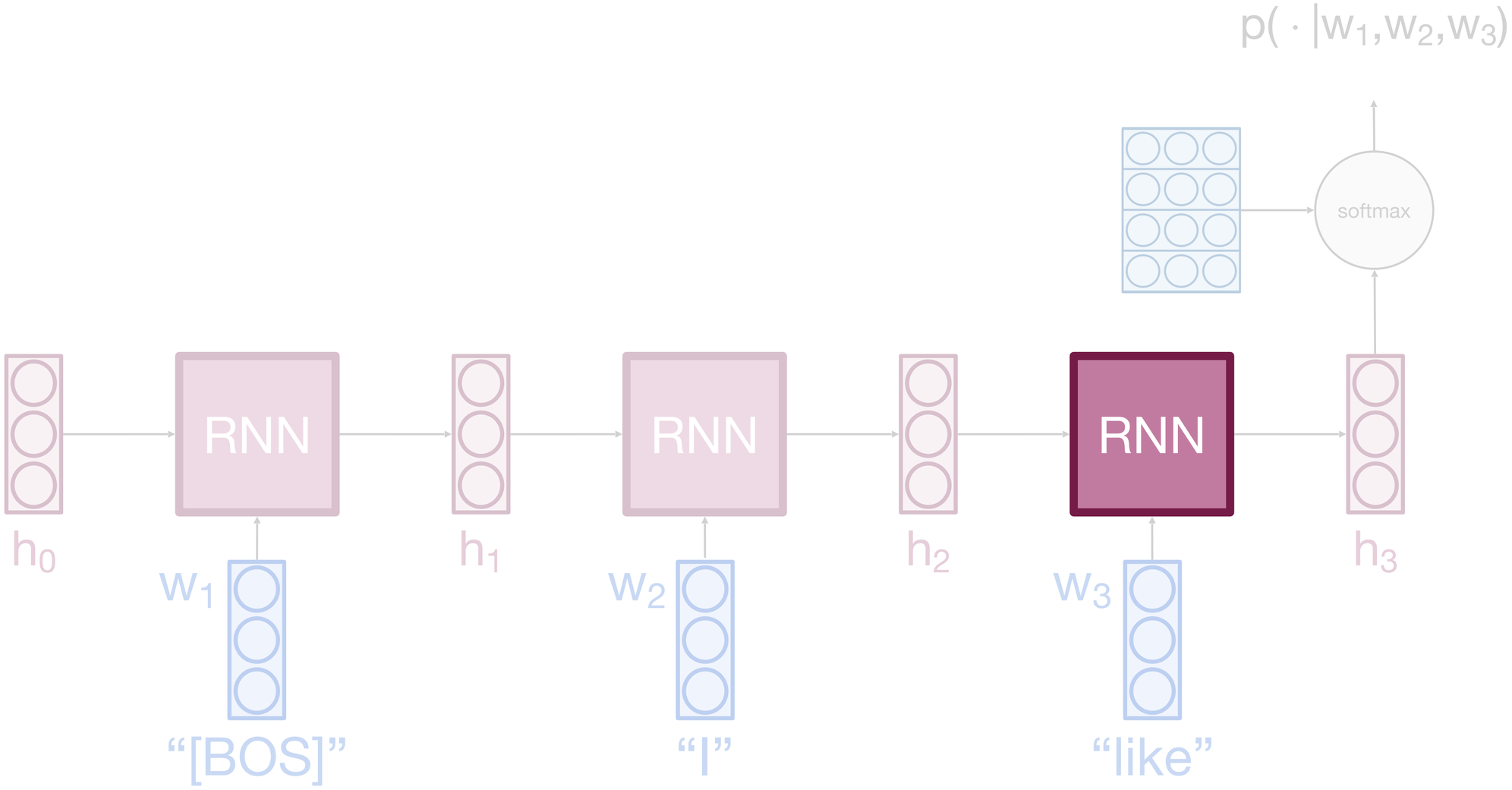
"doughnuts" ~ $p(\cdot \mid w_1, w_2, w_3)$



Recurrent Neural Networks

RNN

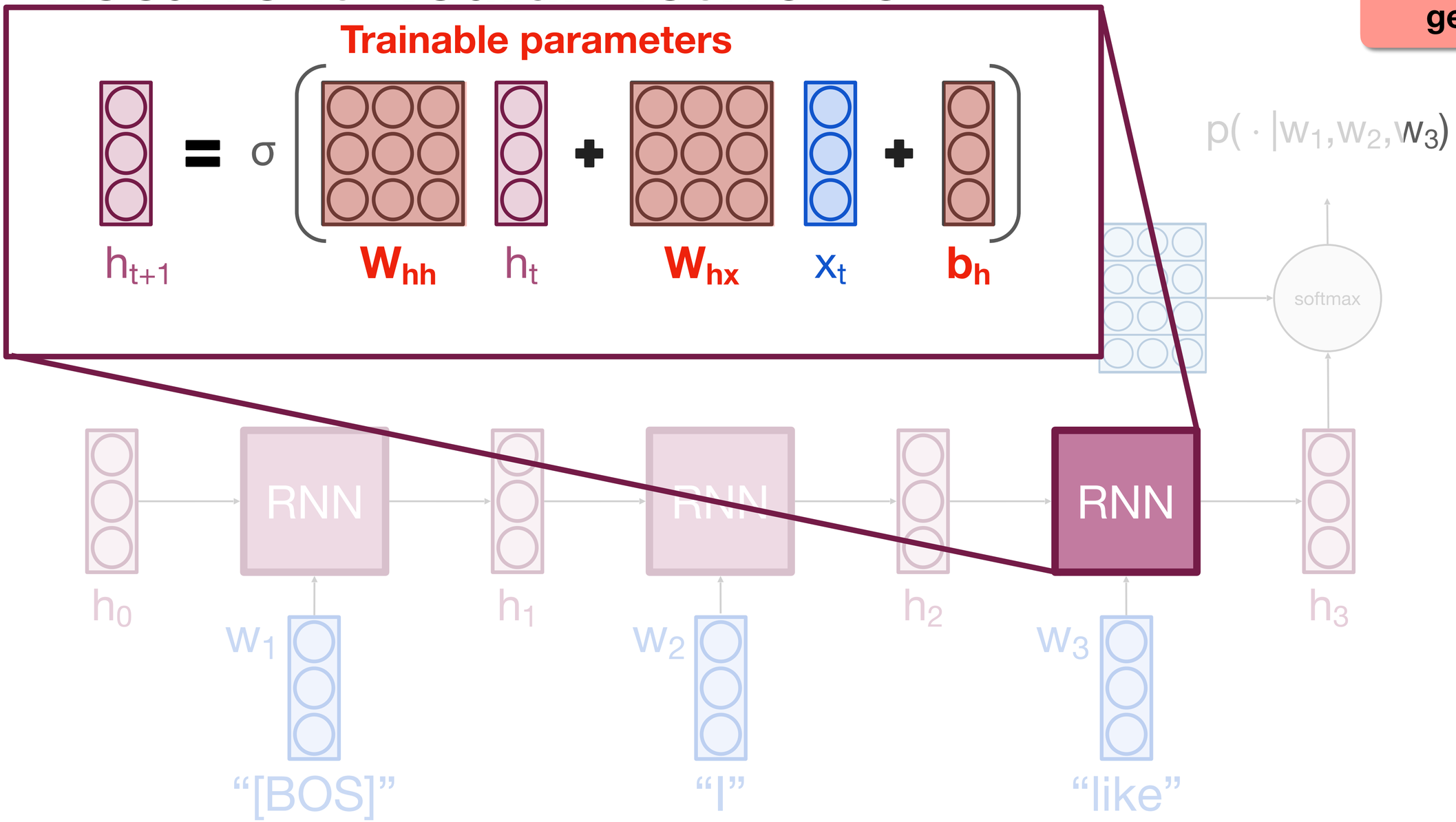
generative



Recurrent Neural Networks

RNN

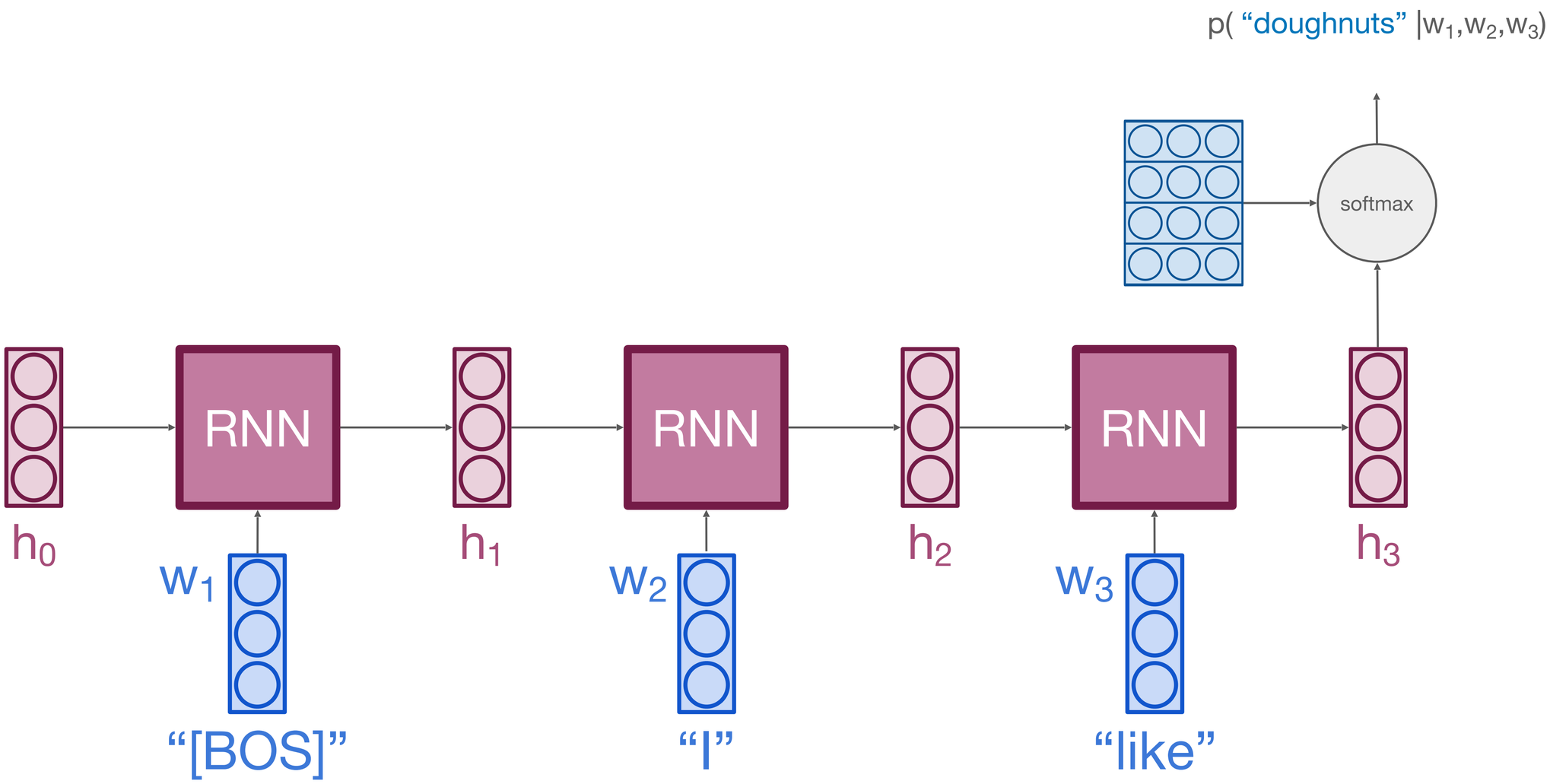
generative



Recurrent Neural Networks

RNN

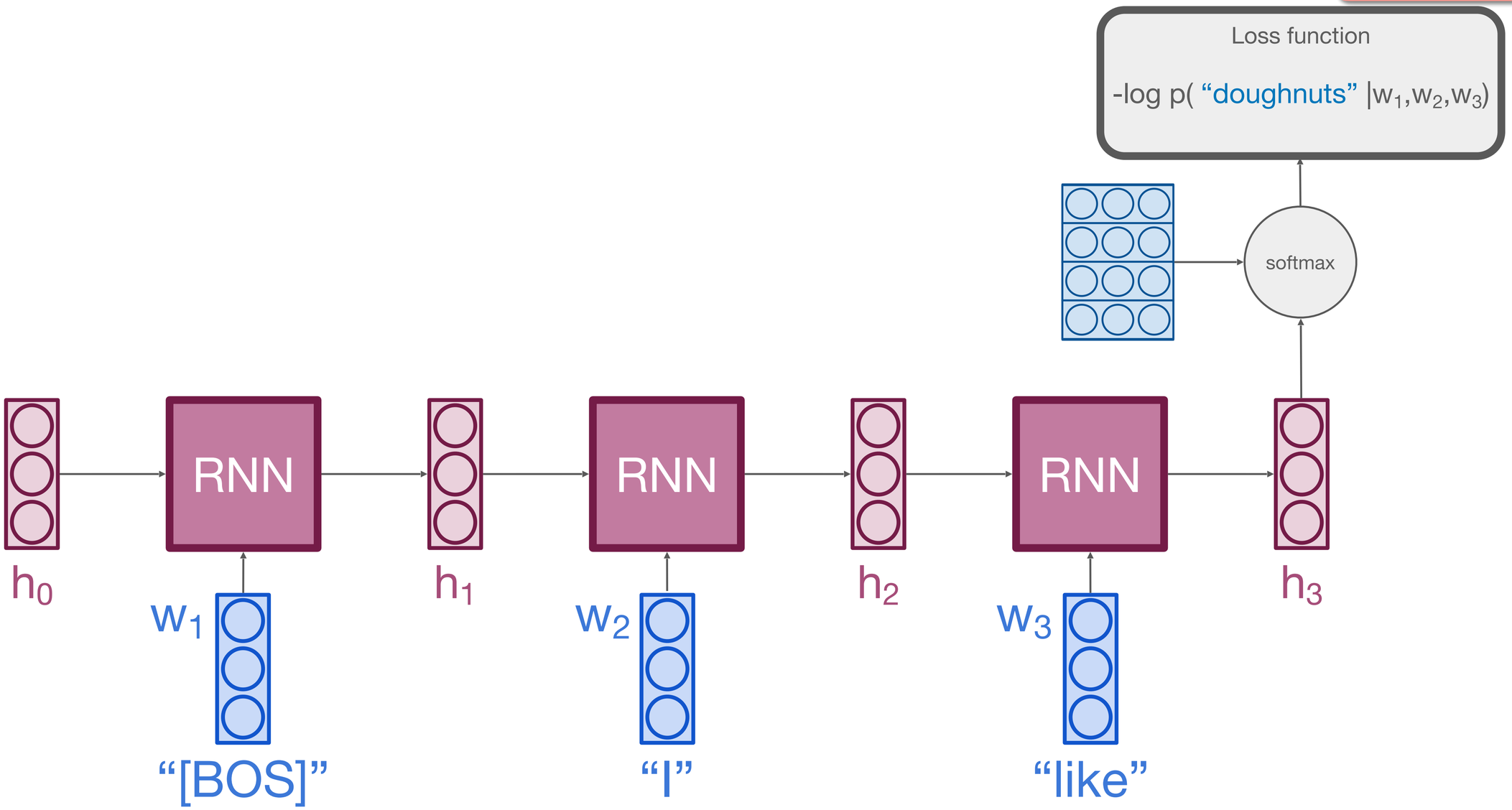
generative



Recurrent Neural Networks

RNN

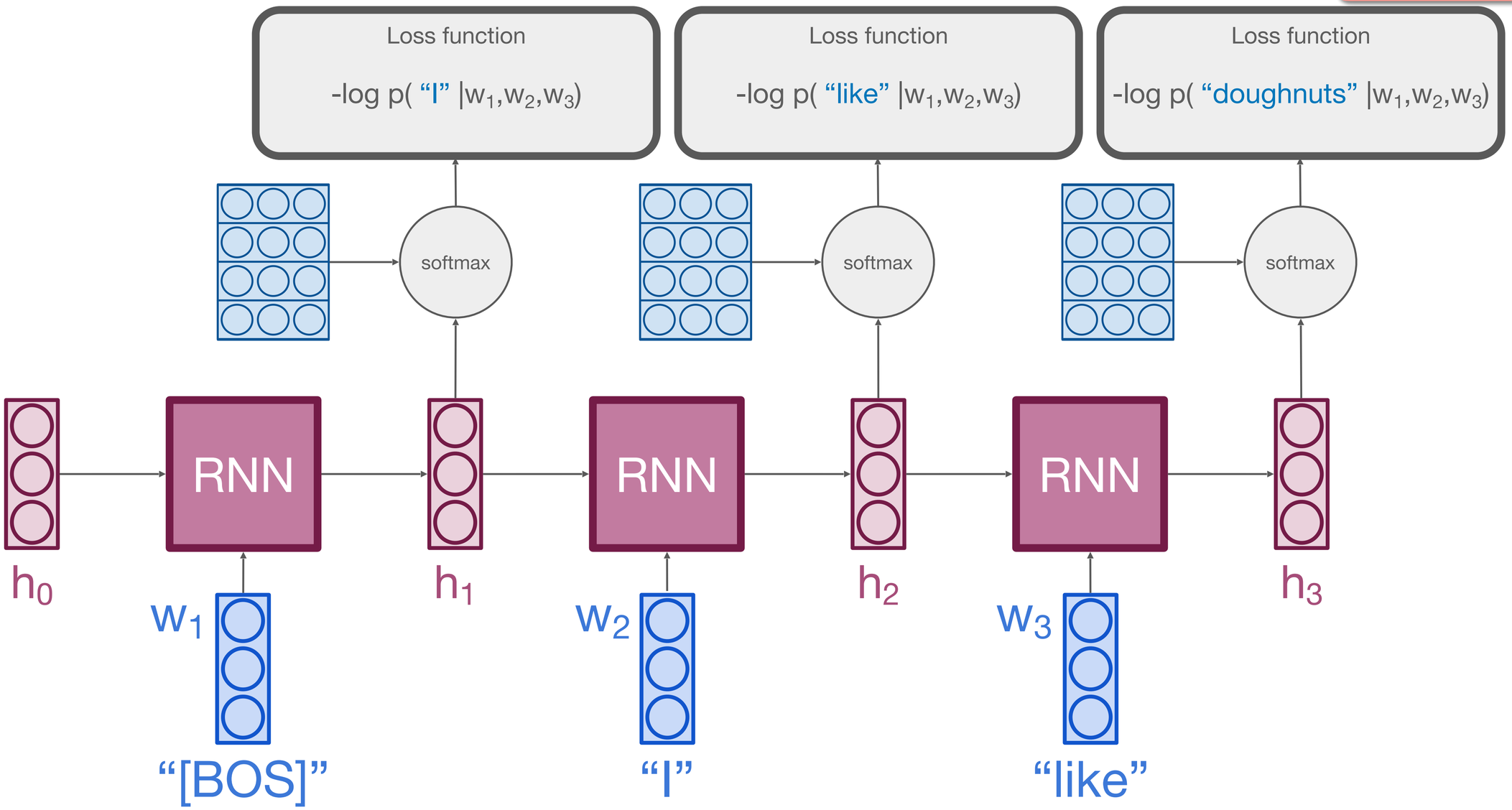
generative



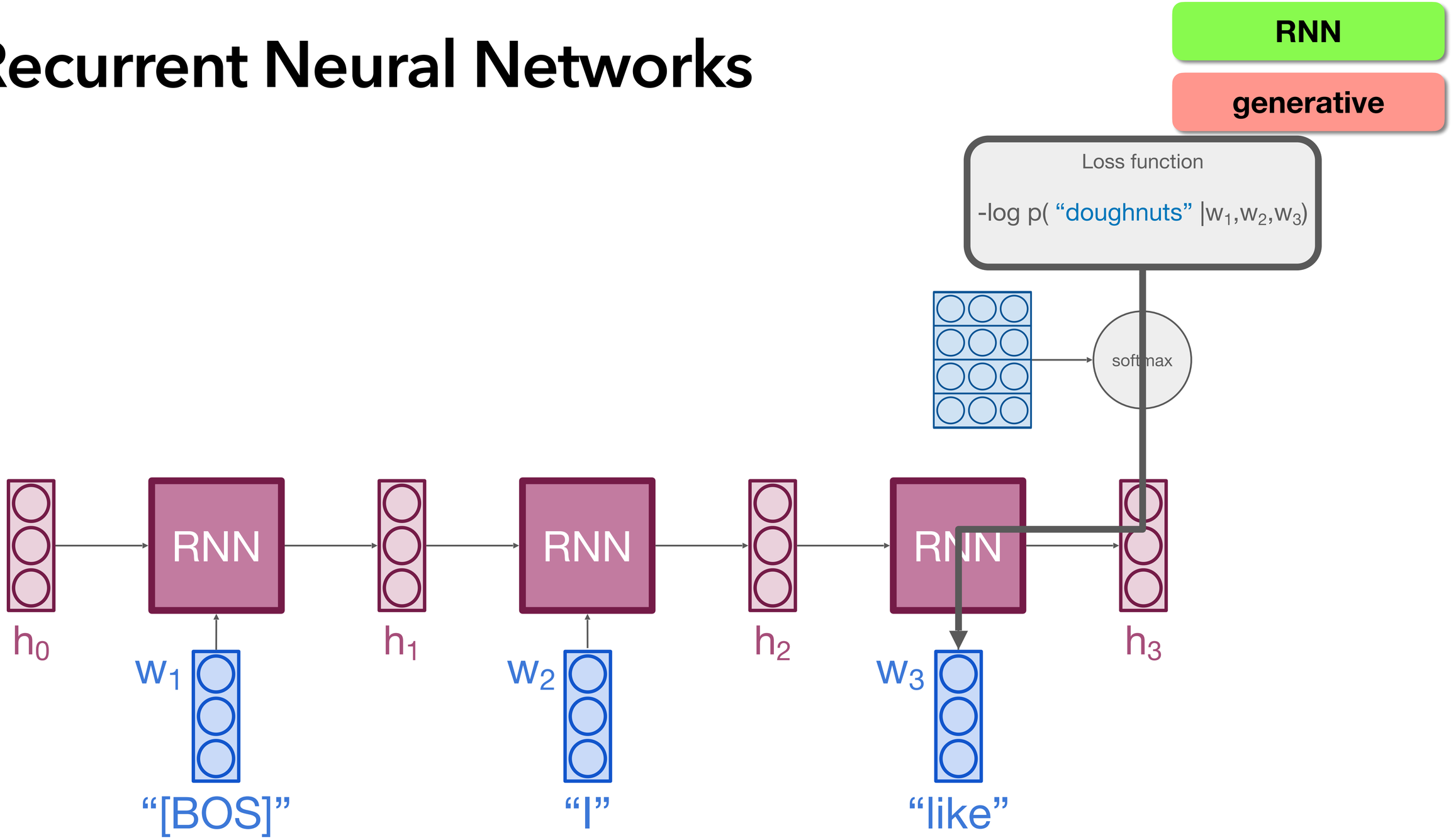
Recurrent Neural Networks

RNN

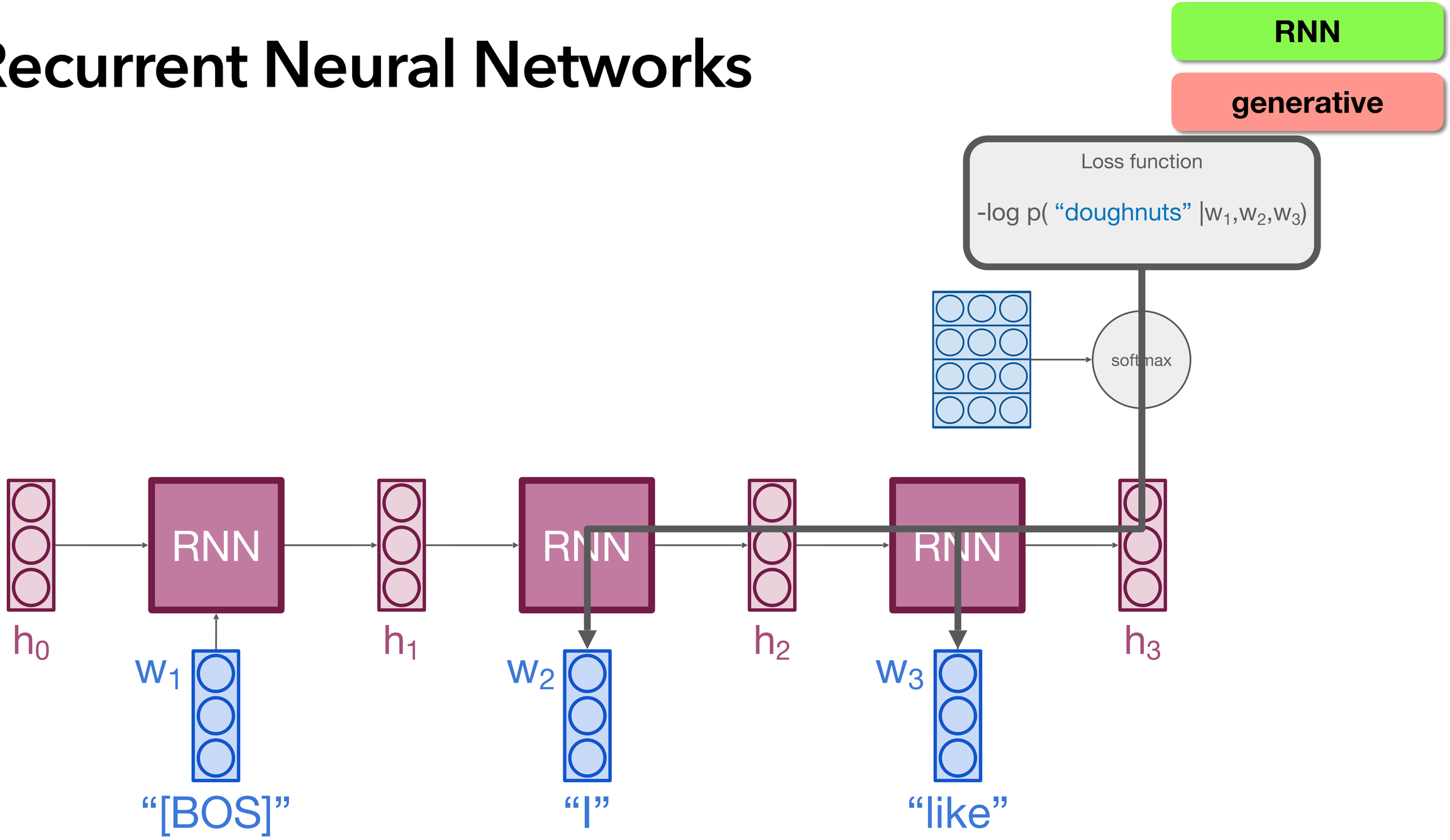
generative



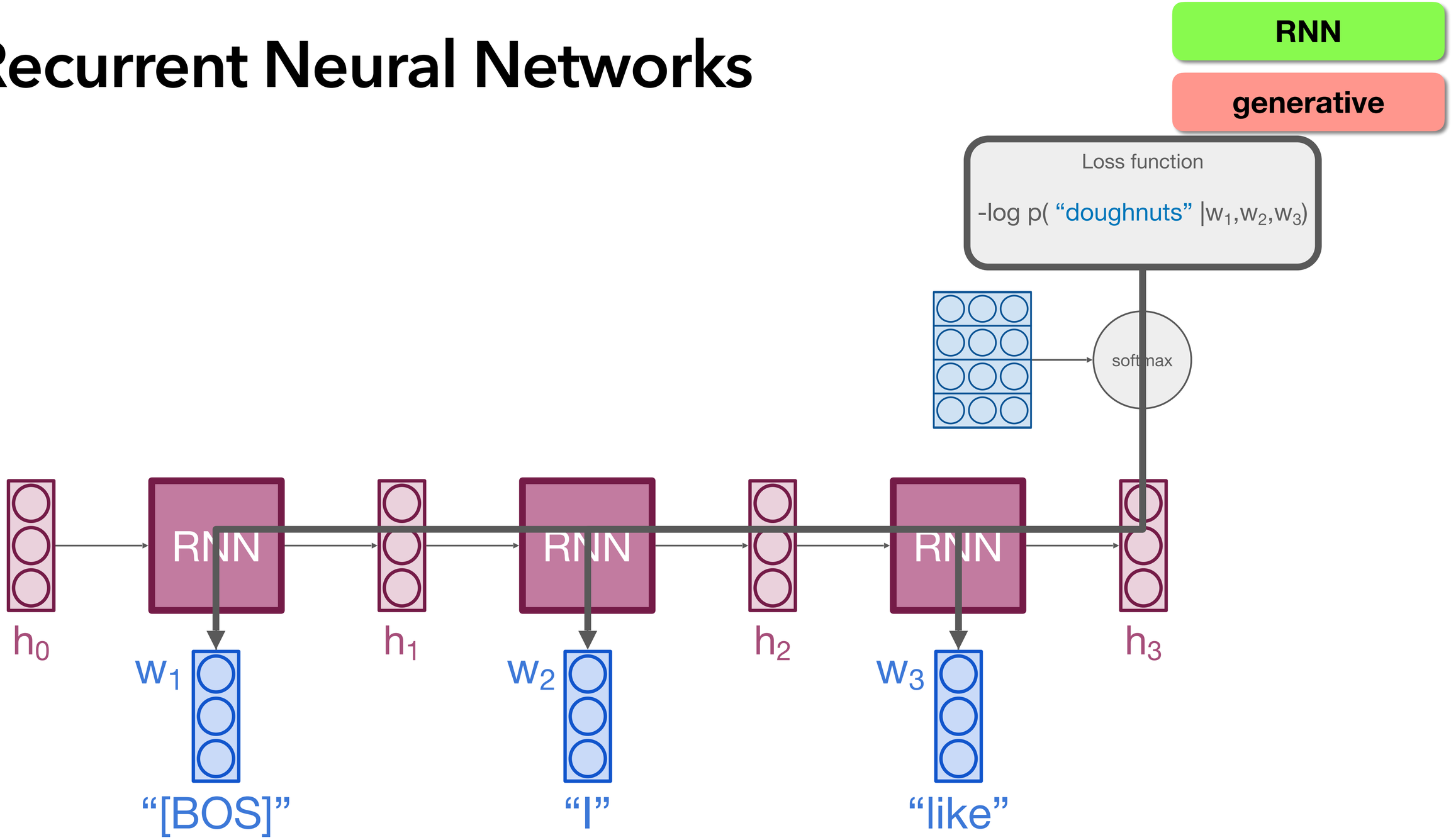
Recurrent Neural Networks



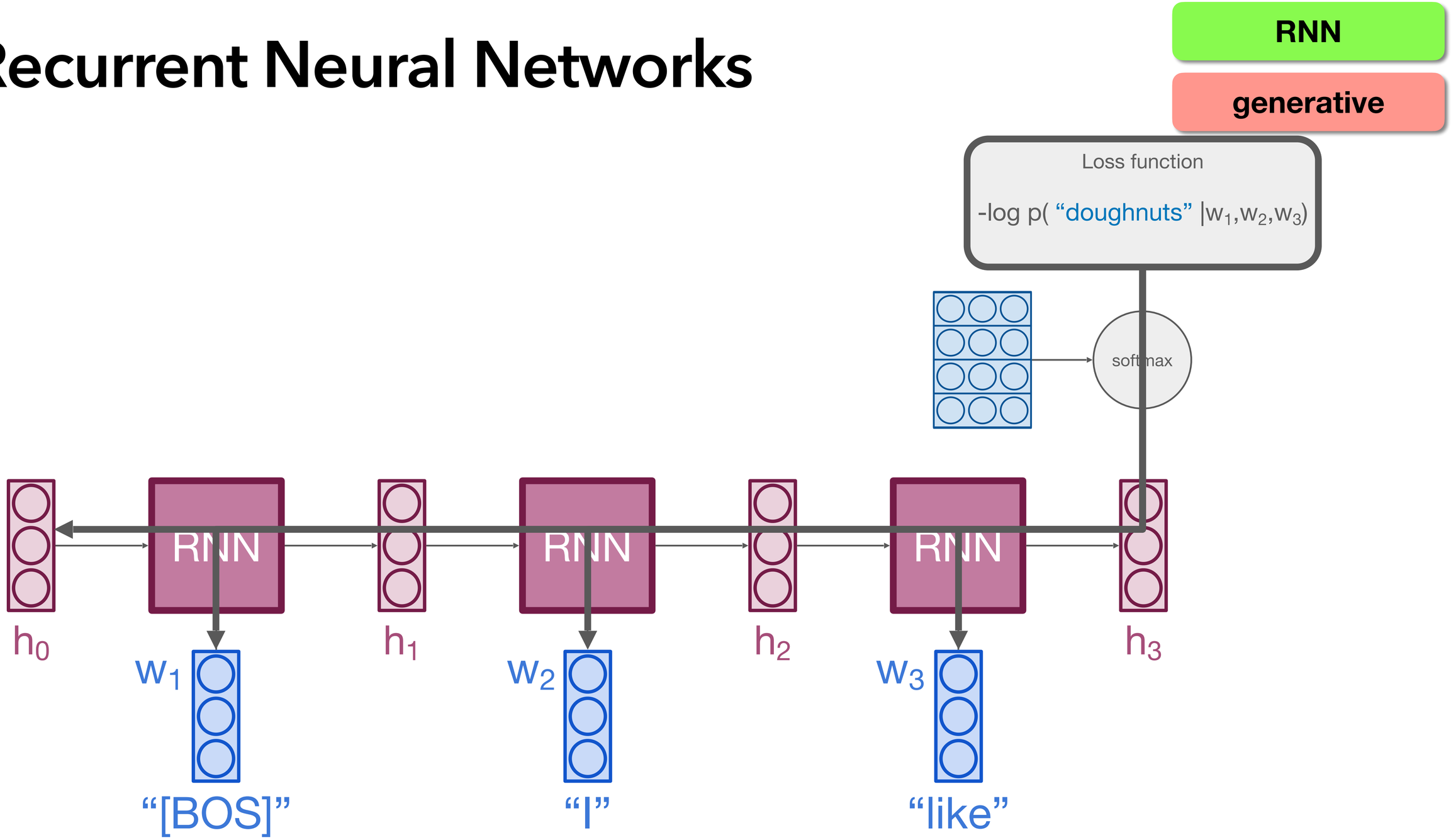
Recurrent Neural Networks



Recurrent Neural Networks



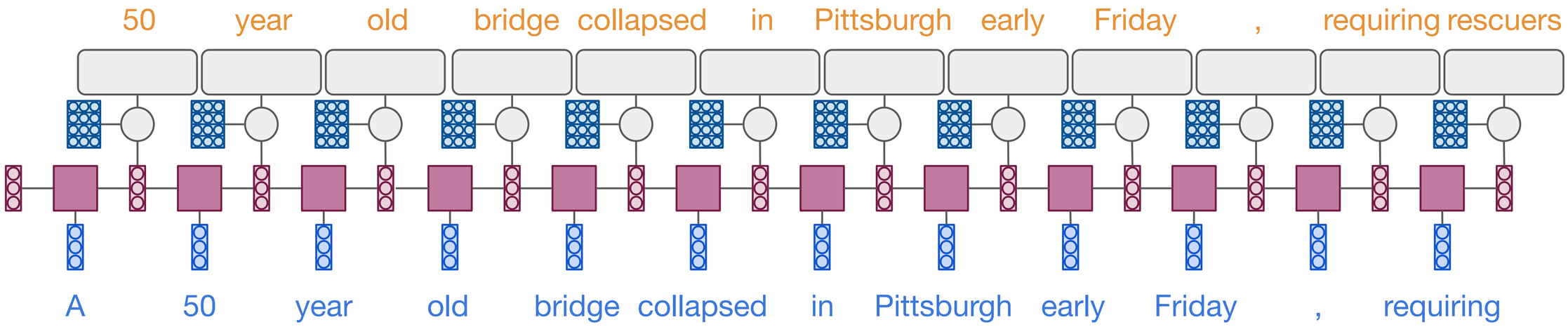
Recurrent Neural Networks



RNNs: long sequences

RNN

generative



RNNs: long sequences

RNN

generative

- Compute gradient on shorter sub-sequence

RNNs: long sequences

RNN

generative

- Compute gradient on shorter sub-sequence

  Update parameters

RNNs: long sequences

RNN

generative

- Compute gradient on shorter sub-sequence
  Keep last hidden state

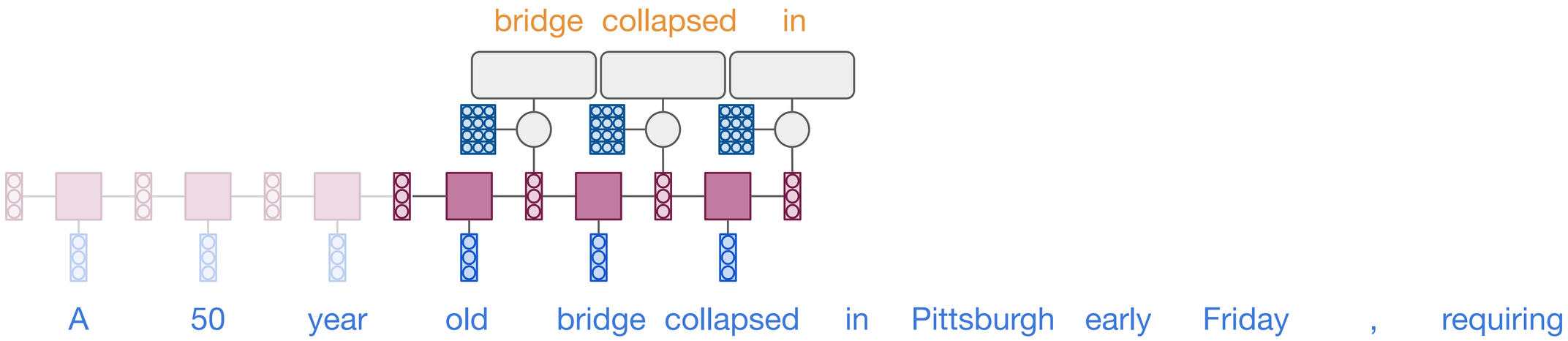
RNNs: long sequences

RNN

generative

- Compute gradient on shorter sub-sequence

  Keep last hidden state



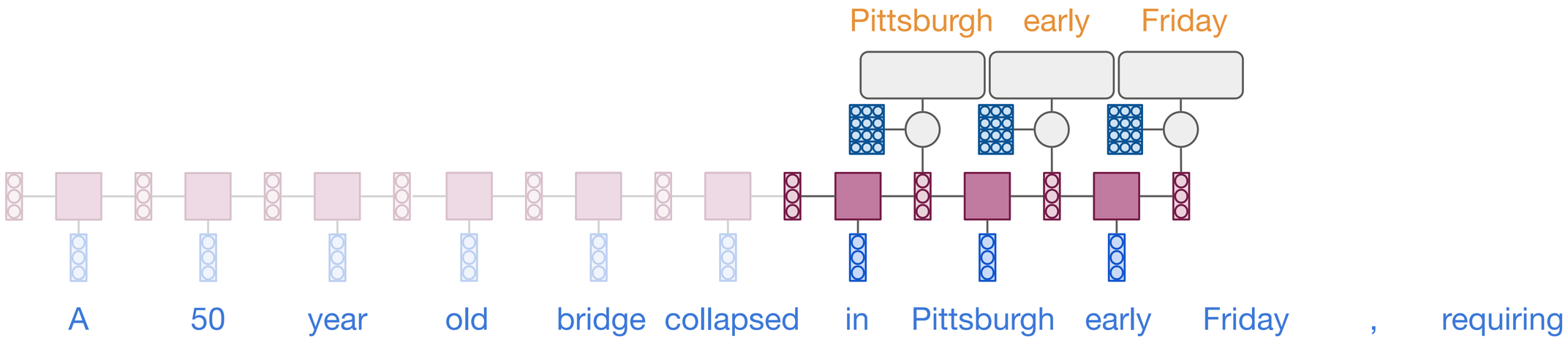
RNNs: long sequences

RNN

generative

- Compute gradient on shorter sub-sequence

  Keep last hidden state



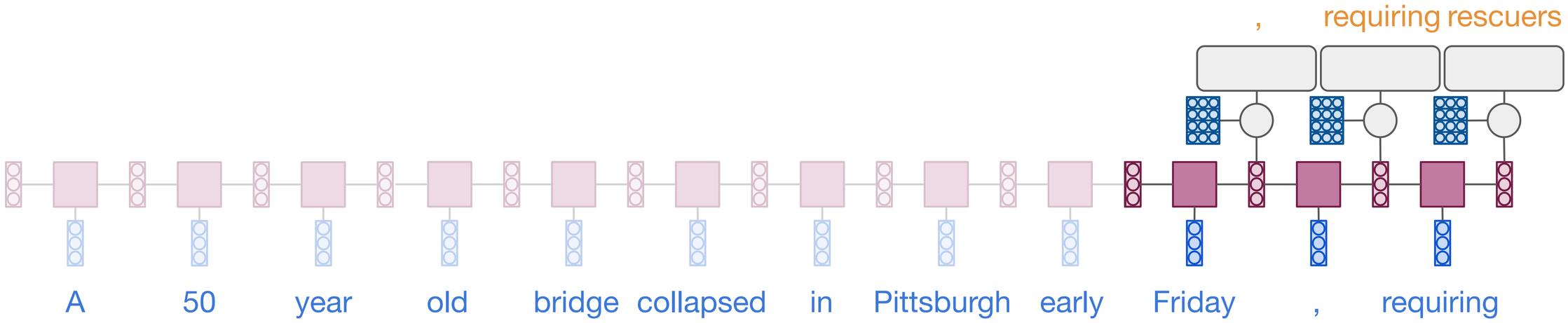
RNNs: long sequences

RNN

generative

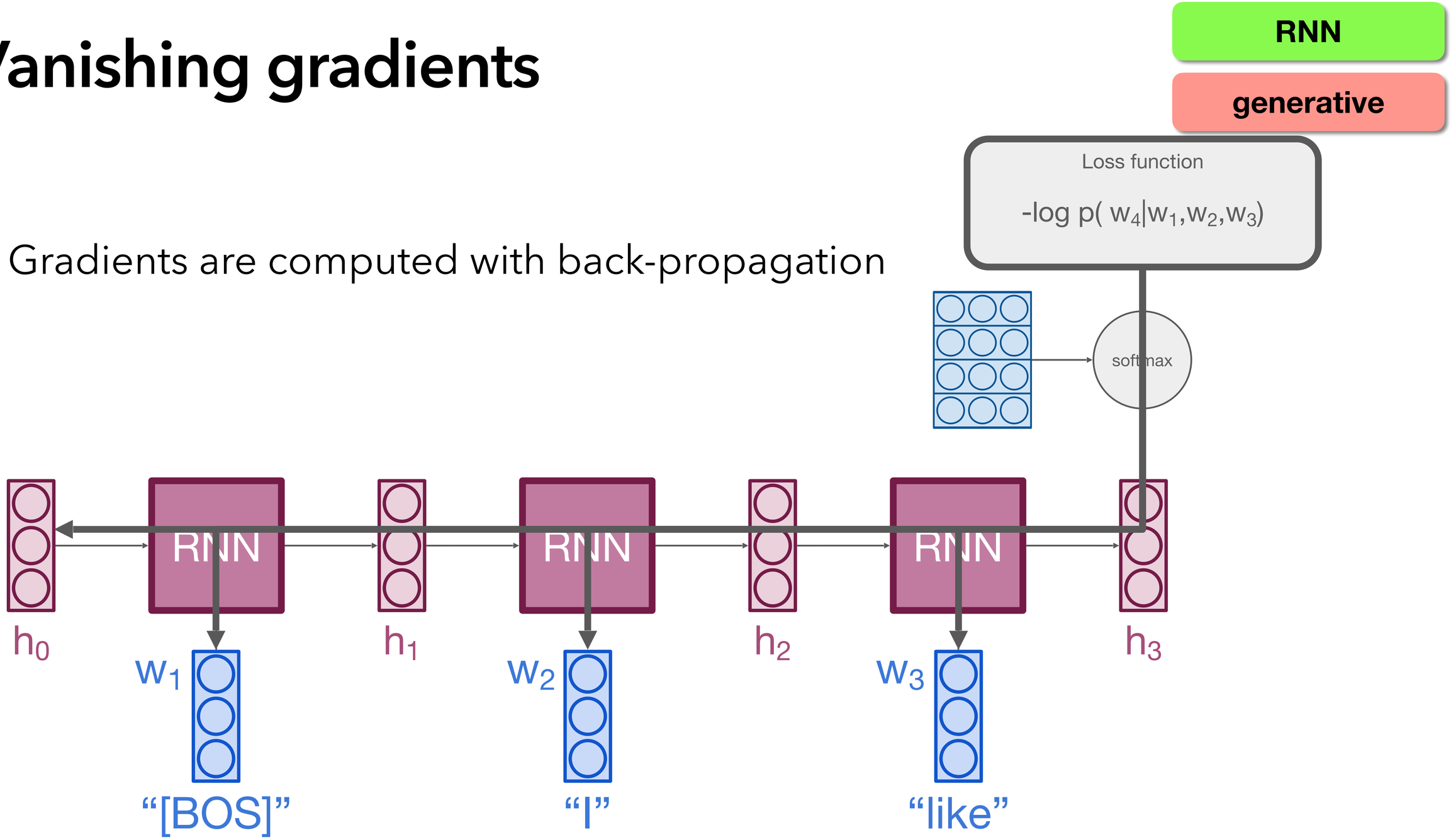
- Compute gradient on shorter sub-sequence

  Keep last hidden state



Vanishing gradients

- Gradients are computed with back-propagation

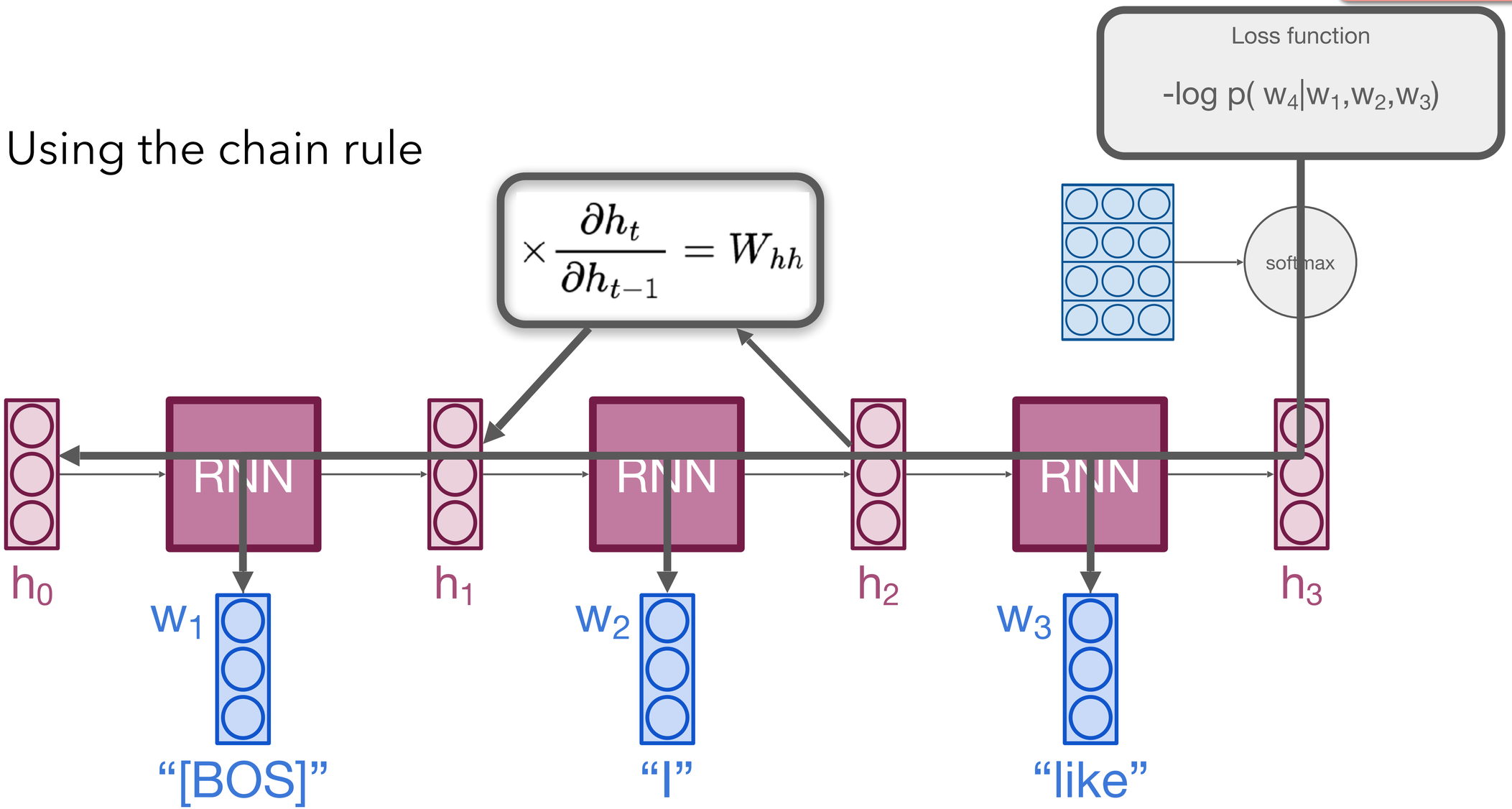


Vanishing gradients

RNN

generative

- Using the chain rule



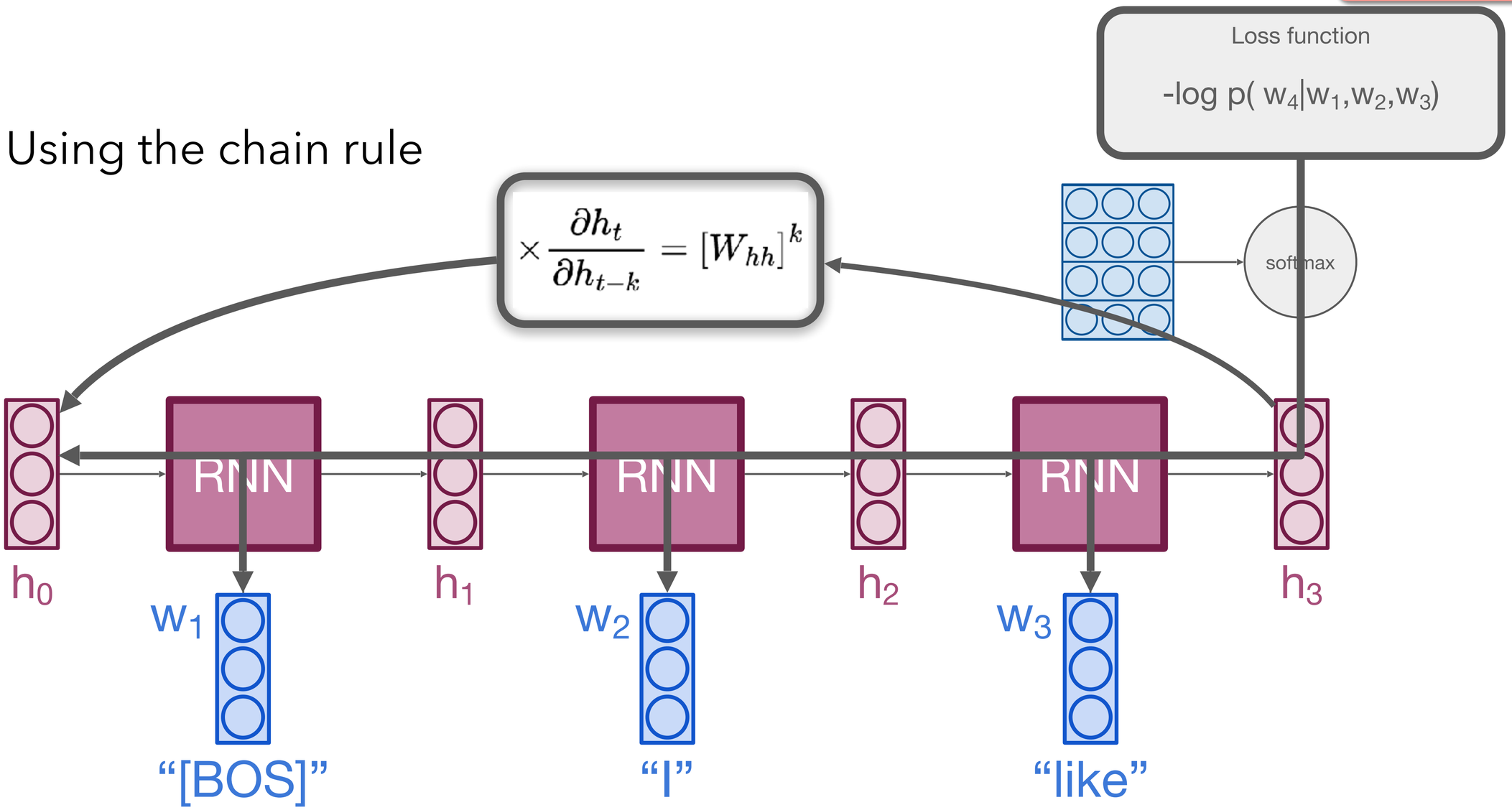
$$\times \frac{\partial h_t}{\partial h_{t-1}} = W_{hh}$$

Vanishing gradients

RNN

generative

- Using the chain rule



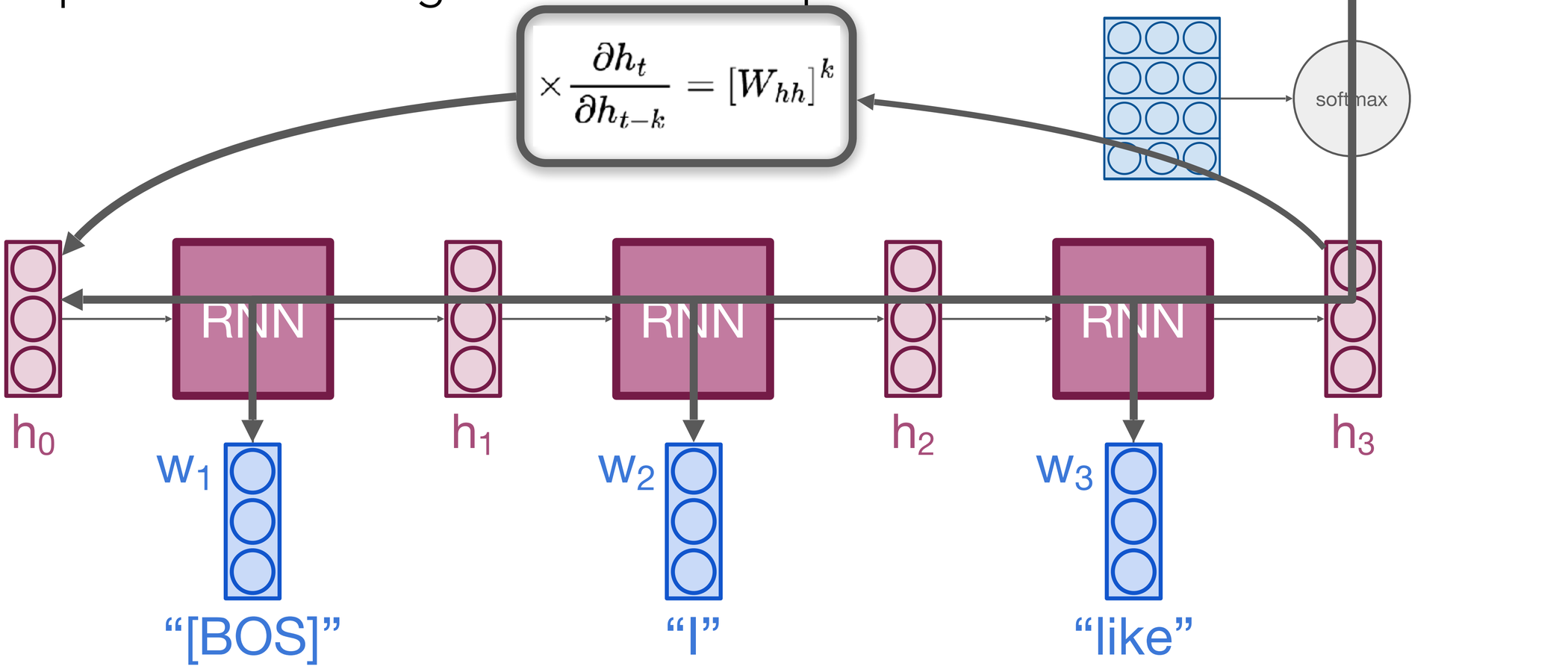
$$\times \frac{\partial h_t}{\partial h_{t-k}} = [W_{hh}]^k$$

Vanishing gradients

RNN

generative

- Explosion/vanishing after several steps



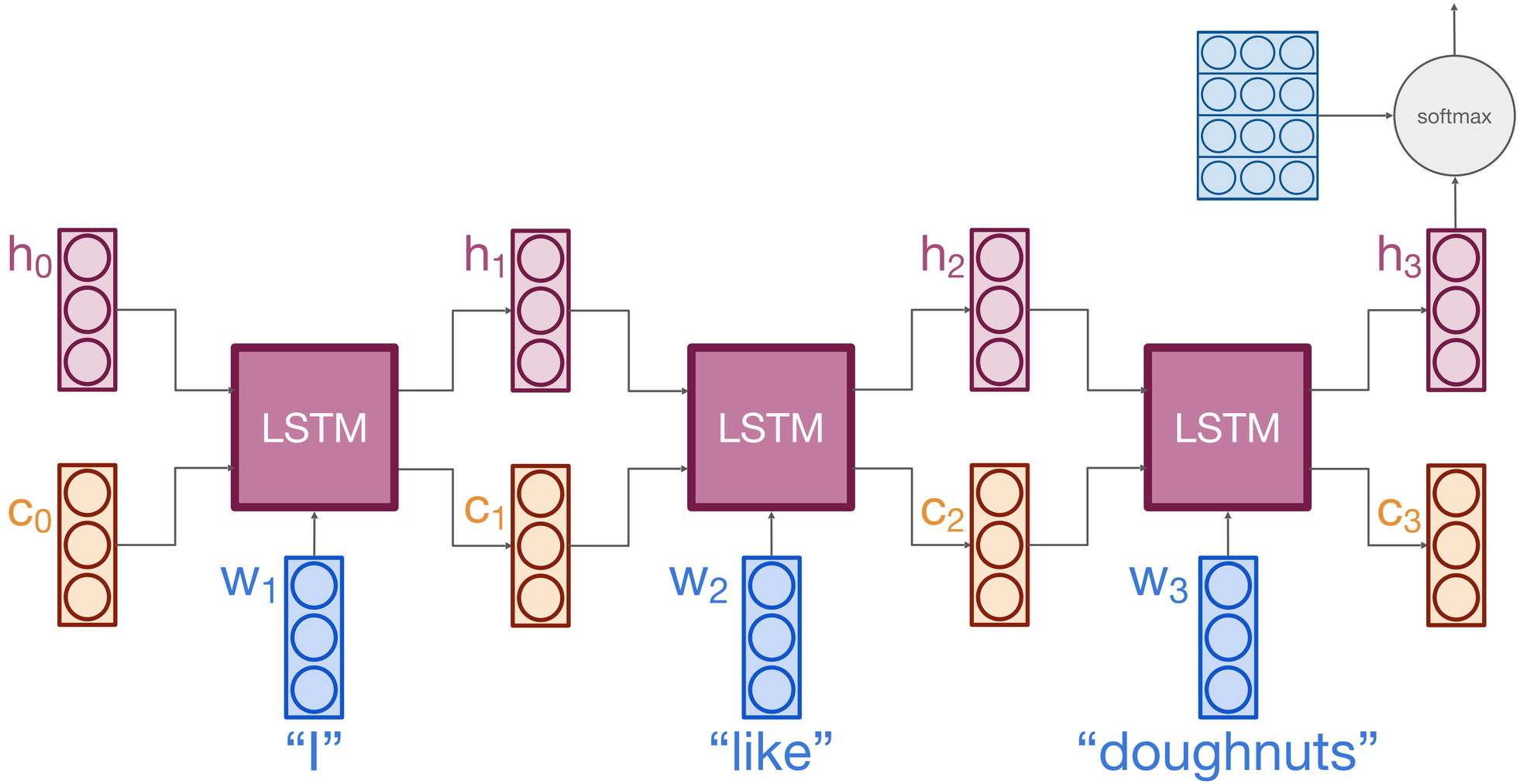
Long Short-Term Memory networks

LSTMs (Hochreiter & Schimdhuber 1997)

RNN

generative

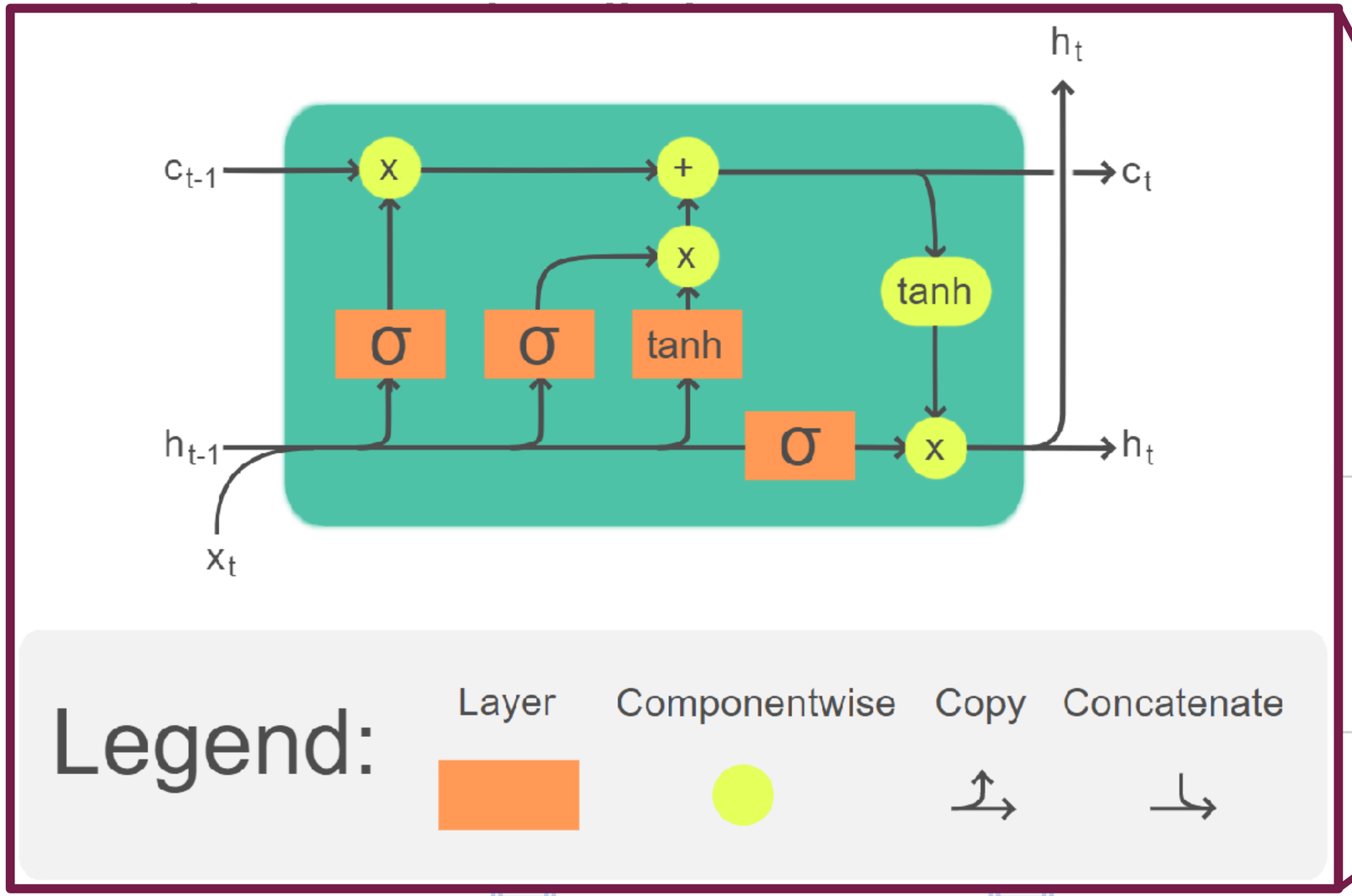
$p(\cdot \mid w_1, w_2, w_3)$

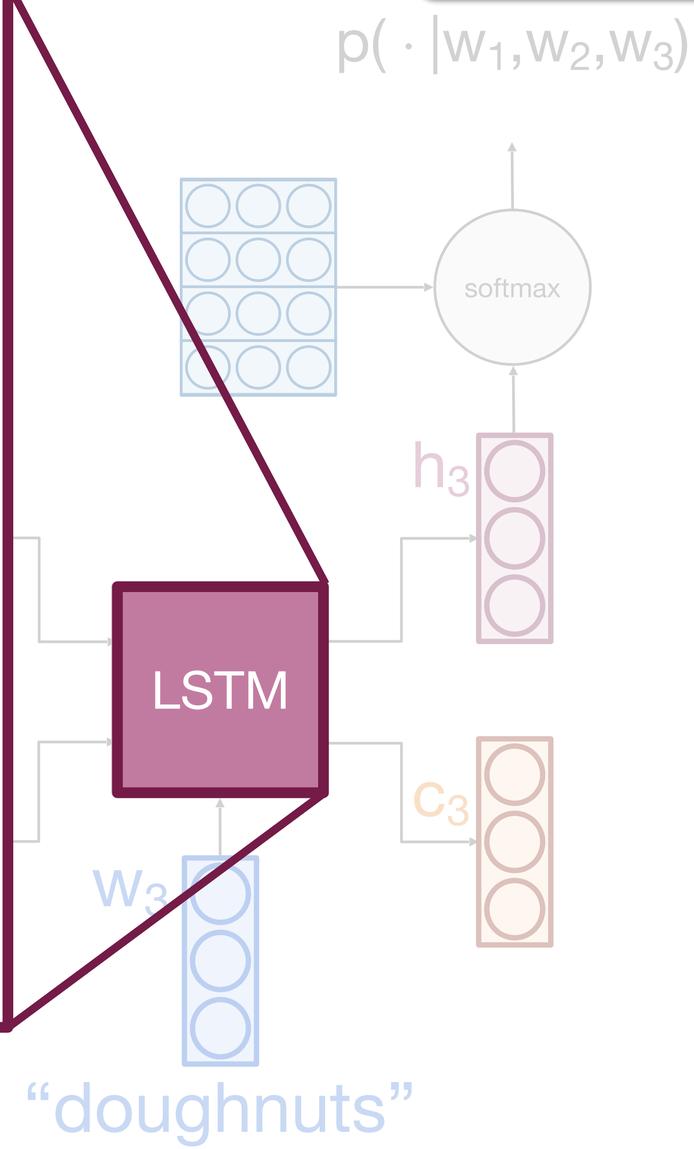


Long Short-Term Memory networks

RNN

generative





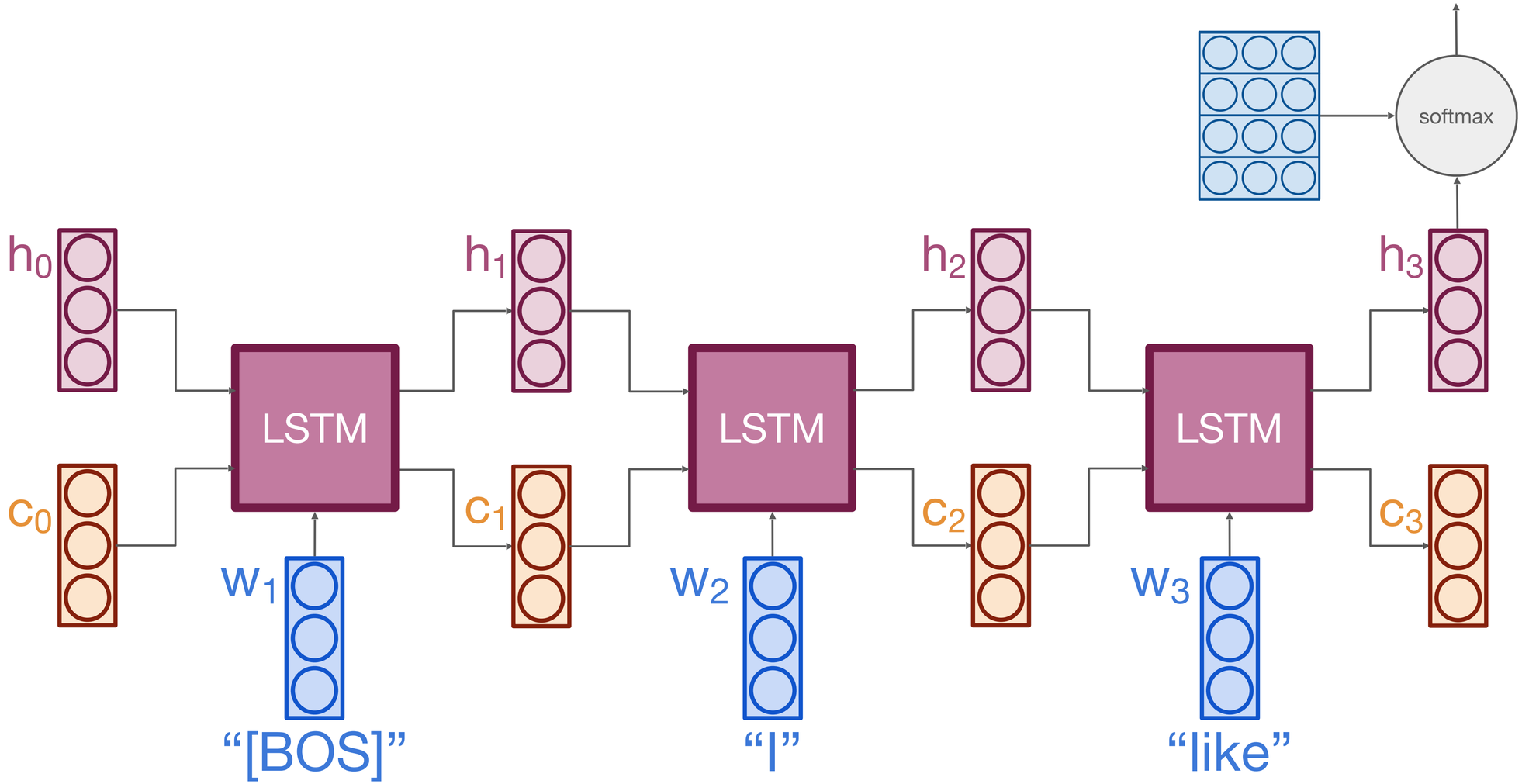
Long Short-Term Memory networks

LSTMs (Hochreiter & Schimdhuber 1997)

RNN

generative

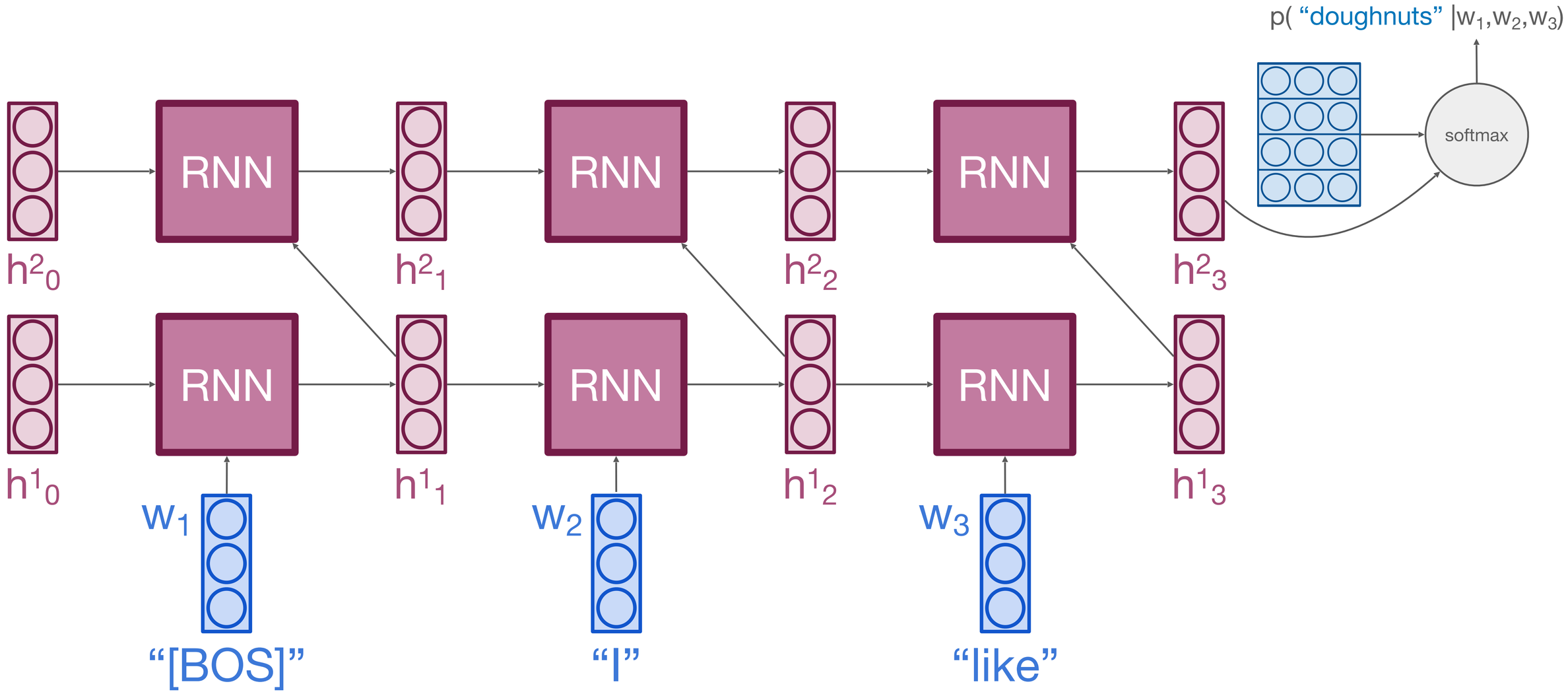
$p(\cdot \mid w_1, w_2, w_3)$



Multilayer RNNs

RNN

generative



Limits of RNNs

RNN

generative

- How much context do they use?
  - Theoretically unbounded
  - In practice, quite limited, even with LSTMs
- **Scalability issues**
  - Need to encode each token sequentially
  - Hard to parallelise during training
  - Difficult to scale to >8 layers (vanishing gradients, etc.)

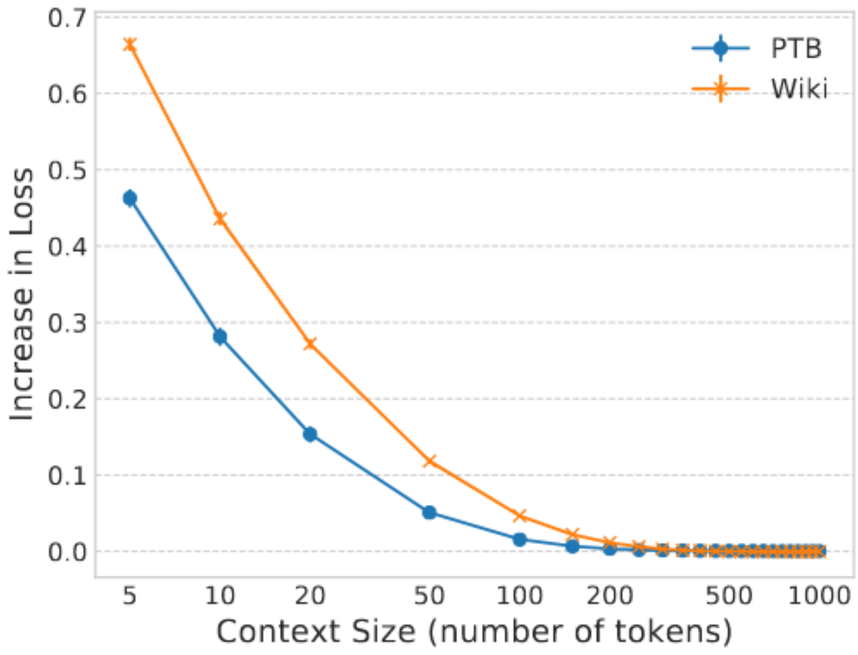


Figure from (Khandelwal et al. 2018)

RNN

MLM

Bidirectionnel recurrent architectures for Masked LMs

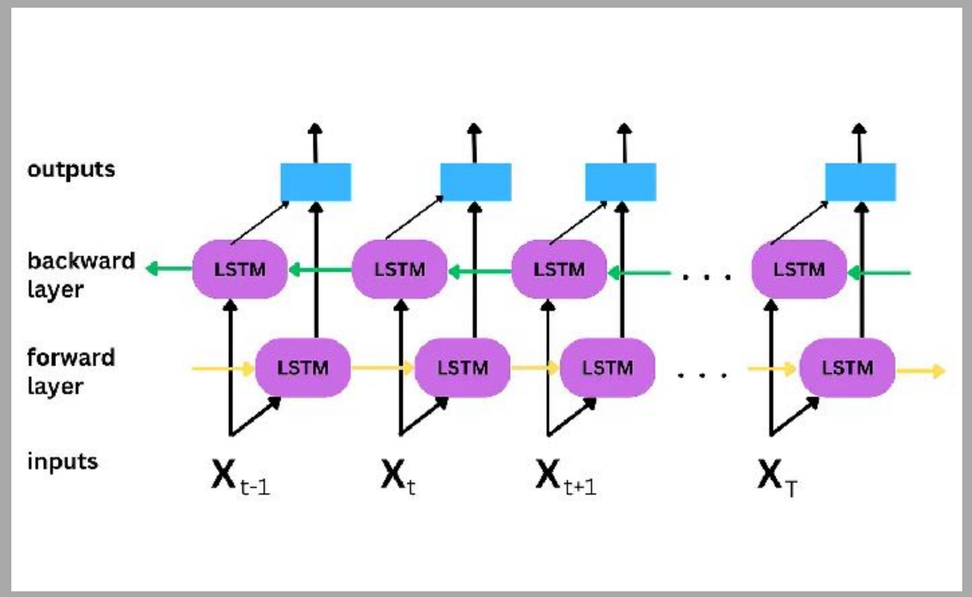
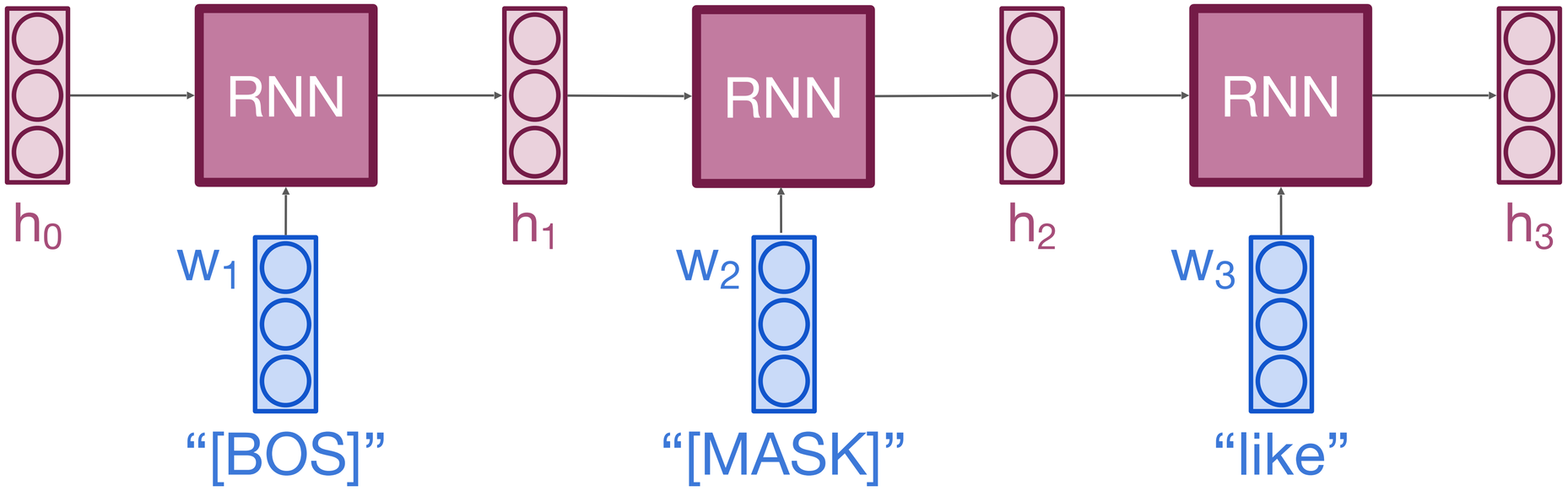


Figure from https://medium.com/@souro400.nath/why-is-bilstm-better-than-lstm-a7eb0090c1e4

MLM with an RNN

RNN

MLM

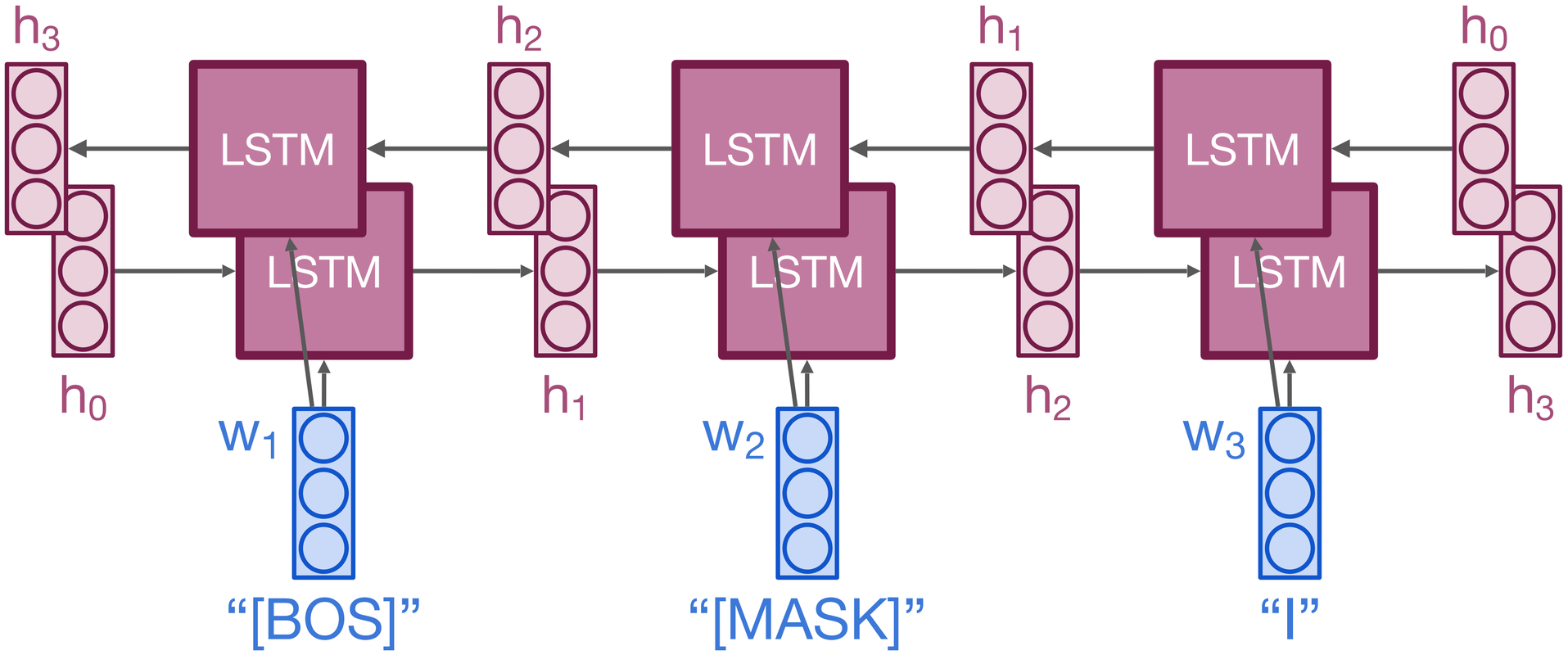


MLM with an RNN

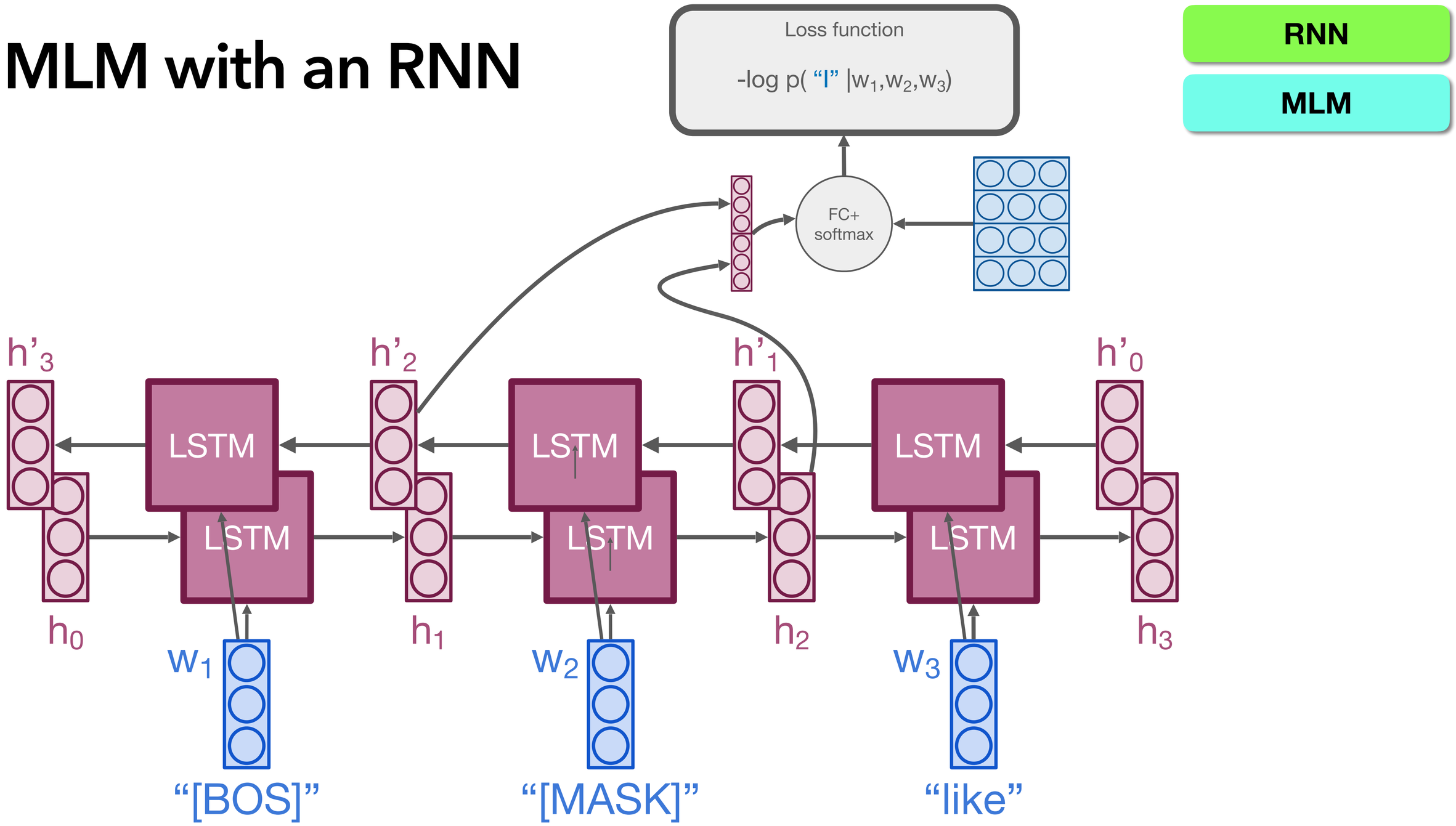
RNN

MLM

- A BiLSTM is made of two independent LSTM models whose outputs are concatenated after their last layer



MLM with an RNN



RNN

seq2seq

Recurrent architectures for sequence-to-sequence LMs

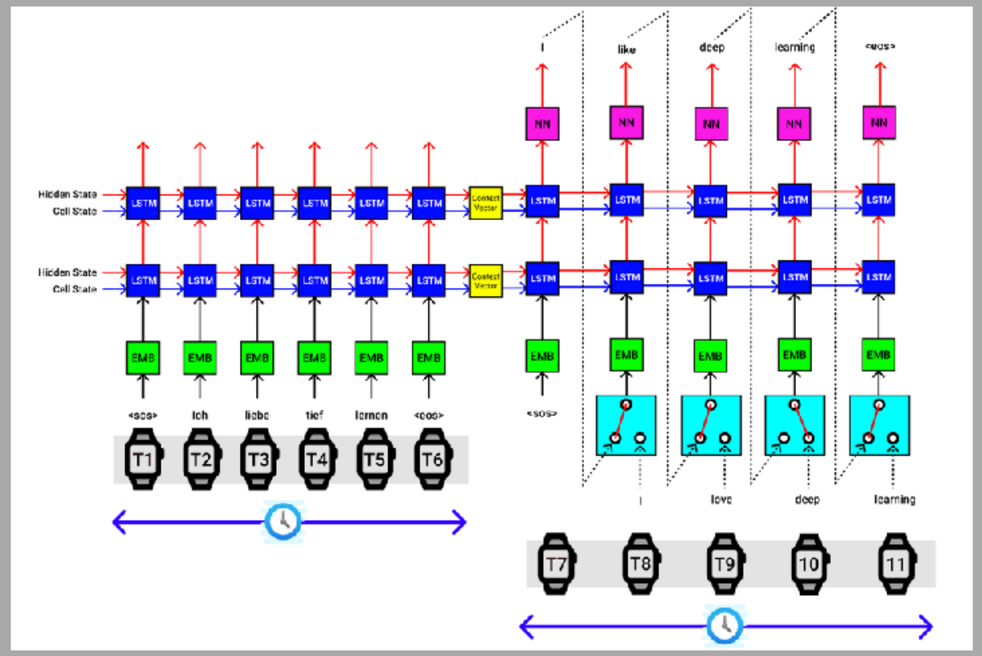


Figure from https://towardsdatascience.com/a-comprehensive-guide-to-neural-machine-translation-using-seq2sequence-modelling-using-pytorch-41c9b84ba350/

Seq2seq RNNs

RNN

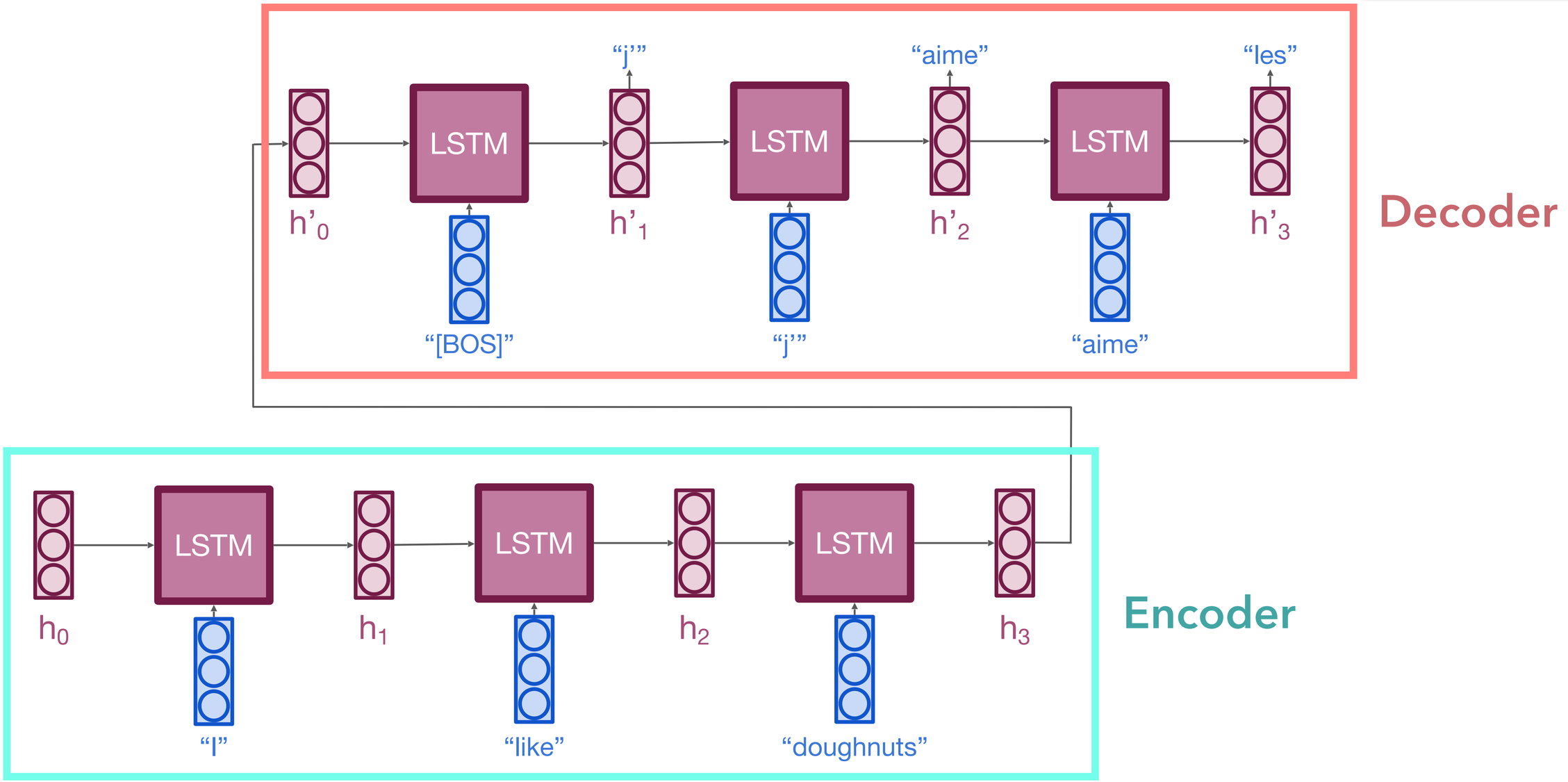
seq2seq

- We want to learn a function f that maps a sequence of tokens X to a sequence of tokens Y. Usually for machine translation (see class on MT)

Vanilla seq2seq RNN

RNN

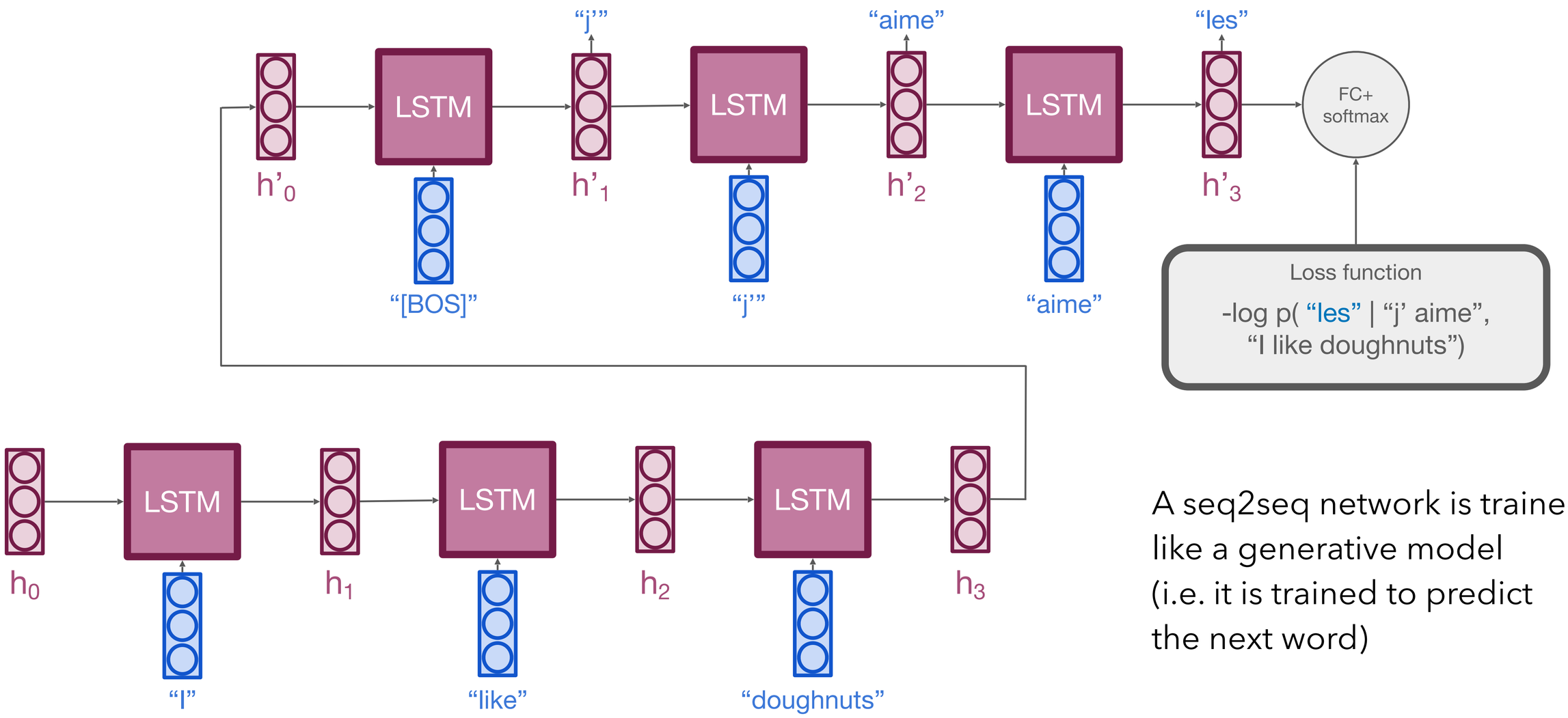
seq2seq



Vanilla seq2seq RNN

RNN

seq2seq

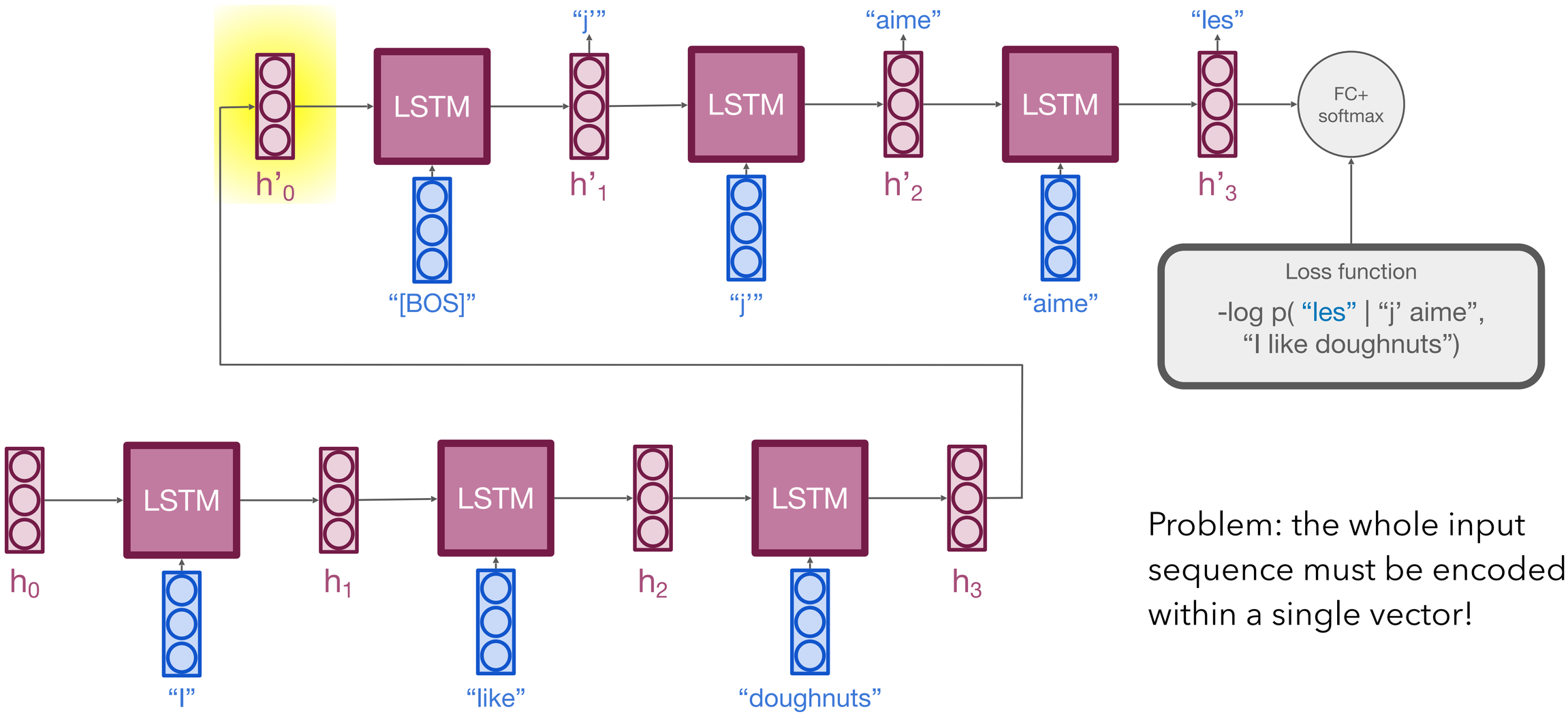


A seq2seq network is trained like a generative model (i.e. it is trained to predict the next word)

Vanilla seq2seq RNN

RNN

seq2seq

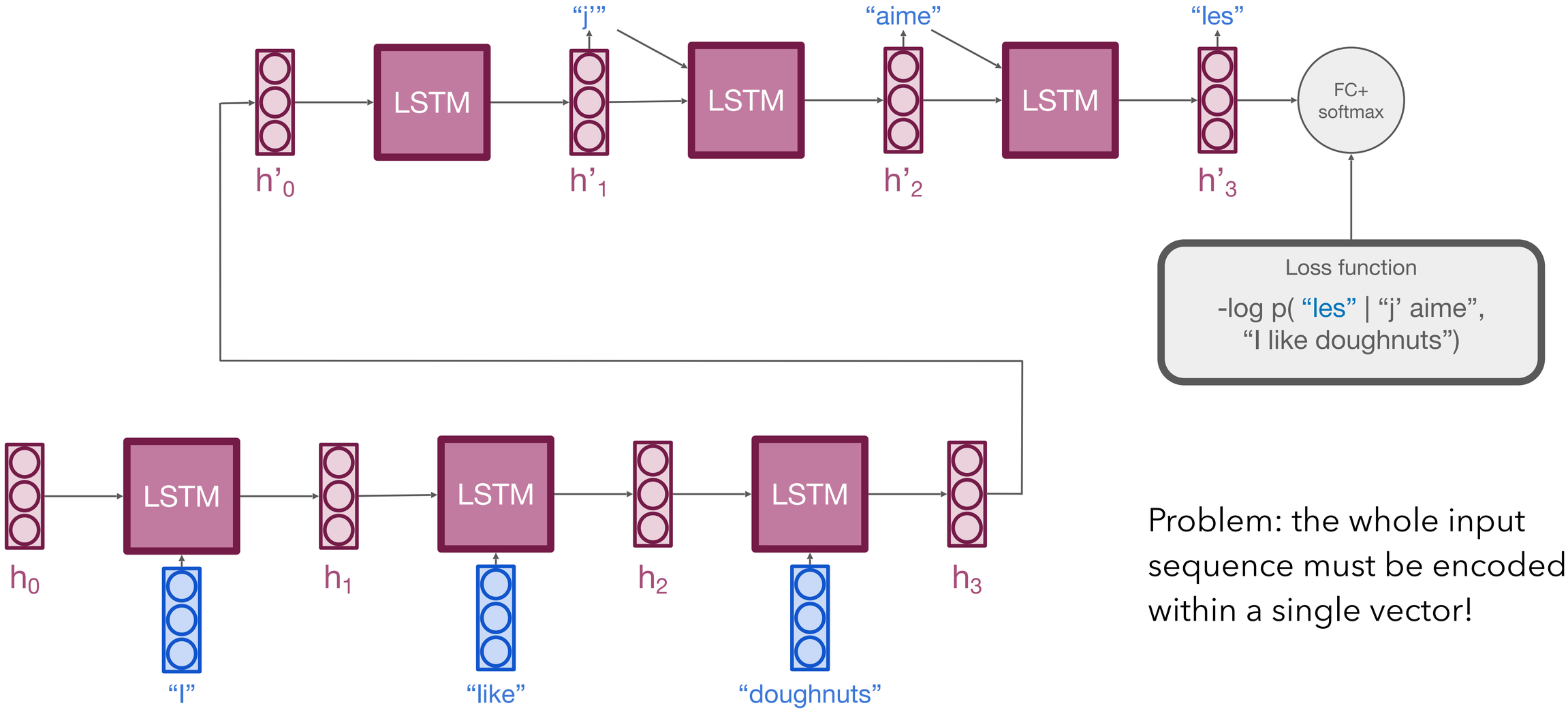


Problem: the whole input sequence must be encoded within a single vector!

Vanilla seq2seq RNN

RNN

seq2seq



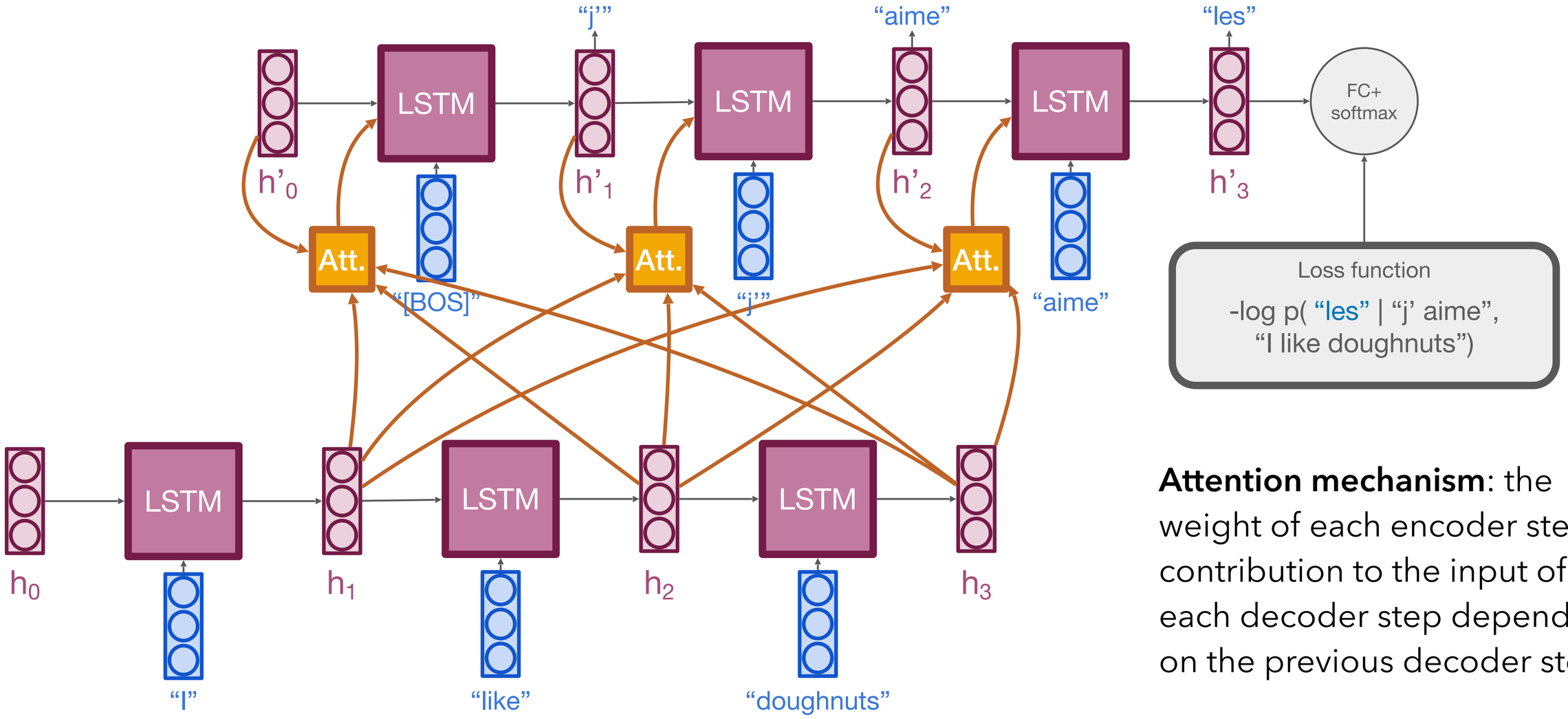
Problem: the whole input sequence must be encoded within a single vector!

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



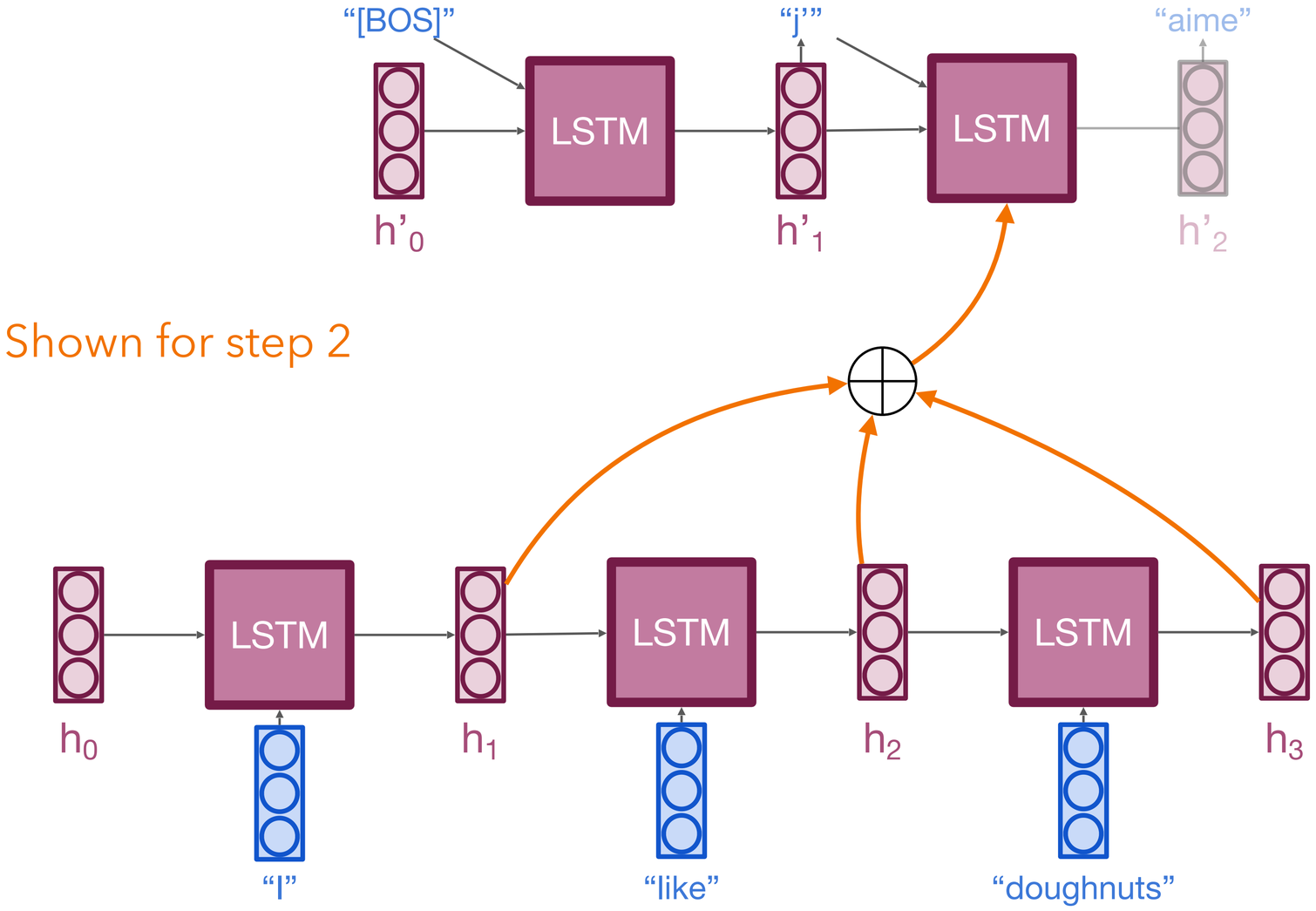
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



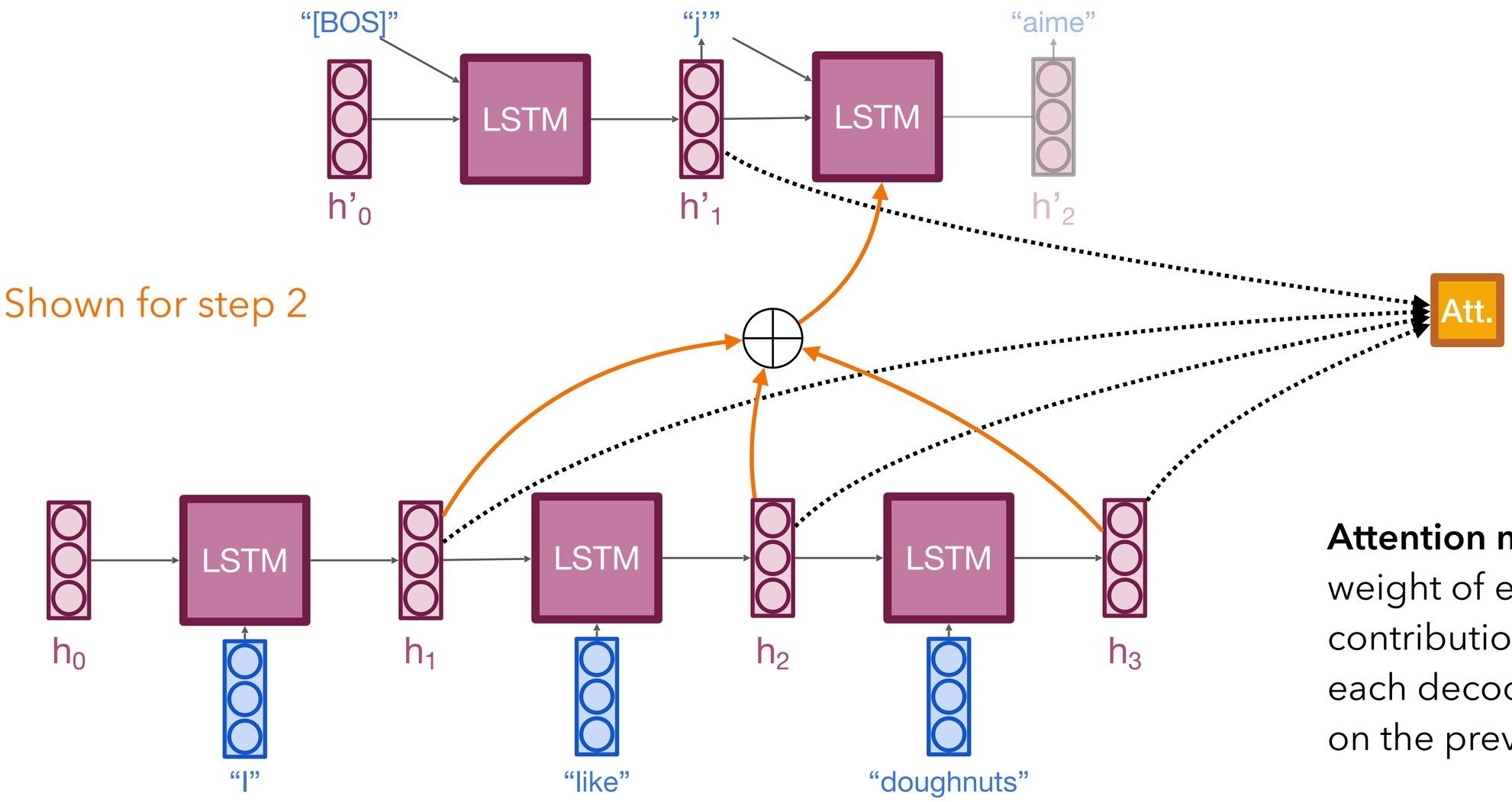
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



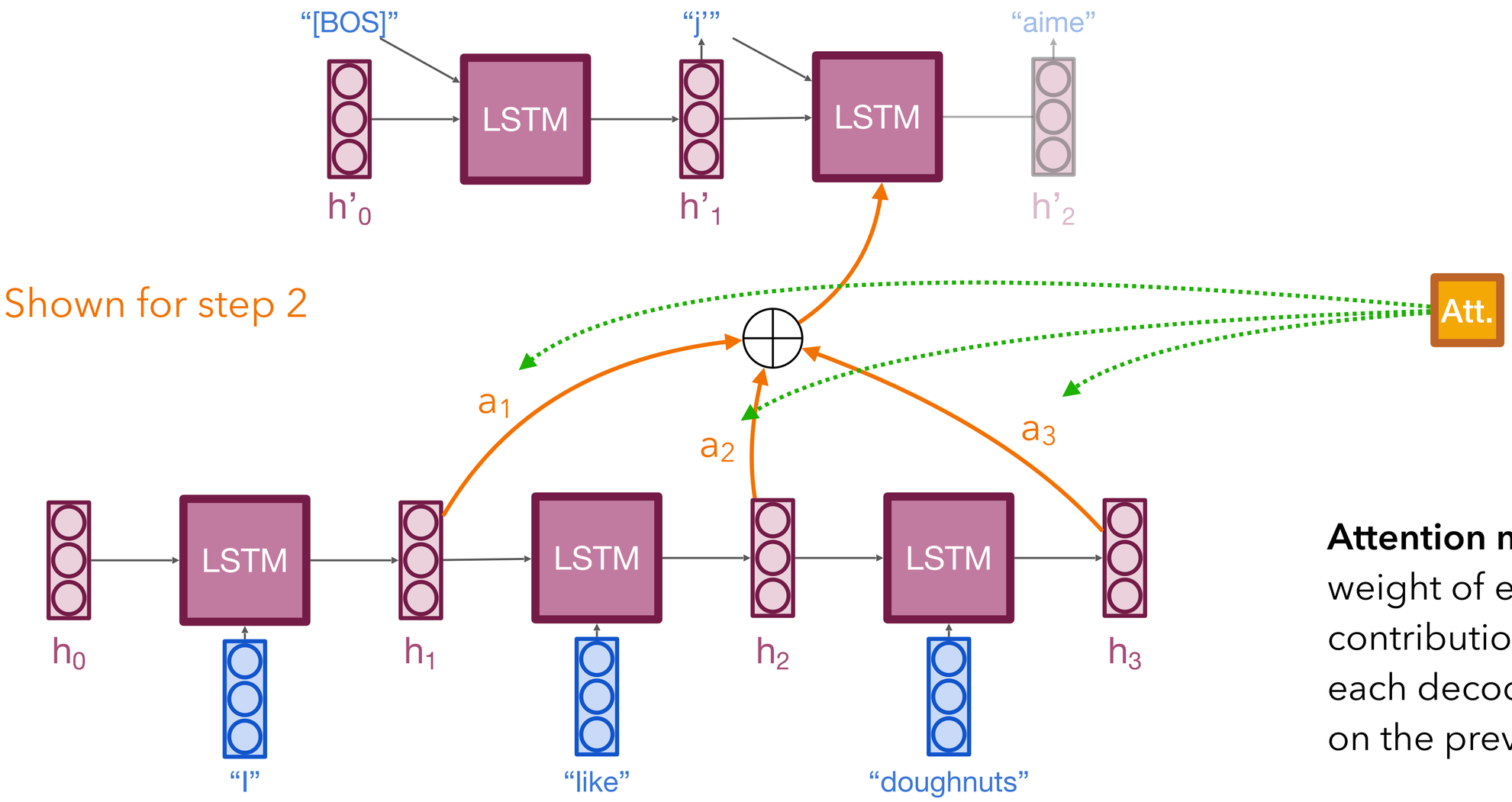
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



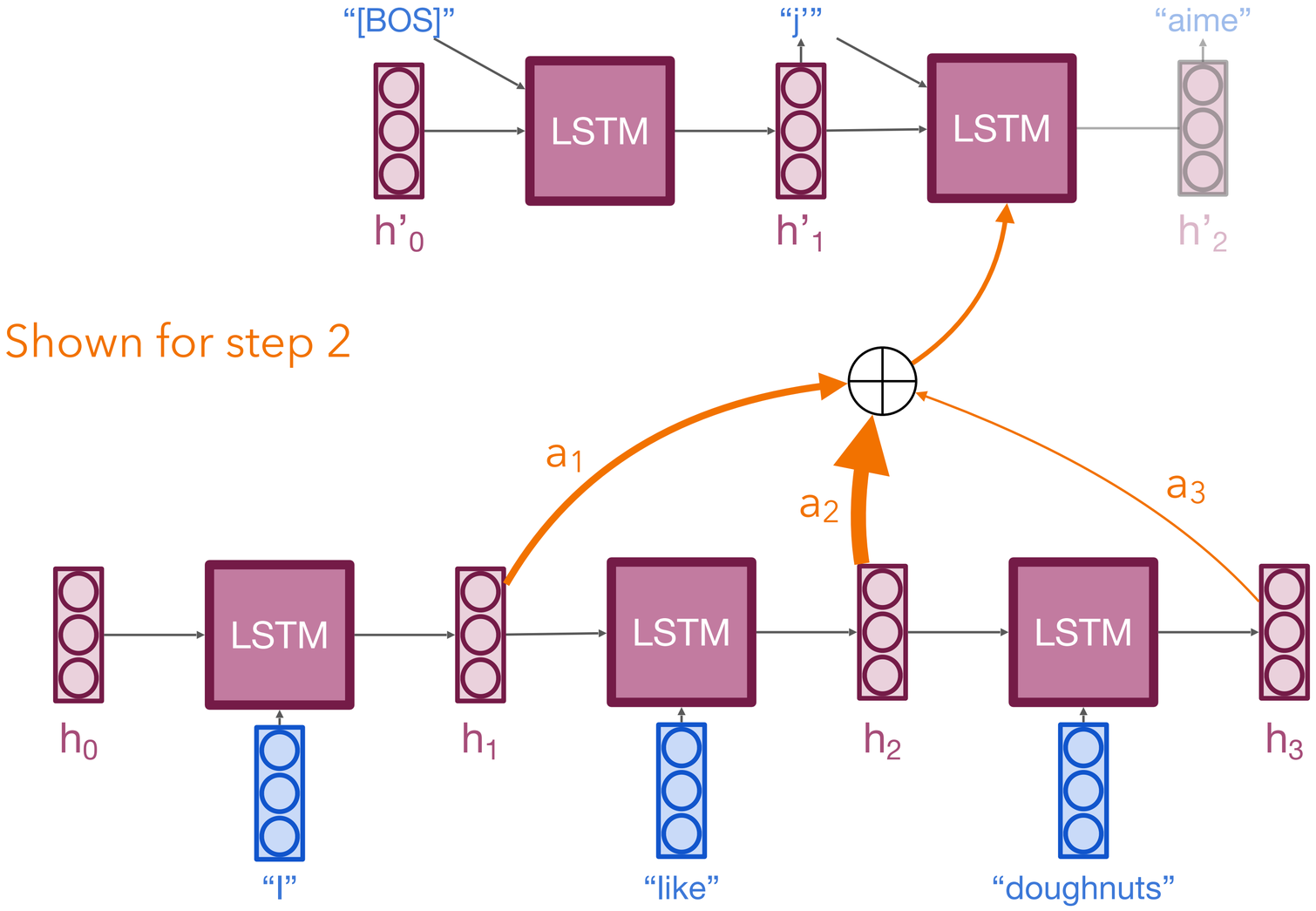
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



Att.

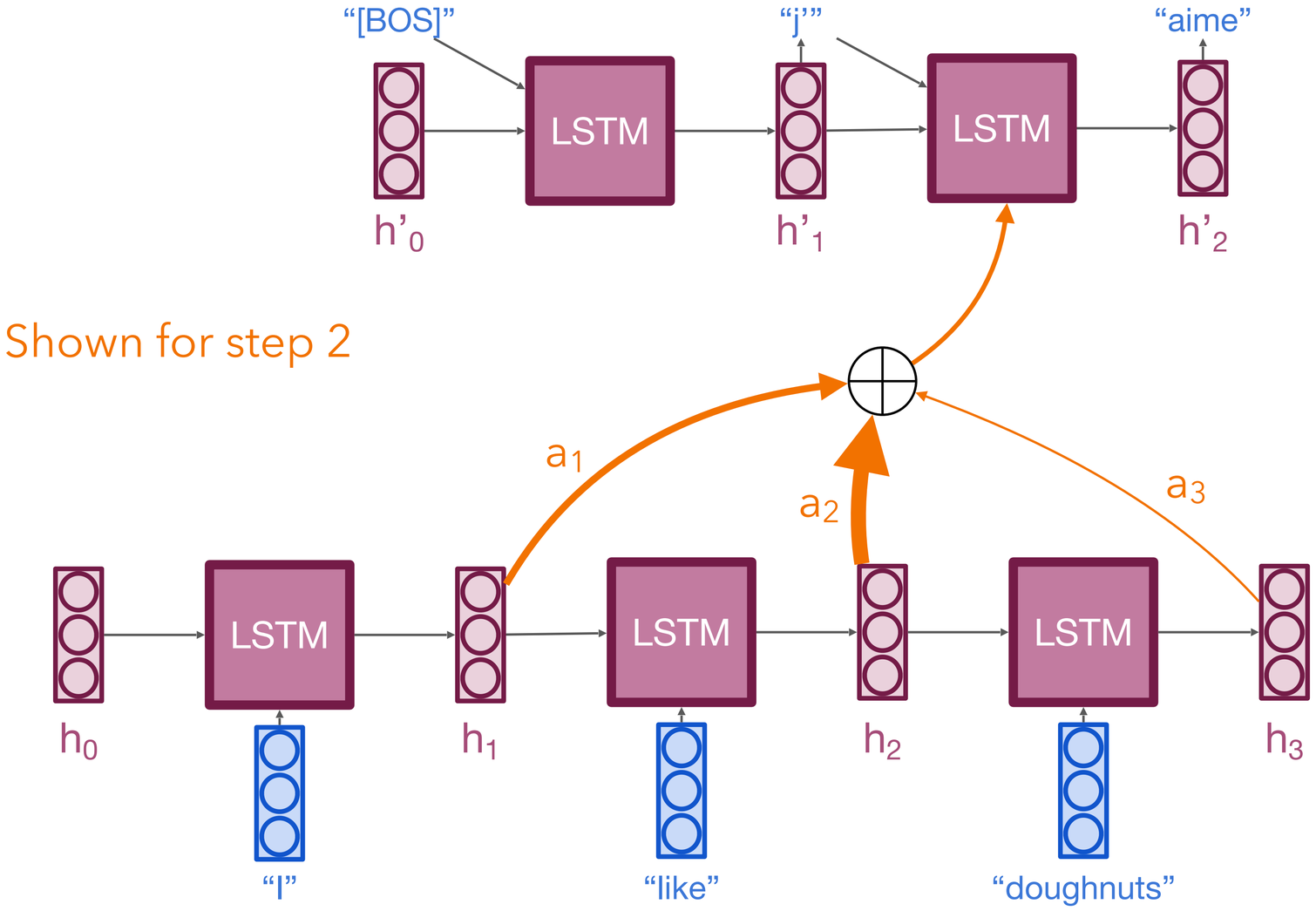
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



Att.

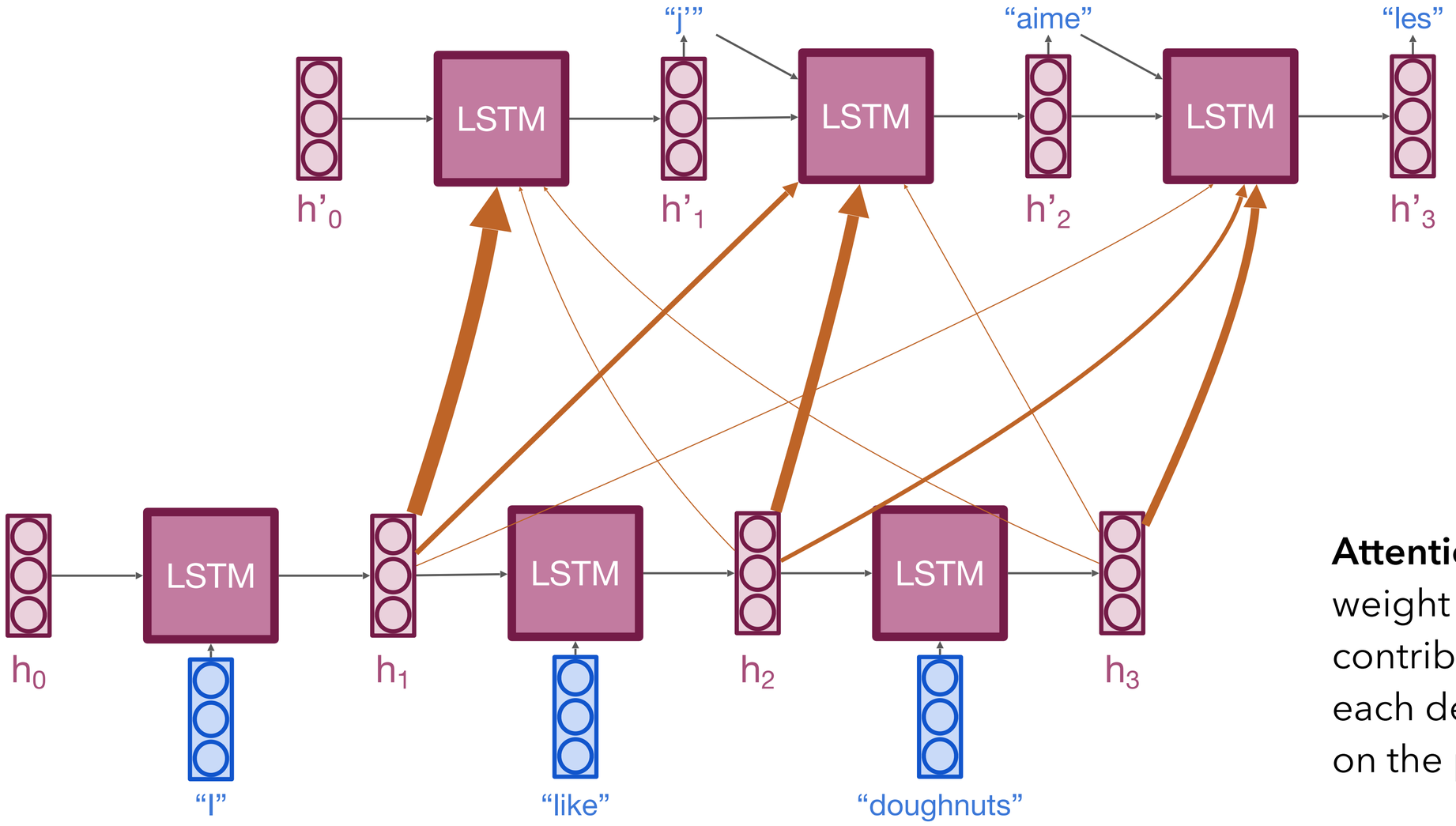
Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Seq2seq RNN with attention

(Bahdanau *et al.* 2014)

RNN

seq2seq



Attention mechanism: the weight of each encoder step's contribution to the input of each decoder step depends on the previous decoder step

Transformers

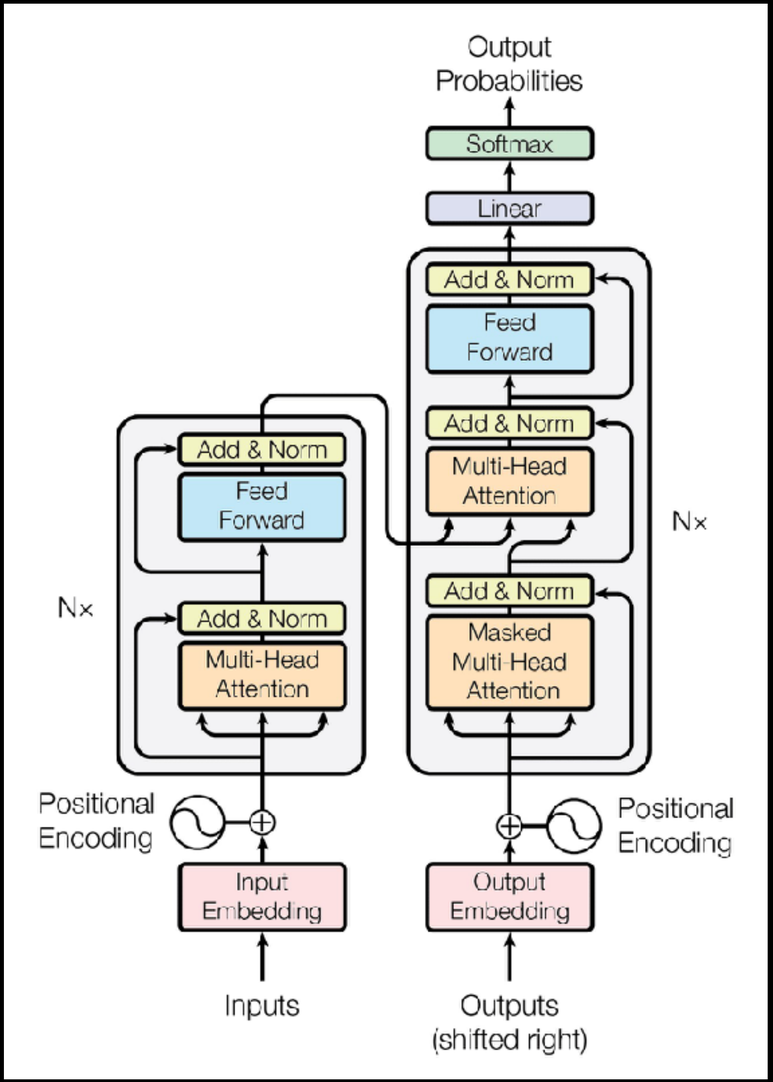


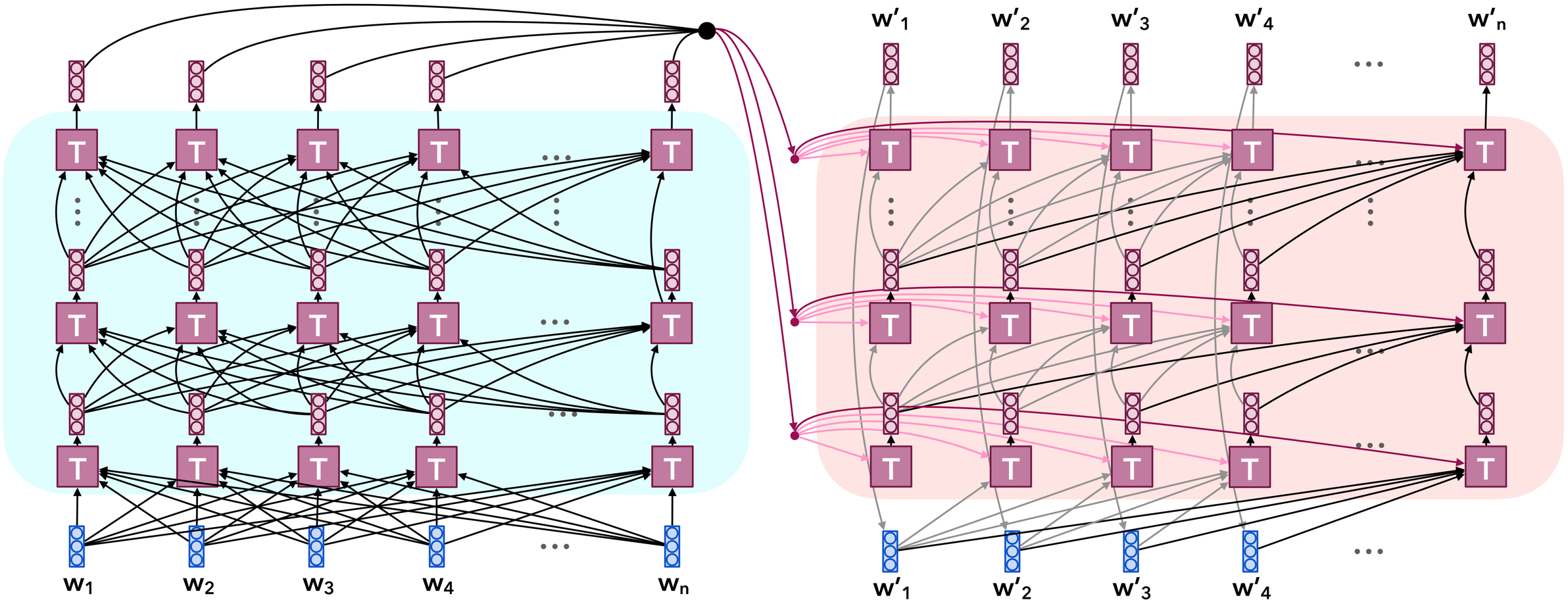
Figure from (Vaswani et al. 2017)

The Transformer architecture

(Vaswani *et al.* 2017)

Transformer

seq2seq



Encoder

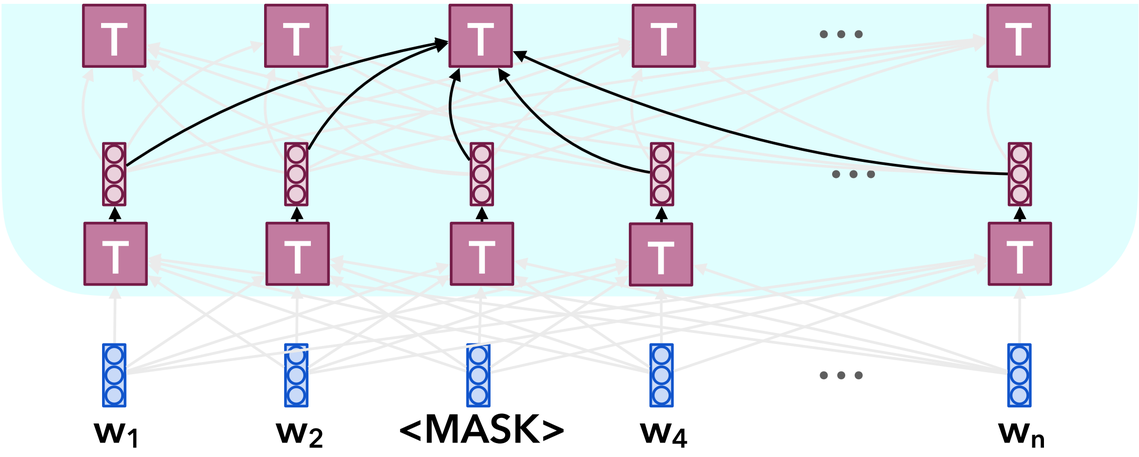
Decoder

Transformer encoder and self-attention

Transformer

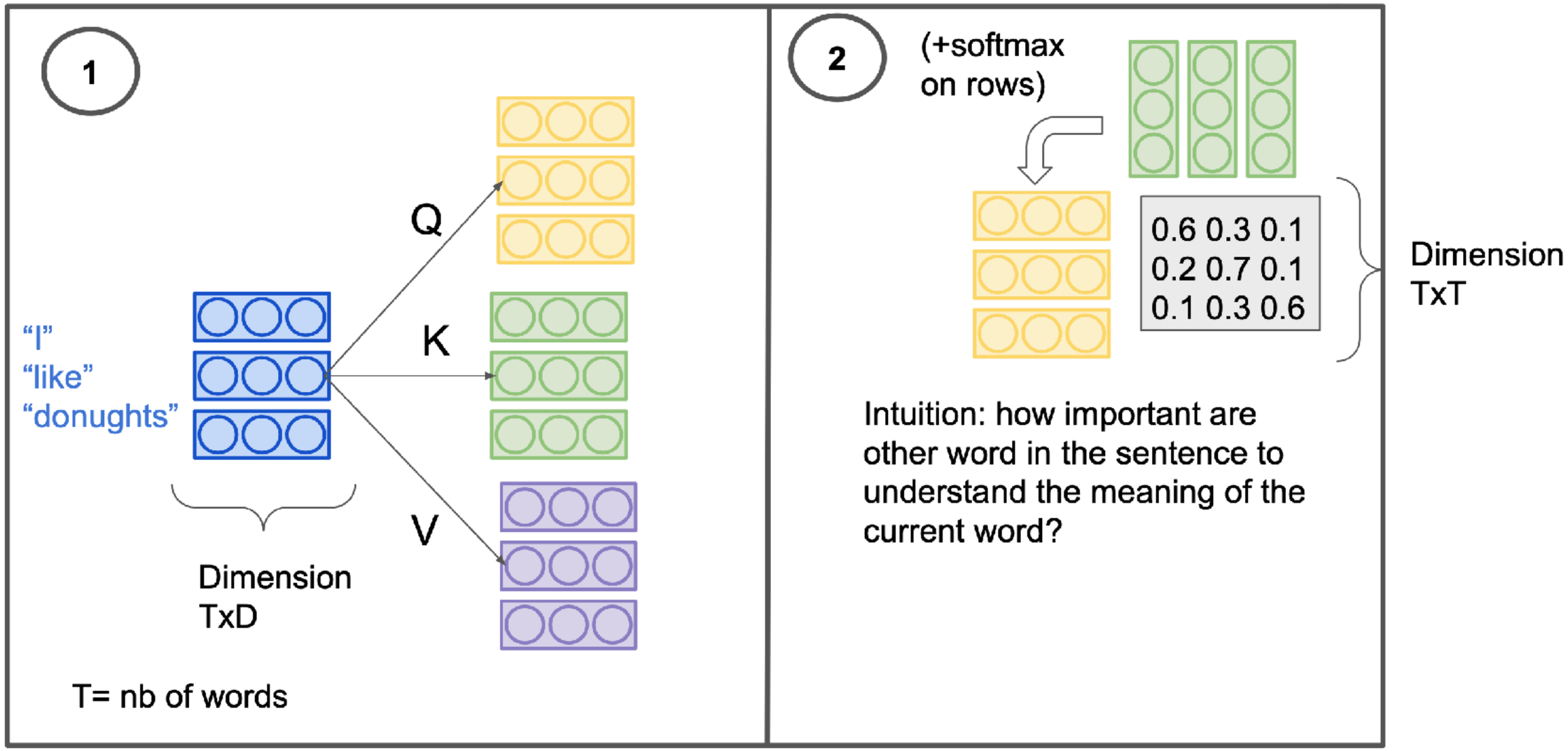
MLM

- Two main changes with respect to RNNs
  - The representation of a token by the layer *k* relies on the representation of *all* tokens by the layer *k*-1, using an attention mechanism
    - It is called the **self-attention** mechanism
  - No horizontal dependencies within layers (no sequential recurrence)



Self-attention

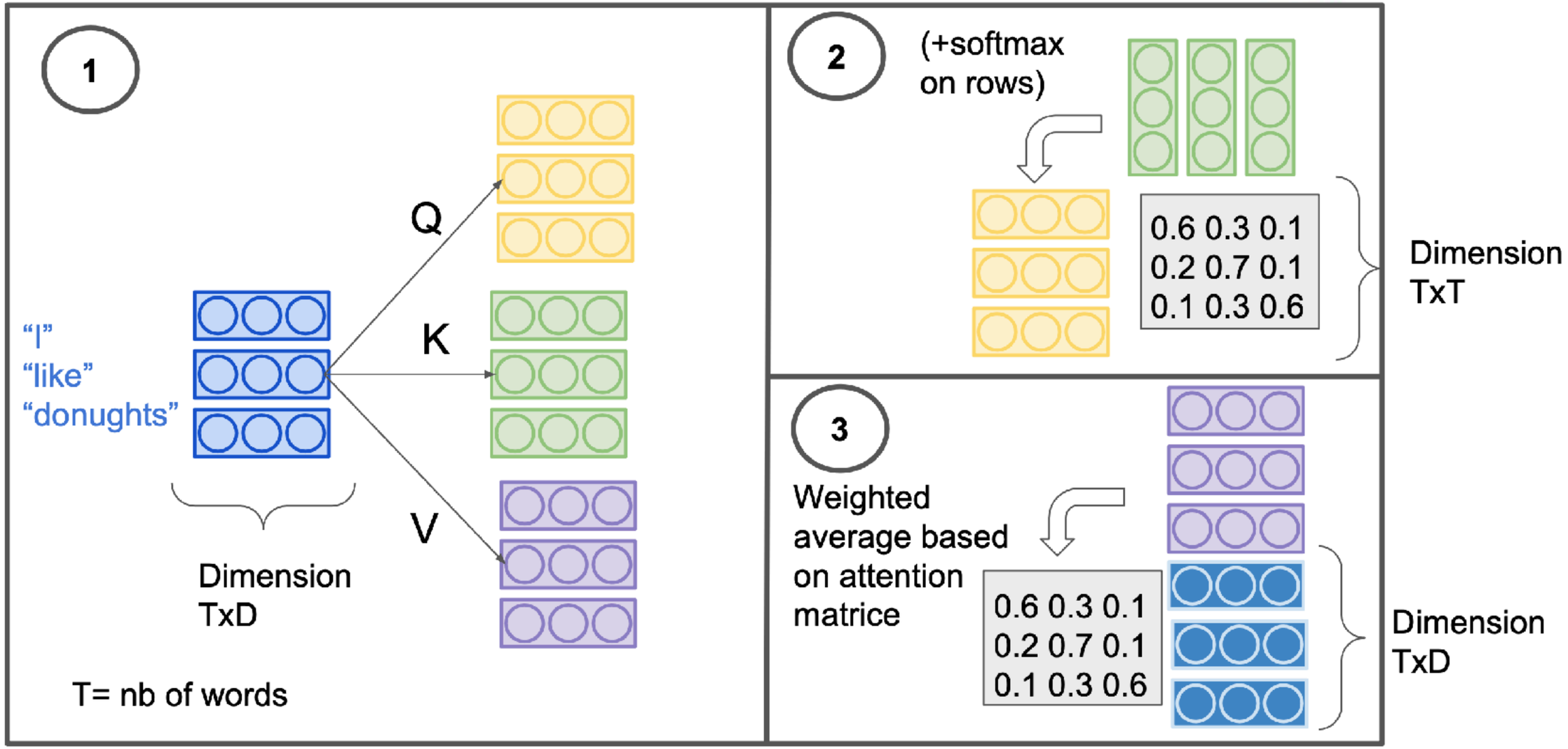
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



*Vaswani et al. Attention is all you need.

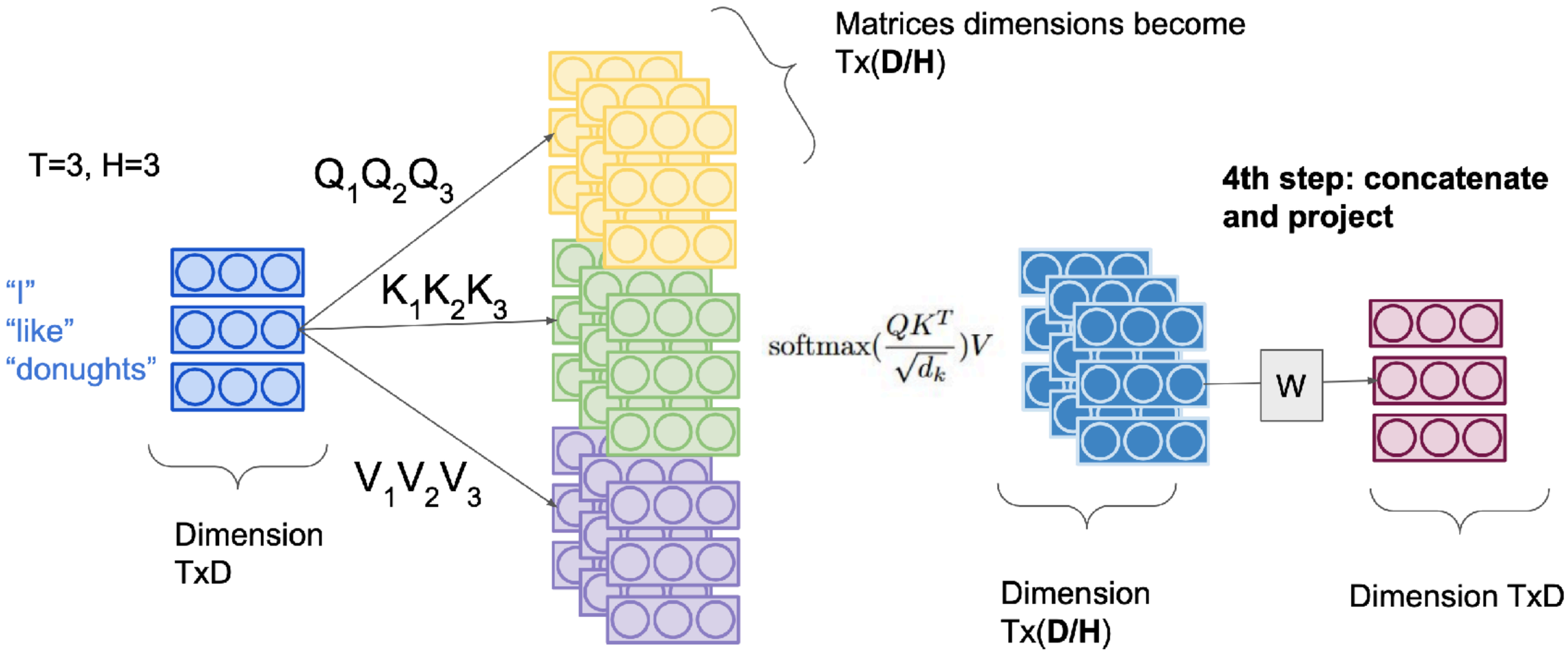
Self-attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



*Vaswani et al. Attention is all you need.

Multi-head self-attention

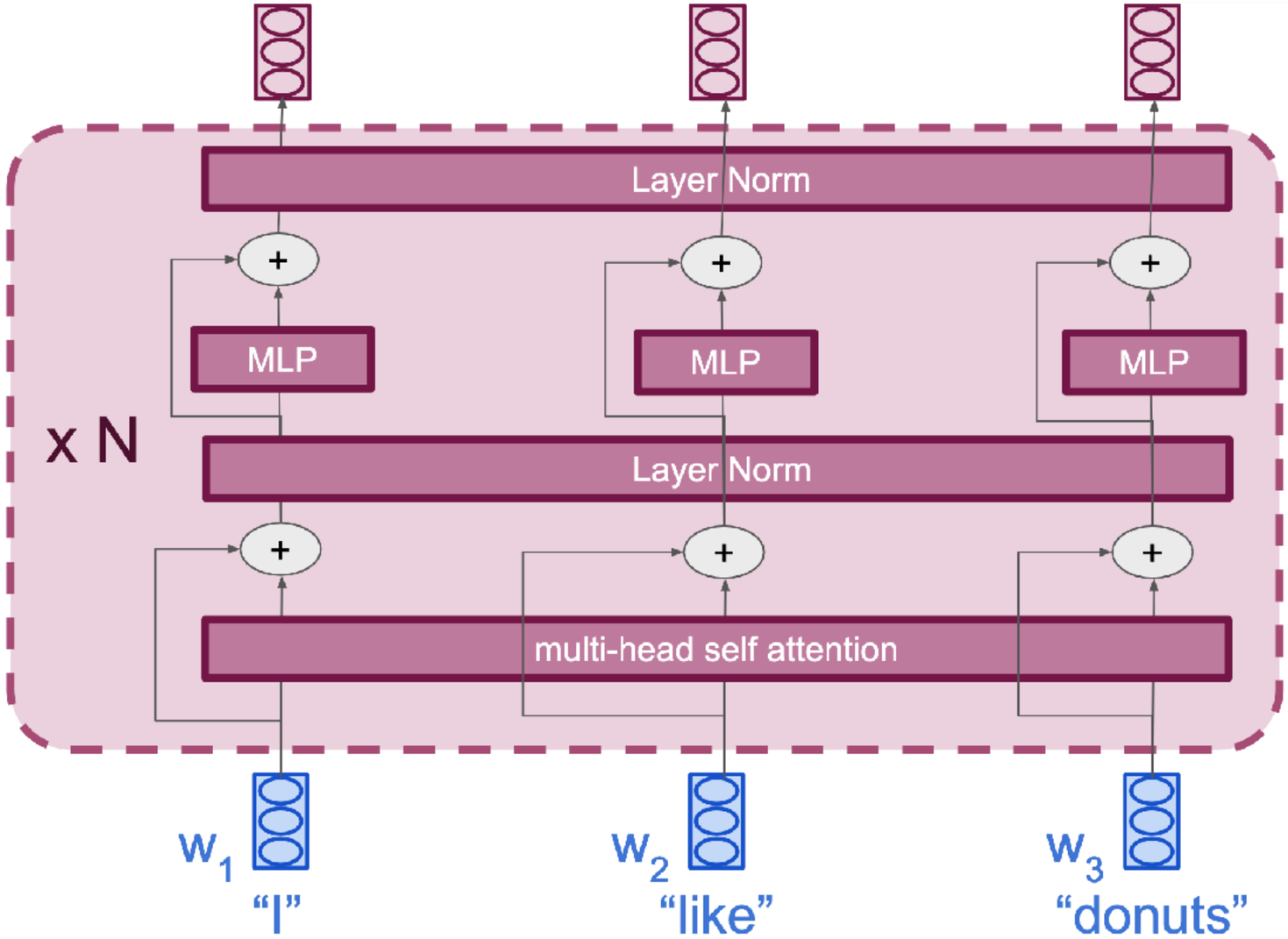


D must be a multiple of H !

Transformer encoder

Transformer

MLM

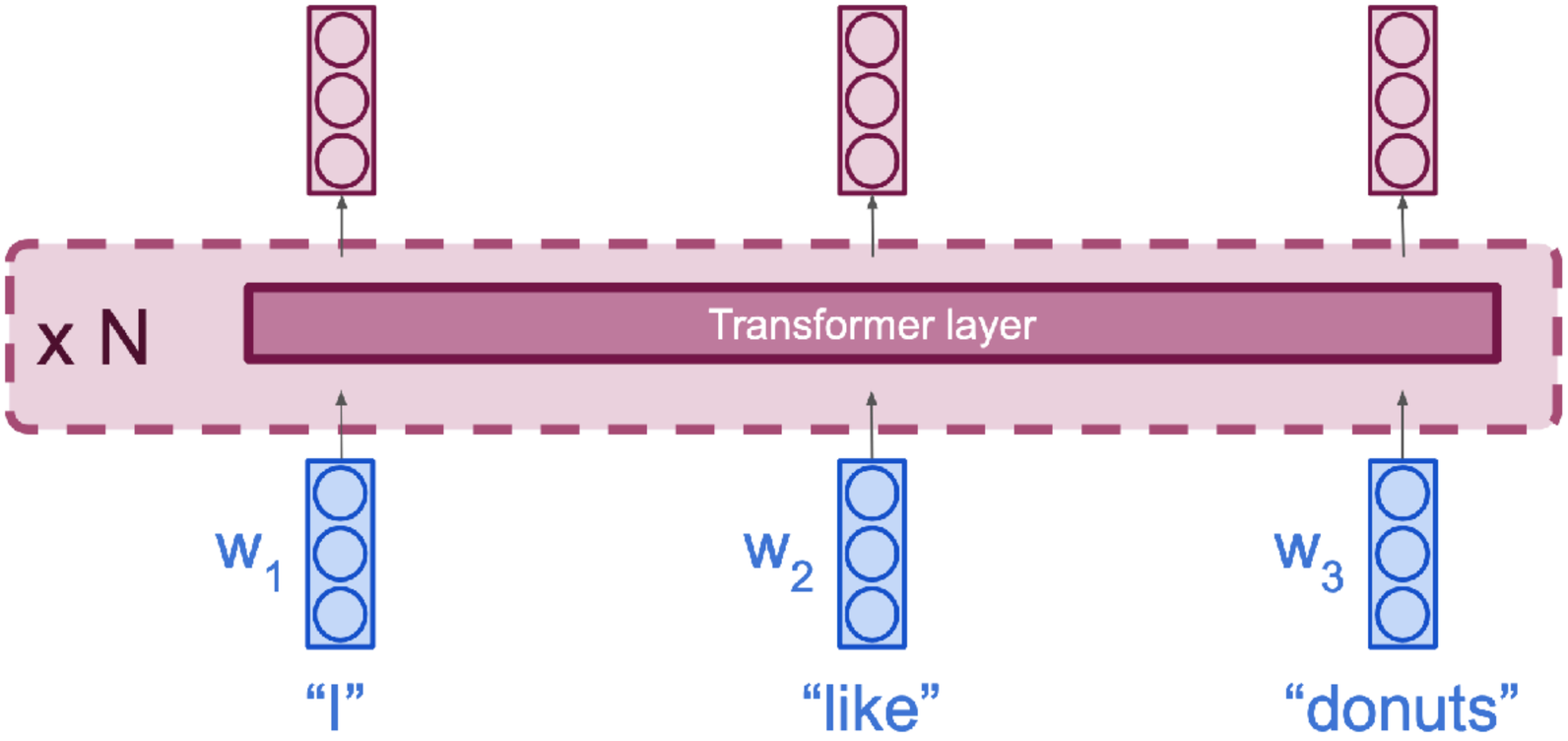


Residual connection: prevent vanishing gradient. Necessary in all deep architectures.

How is word order encoded?

Transformer

MLM

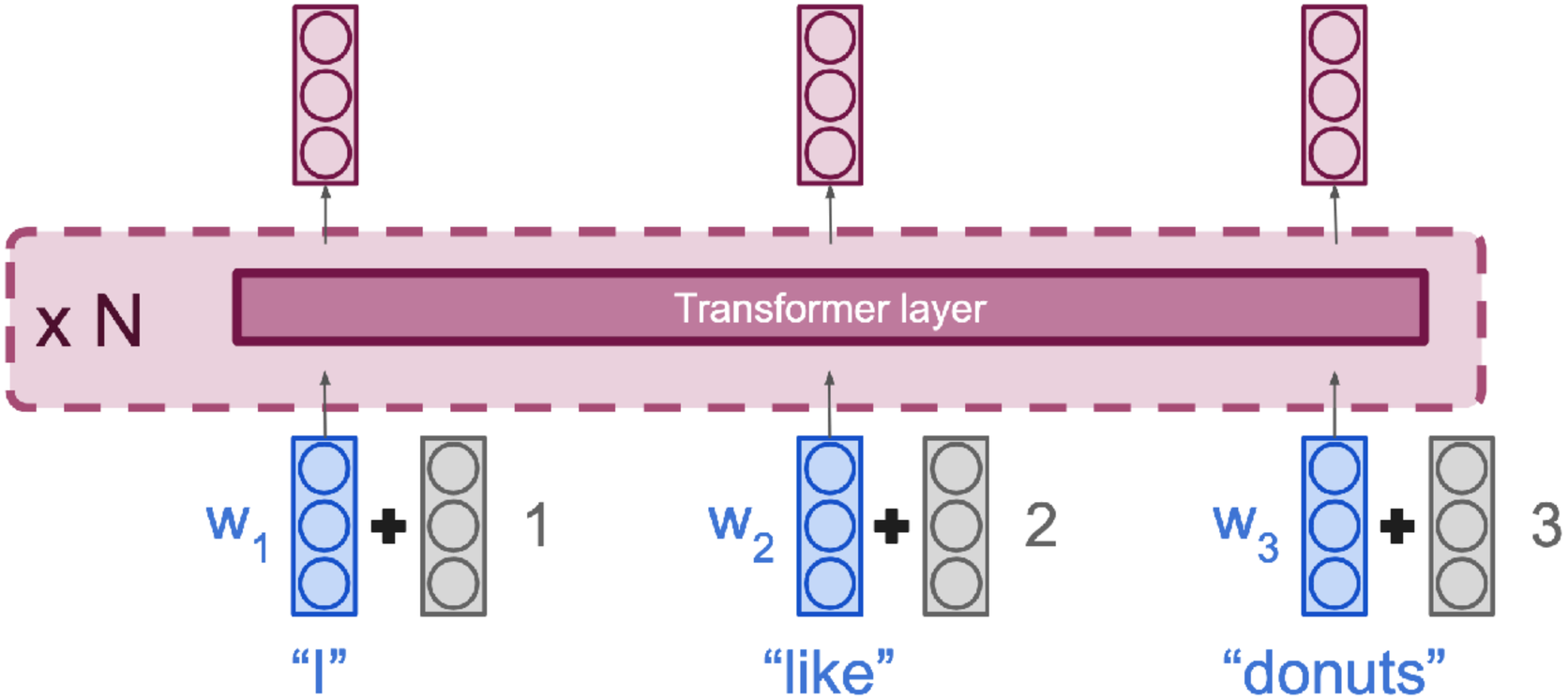


How is word order encoded?

Transformer

MLM

- Positional embeddings



Positional embedding

Transformer

MLM

- Unique encoding for each position
- Distance between two positions should be consistent across sentences with different lengths (cannot use a 1-hot)
- Its values should be bounded (the position itself cannot be used)

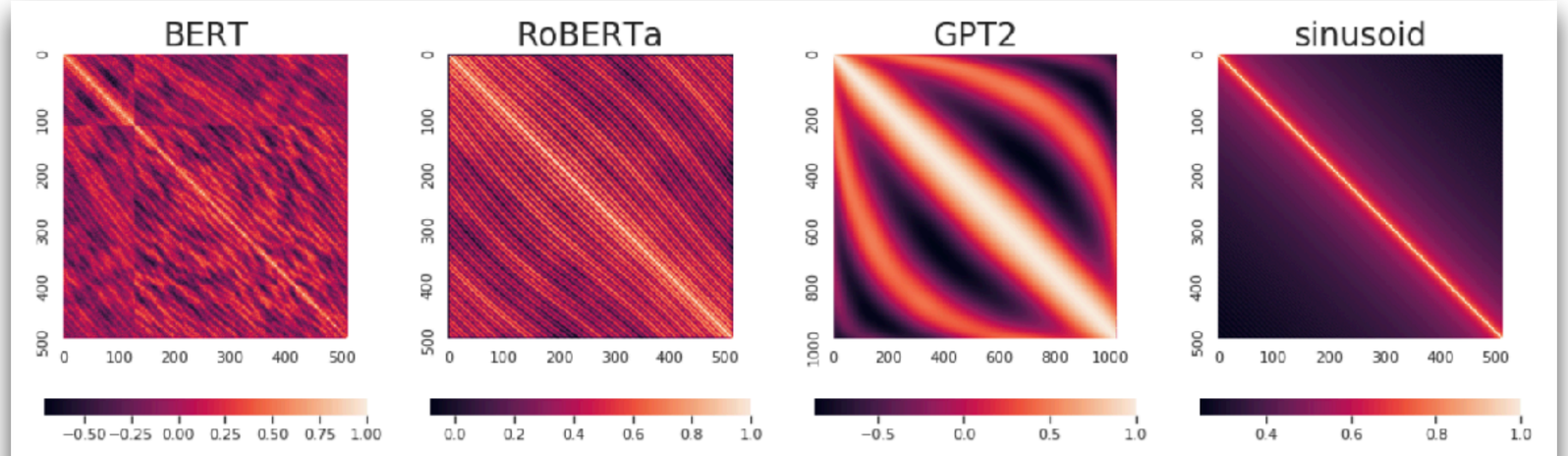


Figure 1: Visualization of position-wise cosine similarity of different position embeddings. Lighter in the figures denotes the higher similarity.

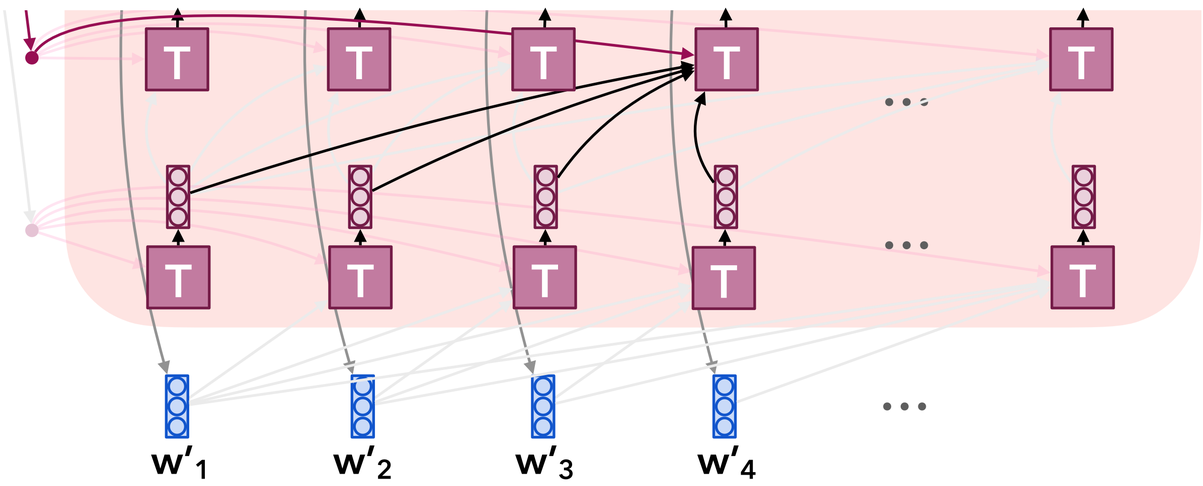
Figure from (Wang et al. 2020)

Transformer decoder

Transformer

generative

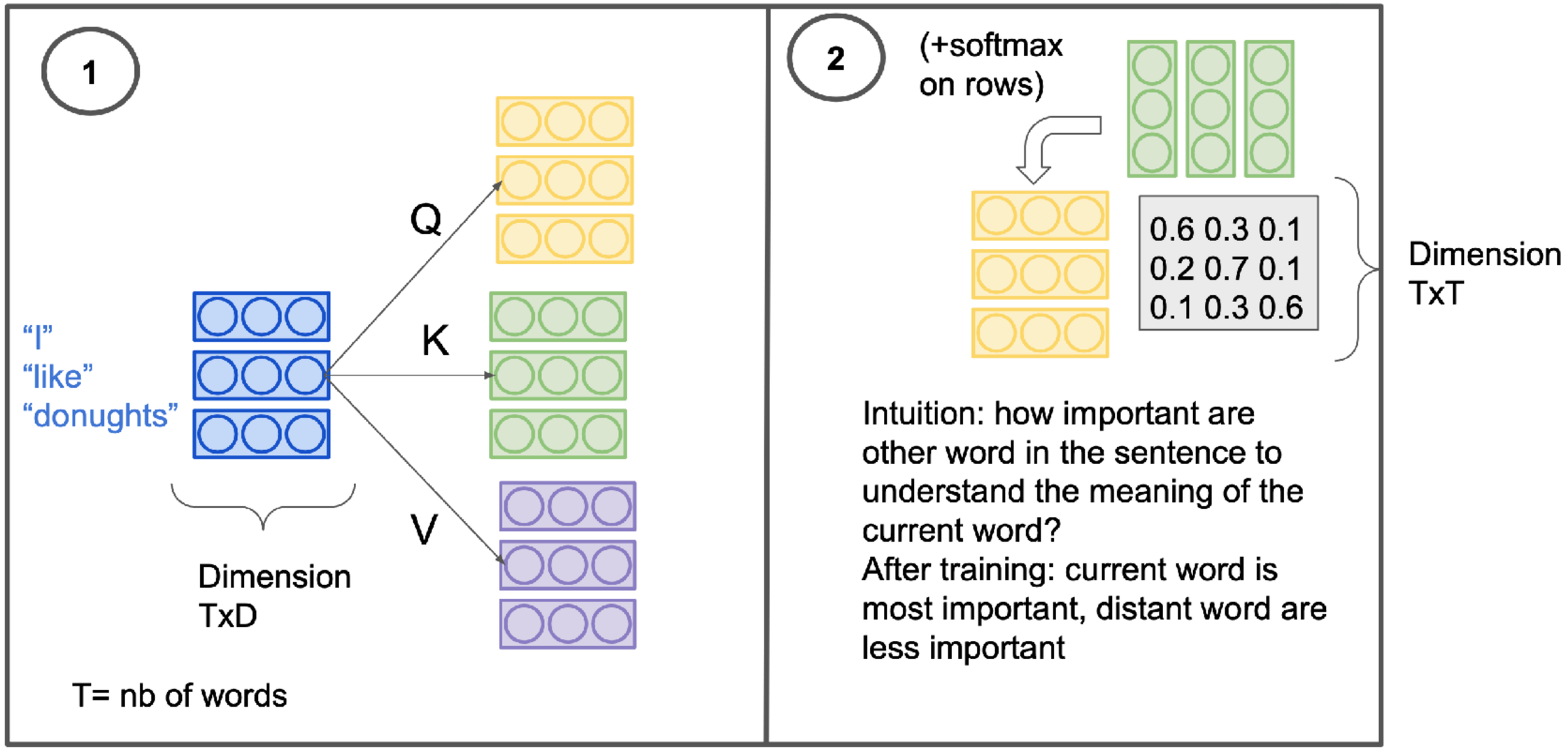
- Same architecture as encoder, with one small yet crucial change: the causal mask on the attention matrices
  - Multi-head self-attention must now only look at past and present tokens, not at future tokens
- If the full architecture is used, cross-attention (to the encoder's outputs) is also used



How to make a transformer causal?

Transformer

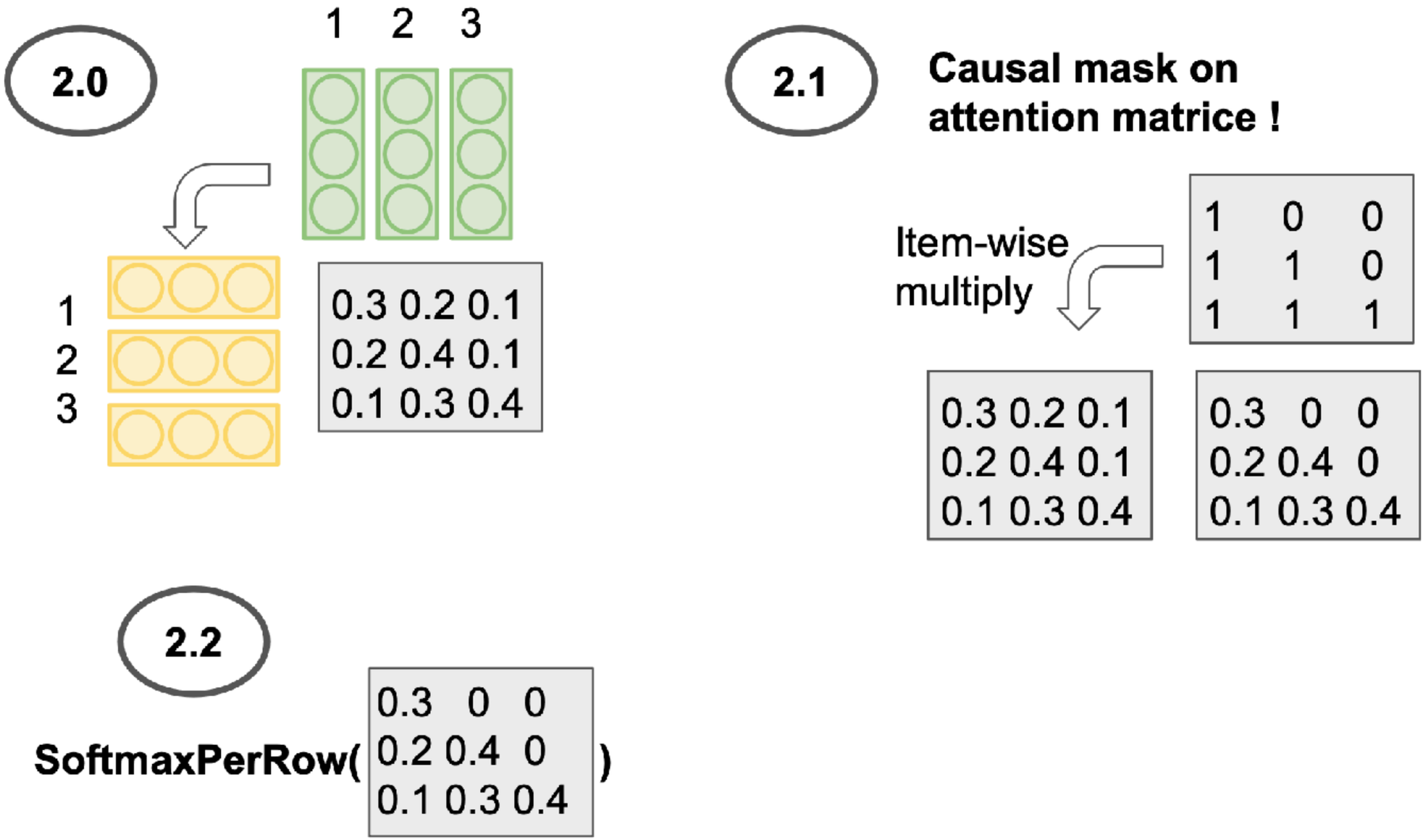
generative



How to make a transformer causal?

Transformer

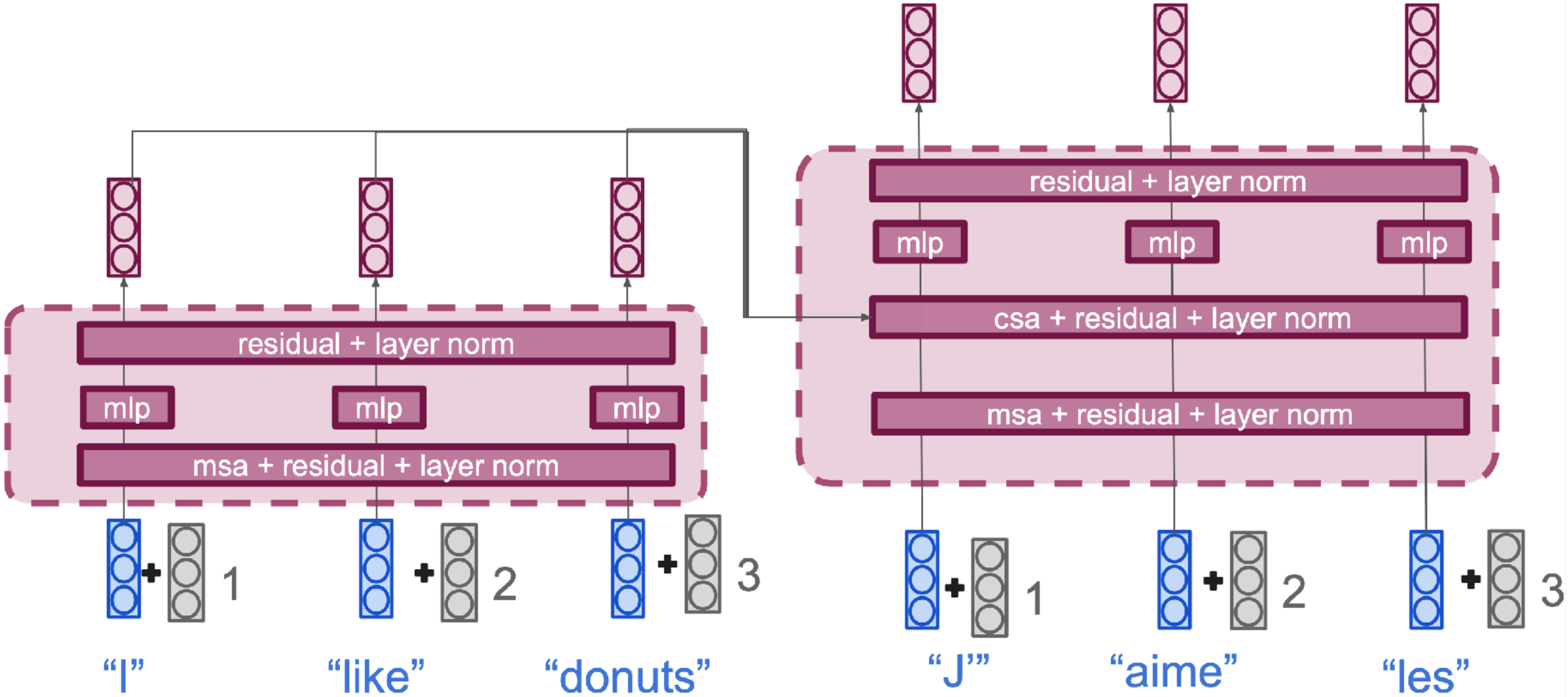
generative



The full Transformer architecture

Transformer

seq2seq



Advantages of Transformers

Transformer

generative

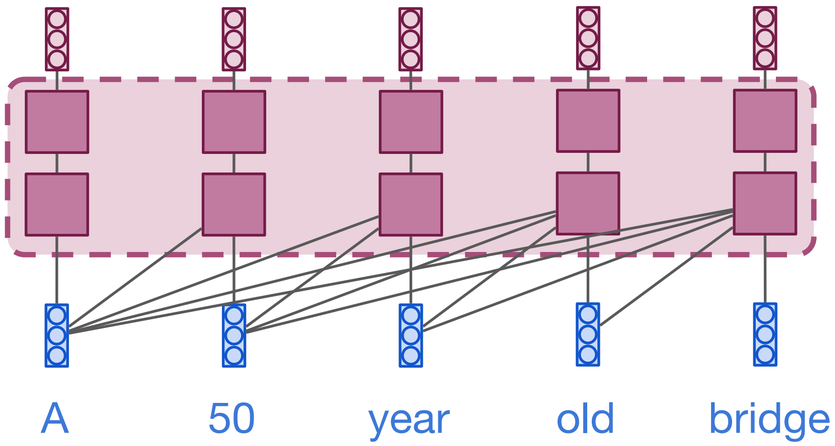
- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism

Advantages of Transformers

Transformer

generative

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism

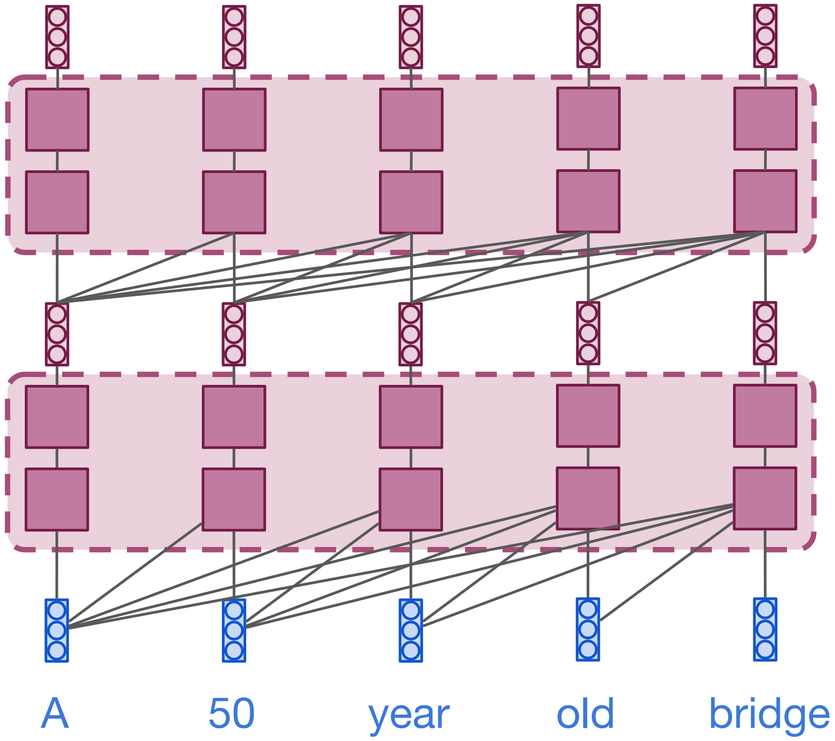


Advantages of Transformers

Transformer

generative

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism

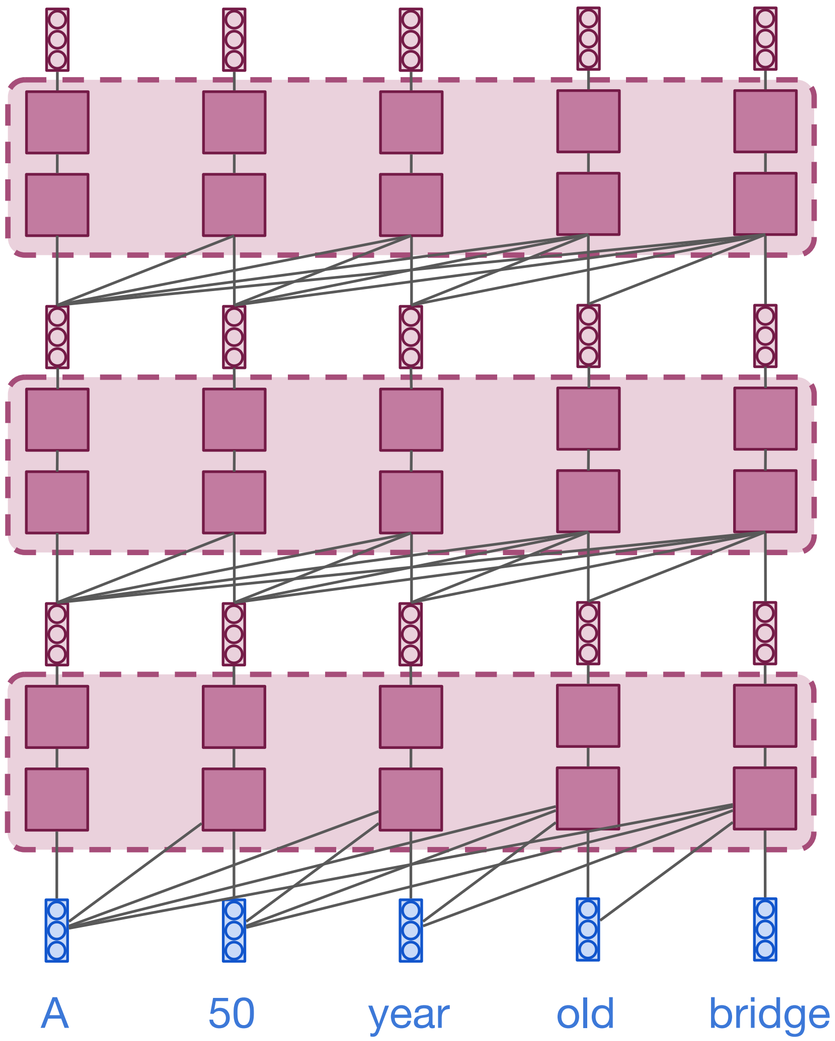


Advantages of Transformers

Transformer

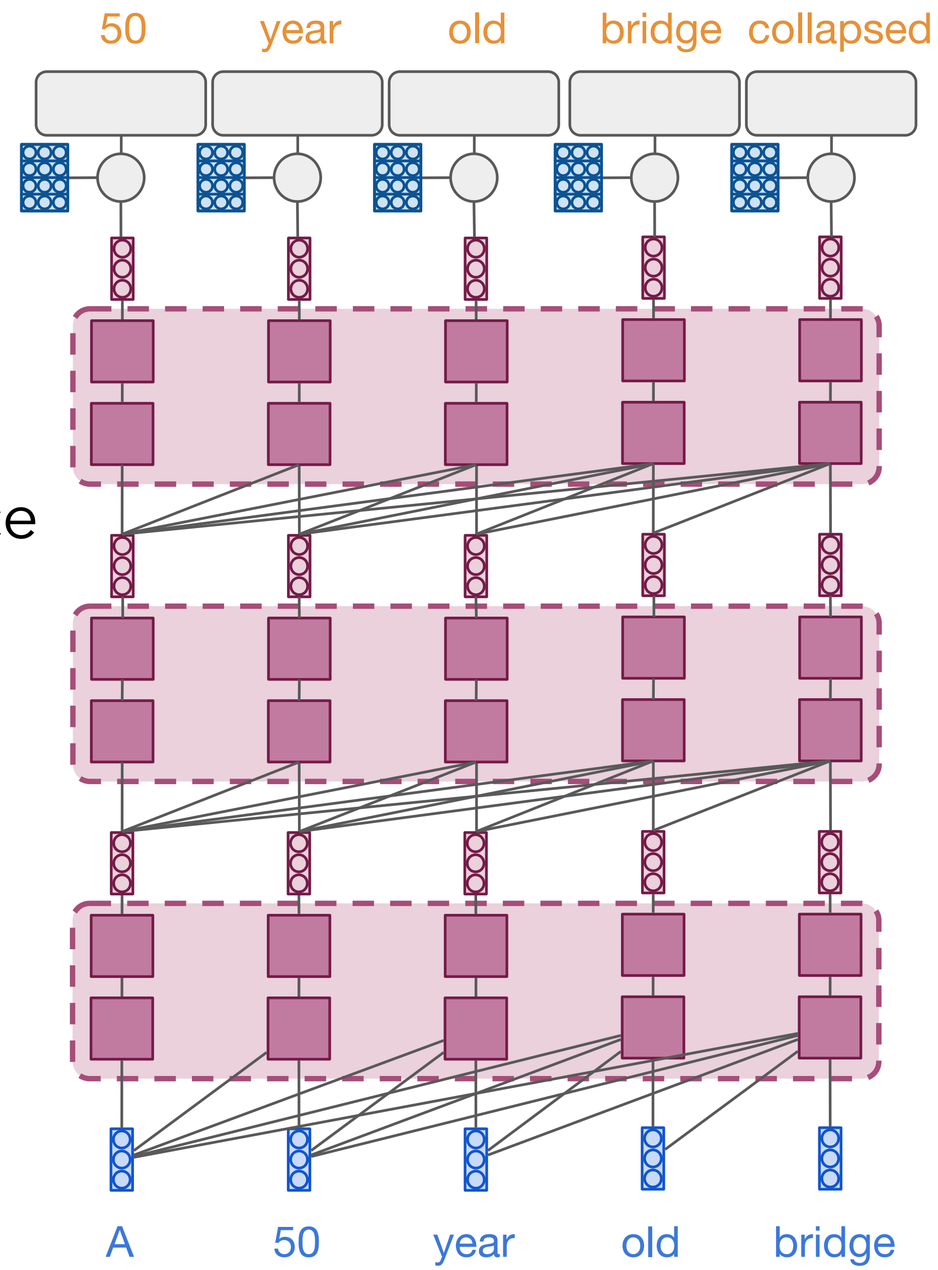
generative

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism



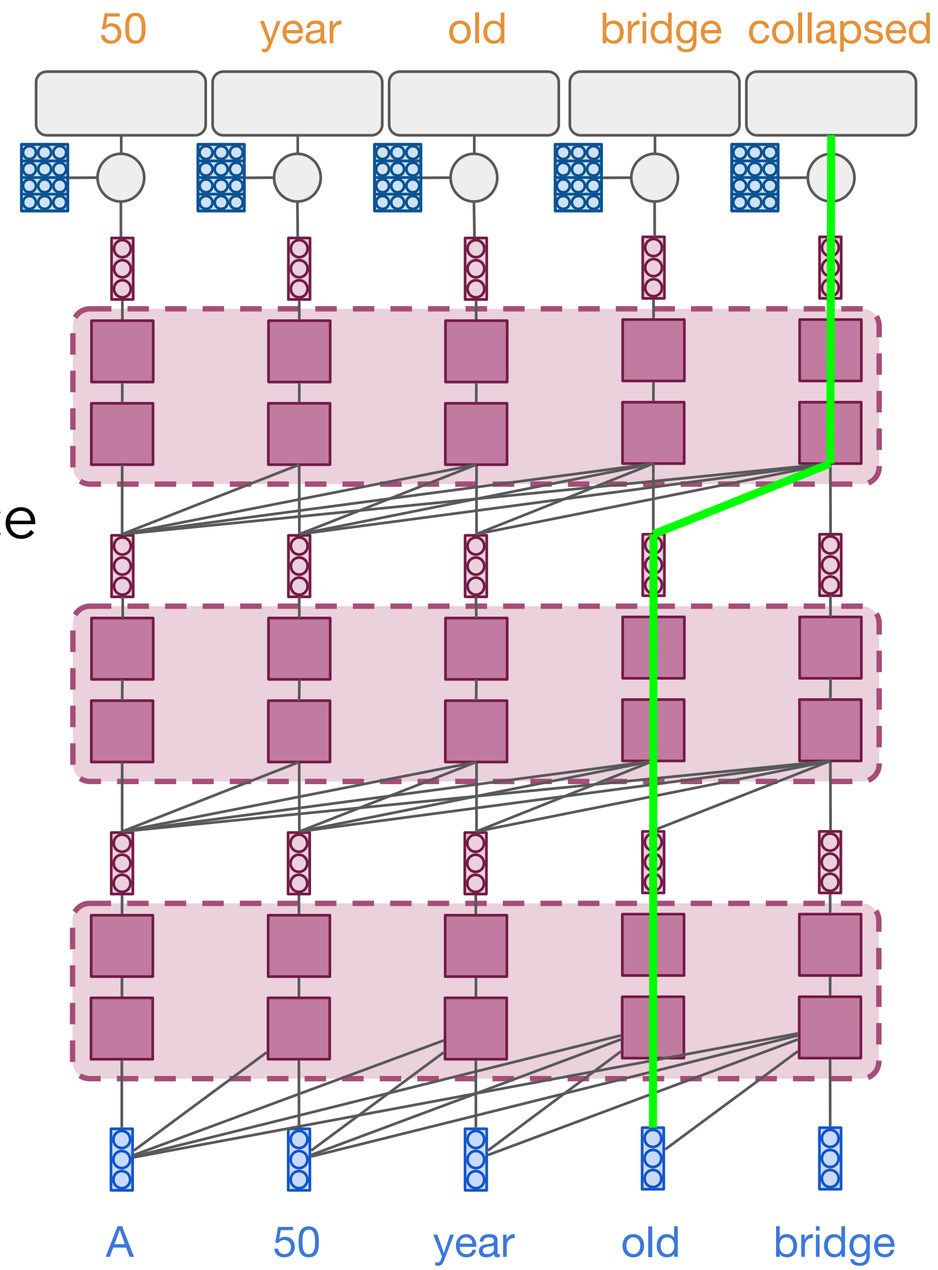
Advantages of Transformers

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism



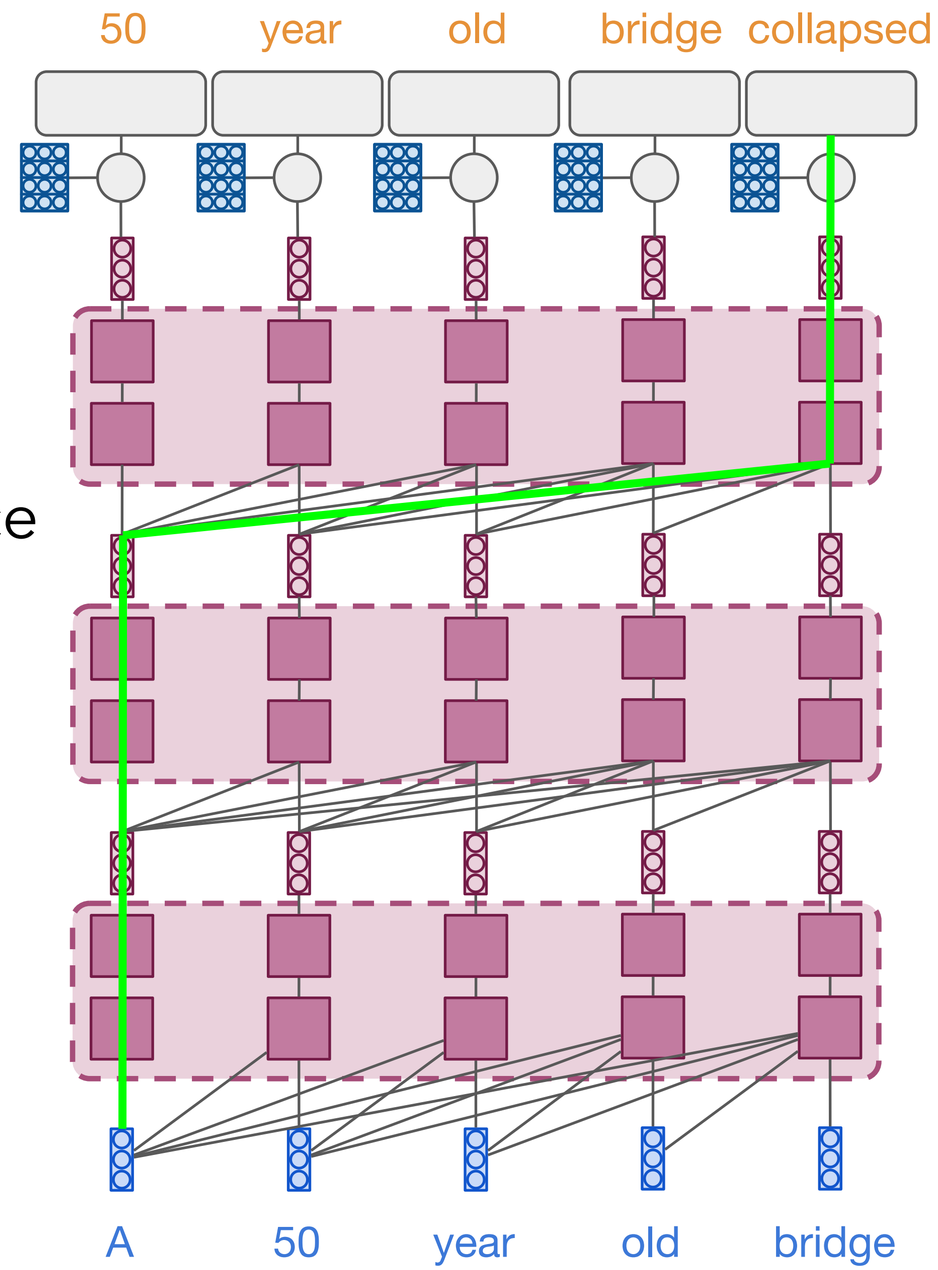
Advantages of Transformers

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism
- **Easier to attend to distant context**
  - Direct path from one position to the other



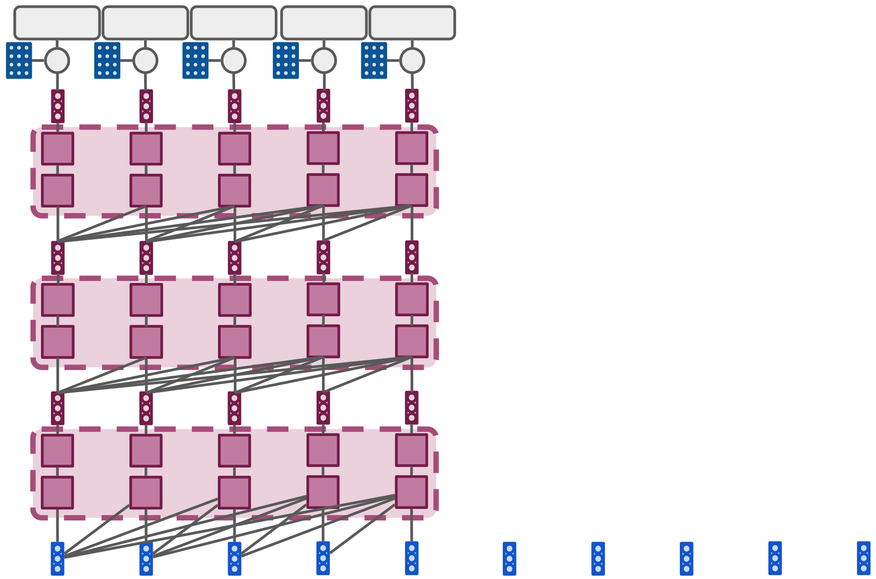
Advantages of Transformers

- **Easy parallelism**
  - Entire sequences can be processed at once
  - Well suited to GPU/TPU parallelism
- **Easier to attend to distant context**
  - Direct path from one position to the other



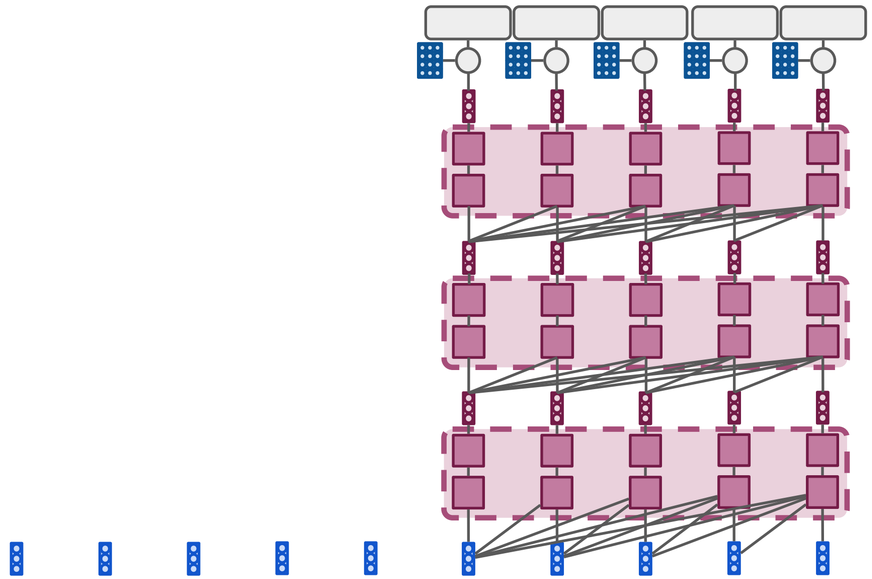
Downsides of Transformers

- **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?
  - In practice, the context window is now very large!

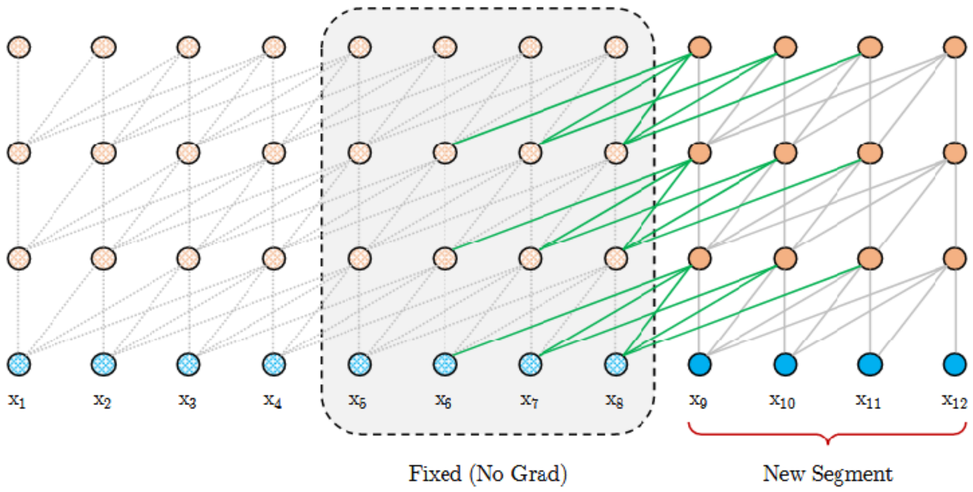


Figure from (Dai et al. 2019)

Downsides of Transformers

- **Fixed context window**
  - Back to the Markov assumption?
  - In practice, the context window is now very large!

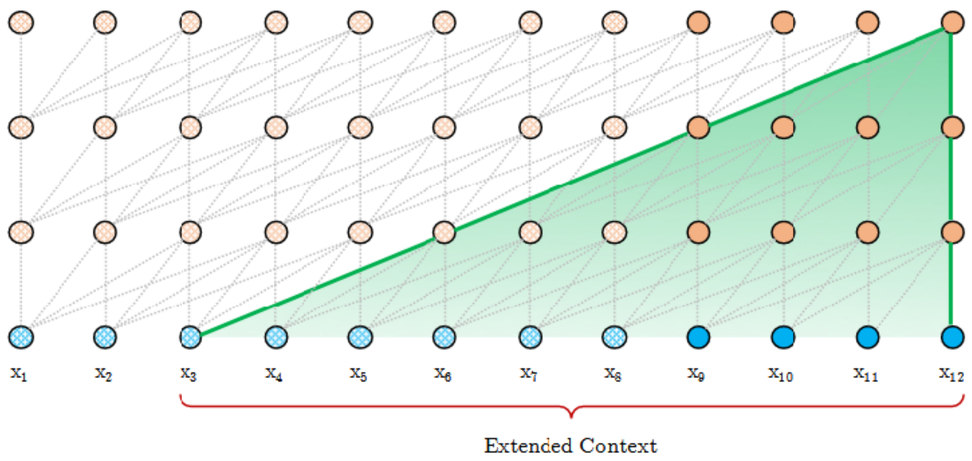
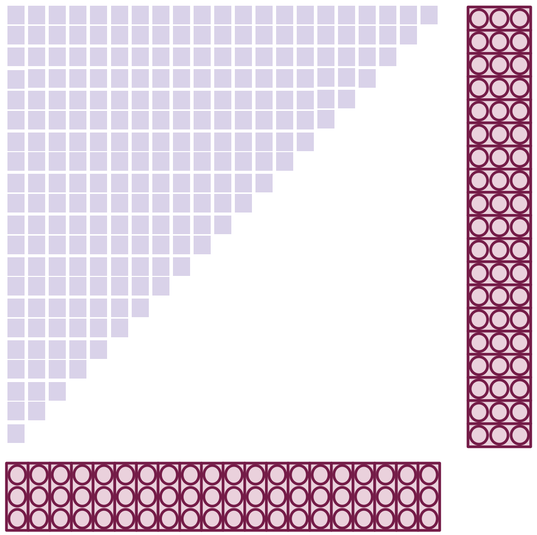


Figure from (Dai et al. 2019)

Downsides of Transformers

- **Fixed context window**
  - Back to the Markov assumption?
  - In practice, the context window is now very large!
- **Compute-intensive for long sequences**
  - Self-attention is quadratic in sequence length



Downsides of Transformers

- **Fixed context window**
  - Back to the Markov assumption?
  - In practice, the context window is now very large!
- **Compute-intensive for long sequences**
  - Self-attention is quadratic in sequence length
  - Lots of work on making attention more efficient

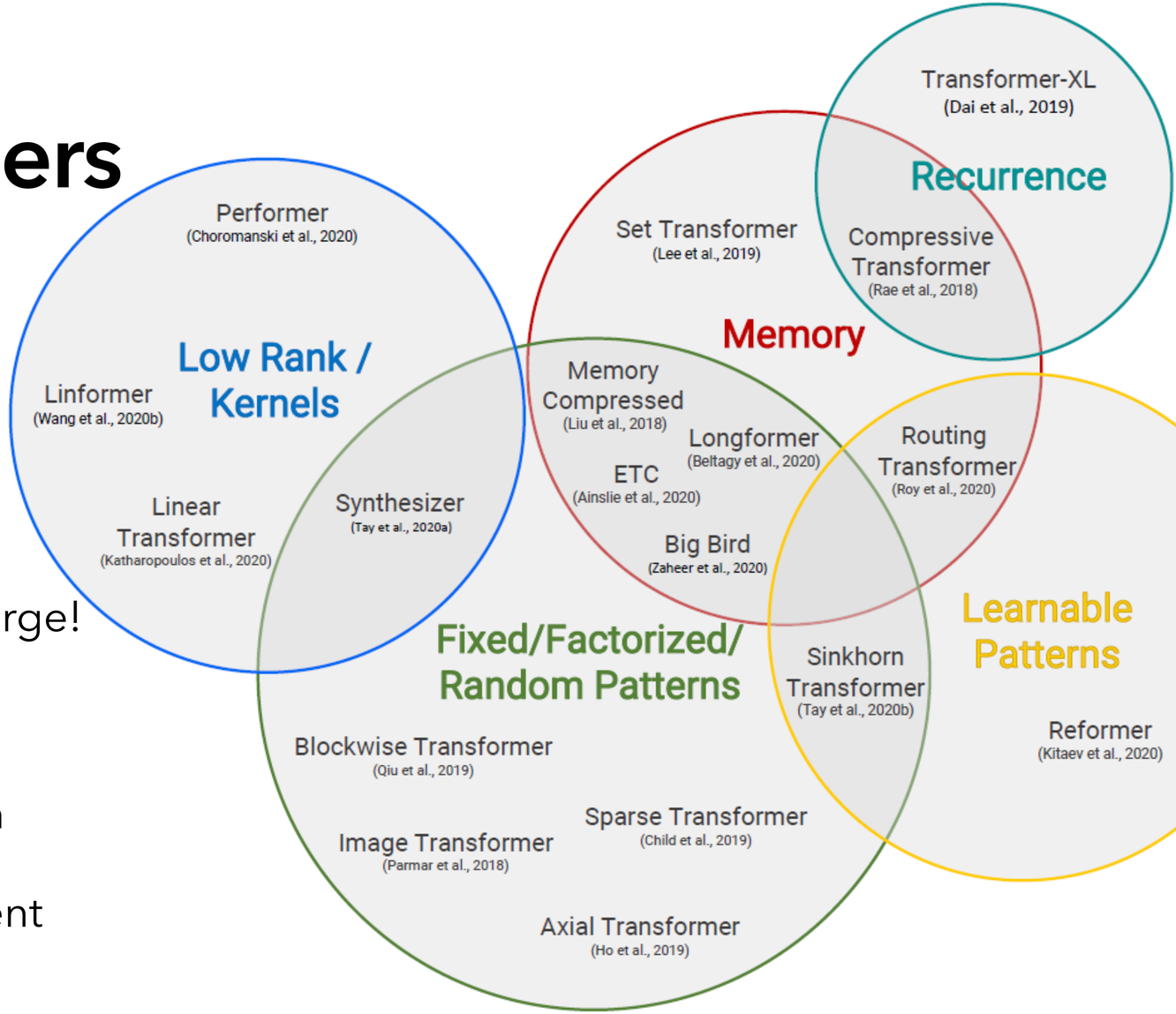


Figure from (Tay et al. 2020)

Downsides of Transformers

- **Fixed context window**
  - Back to the Markov assumption?
  - In practice, the context window is now very large!
- **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**

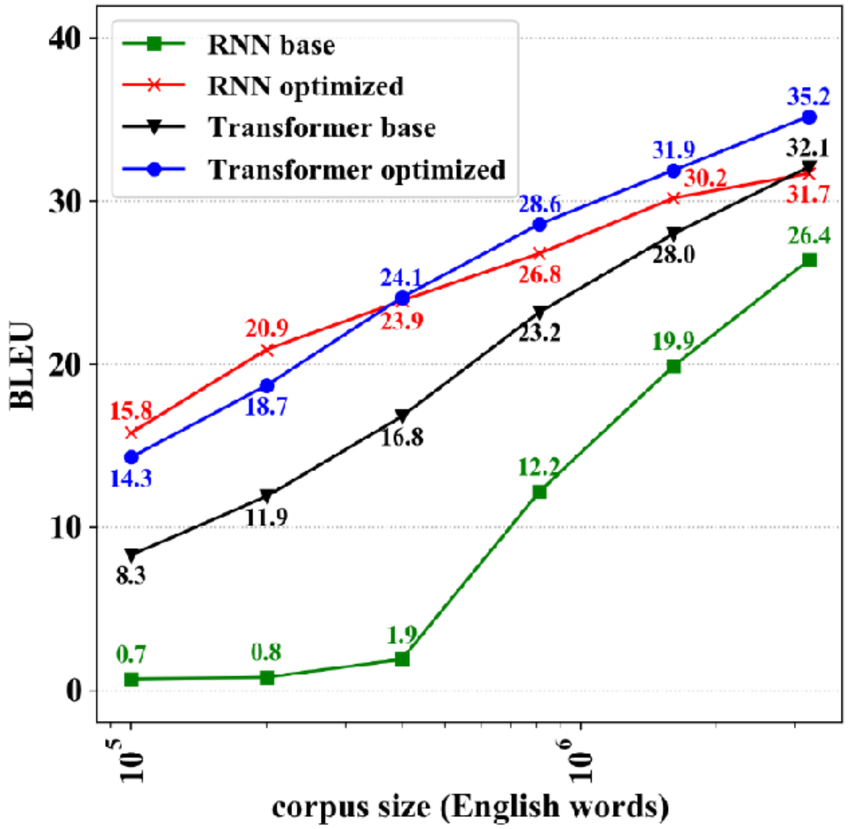
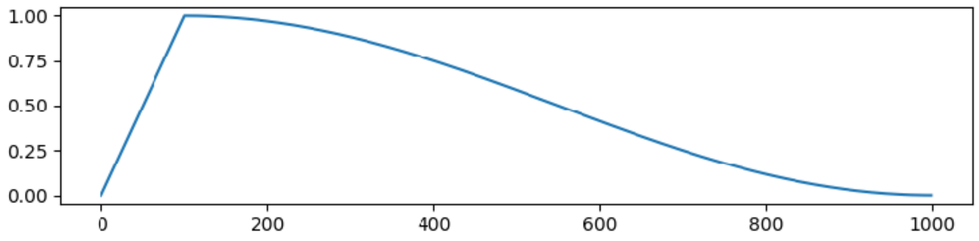


Figure from (Tay et al. 2020)

Downsides of Transformers

- **Fixed context window**
  - Back to the Markov assumption?
  - In practice, the context window is now very large!
- **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 optimiser, learning rate schedule…

Cosine learning rate schedule with warmup



Transformer LM results on WikiText103

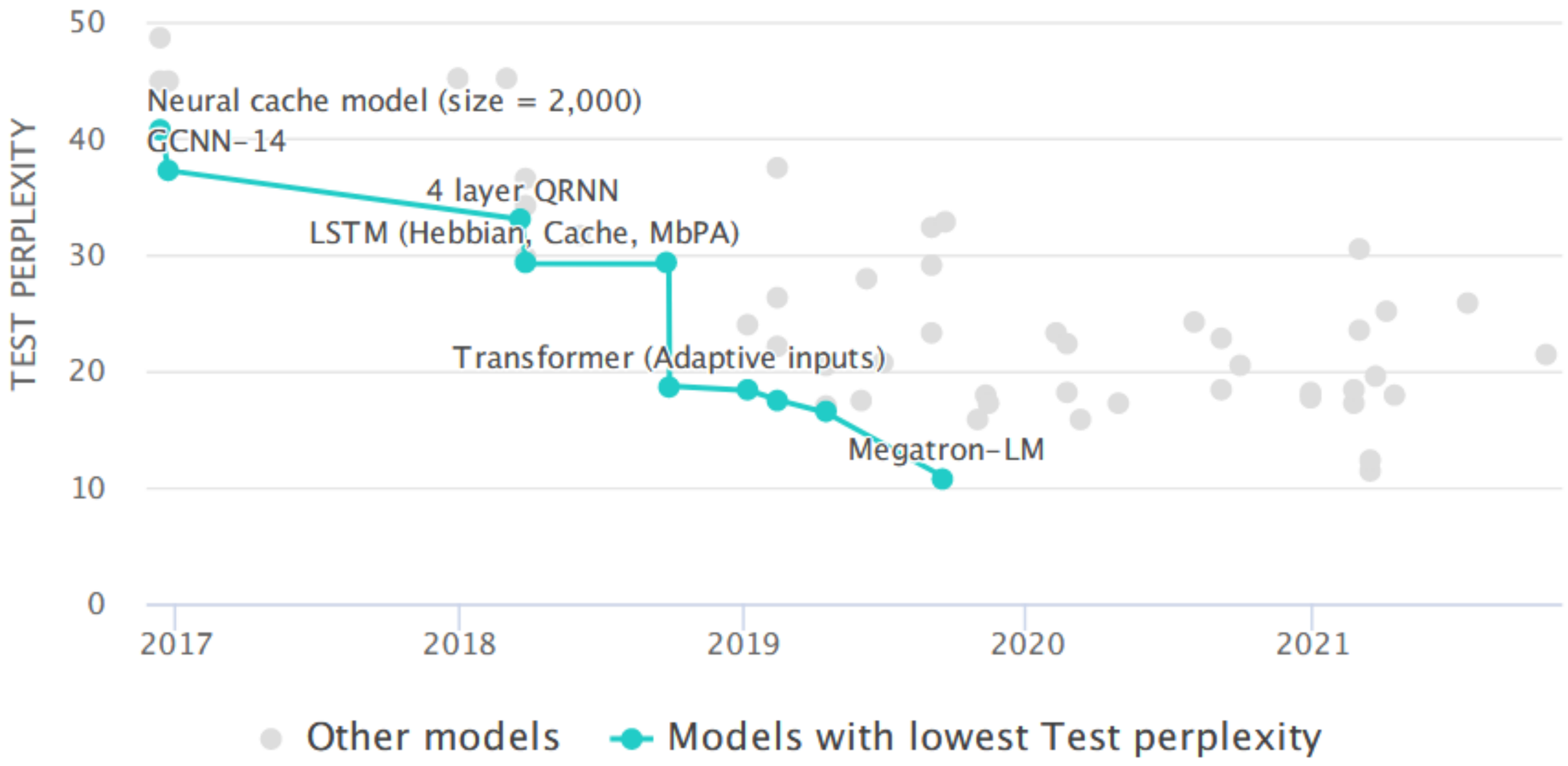
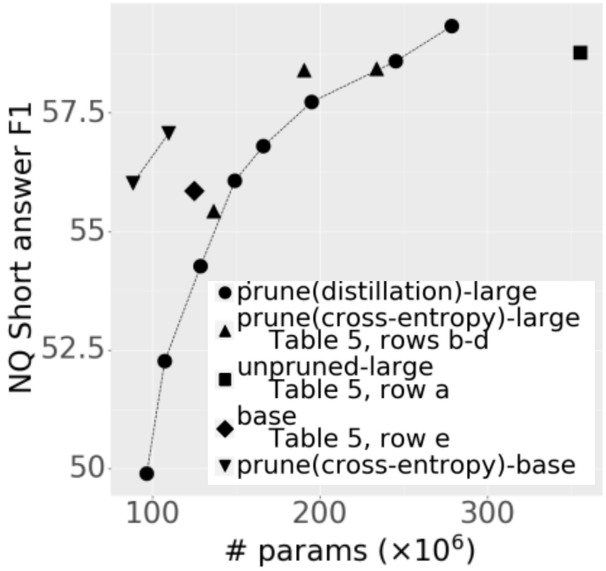


Figure from https://paperswithcode.com/sota/language-modelling-on-wikitext-103

More Efficient Models: Pruning

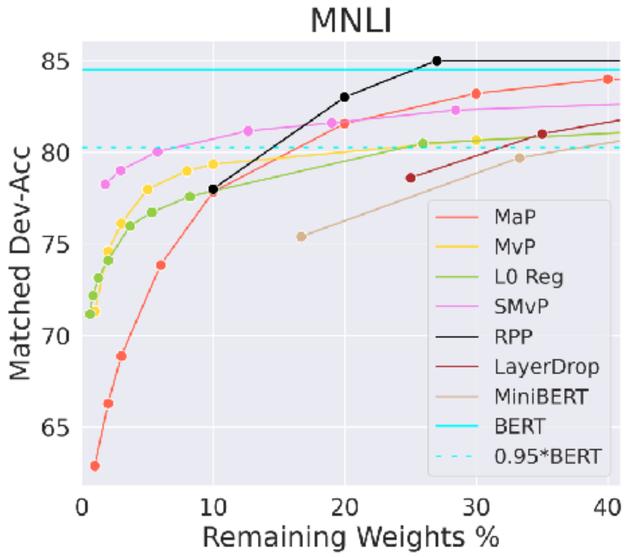
A significant number of parameters can be **removed after training**

- Structured Pruning
  - Remove dimensions, attention heads, etc.
  - Faster models
- Unstructured Pruning
  - Remove individual parameters
    - e.g. parameters close to 0
  - Not necessarily faster
  - Allows for a greater reduction in parameters



Structured pruning of RoBERTa on question answering

Figure from (McCarley *et al.* 2020)



Unstructured pruning of BERT (NLI)

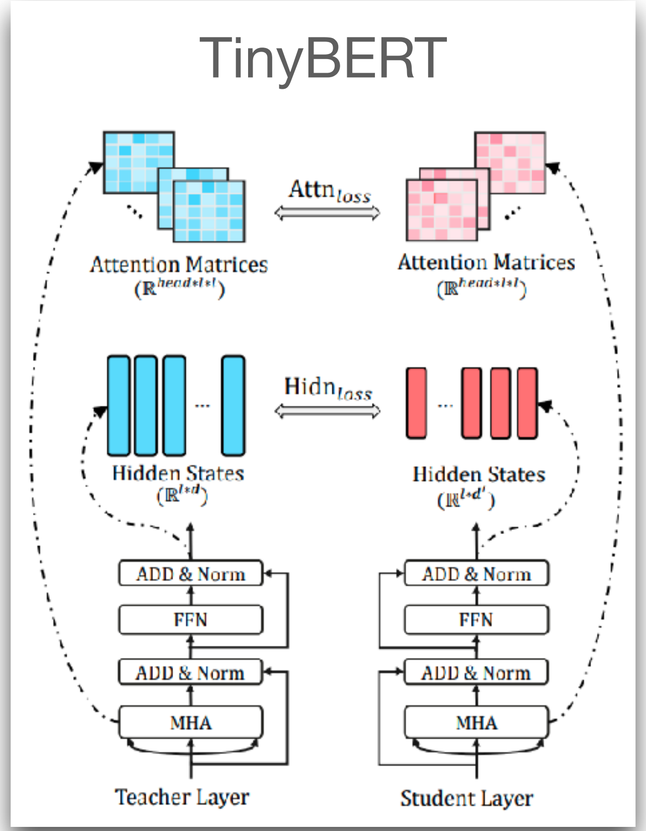
Figure from (Sanh *et al.* 2020)

More Efficient Models: Distillation

Train a **smaller** model to imitate a big model

- Better performance than small model trained from scratch
- Smaller model can be fine-tuned

		Compression	Performance	Speedup	Model	Evaluation
	BERT-base (Devlin et al., 2019)	×1	100%	×1	$BERT_{12}$	All GLUE tasks, SQuAD
	BERT-small	×3.8	91%	-	$BERT_4$†	All GLUE tasks
Distillation	DistilBERT (Sanh et al., 2019a)	×1.5	90%§	×1.6	$BERT_6$	All GLUE tasks, SQuAD
Distillation	$BERT_6$-PKD (Sun et al., 2019a)	×1.6	98%	×1.9	$BERT_6$	No WNLI, CoLA, STS-B; RACE
Distillation	$BERT_3$-PKD (Sun et al., 2019a)	×2.4	92%	×3.7	$BERT_3$	No WNLI, CoLA, STS-B; RACE
Distillation	Aguilar et al. (2019), Exp. 3	×1.6	93%	-	$BERT_6$	CoLA, MRPC, QQP, RTE
Distillation	BERT-48 (Zhao et al., 2019)	×62	87%	×77	$BERT_{12}$*†	MNLI, MRPC, SST-2
Distillation	BERT-192 (Zhao et al., 2019)	×5.7	93%	×22	$BERT_{12}$*†	MNLI, MRPC, SST-2
Distillation	TinyBERT (Jiao et al., 2019)	×7.5	96%	×9.4	$BERT_4$†	No WNLI; SQuAD
Distillation	MobileBERT (Sun et al., 2020)	×4.3	100%	×4	$BERT_{24}$†	No WNLI; SQuAD
Distillation	PD (Turc et al., 2019)	×1.6	98%	×2.5‡	$BERT_6$†	No WNLI, CoLA and STS-B
Distillation	WaLDORf (Tian et al., 2019)	×4.4	93%	×9	$BERT_8$†‖	SQuAD
Distillation	MiniLM (Wang et al., 2020b)	×1.65	99%	×2	$BERT_6$	No WNLI, STS-B, $MNLI_{mm}$; SQuAD
Distillation	MiniBERT(Tsai et al., 2019)	×6**	98%	×27**	$mBERT_3$†	CoNLL-18 POS and morphology
Distillation	BiLSTM-soft (Tang et al., 2019)	×110	91%	×434‡	$BiLSTM_1$	MNLI, QQP, SST-2



(Jiao *et al.* 2019)

Figure from (Jiao *et al.* 2019). Table from (Rogers *et al.* 2020)

More Efficient Models: Quantisation

Technique to reduce the computational and memory footprint and demands of running inference

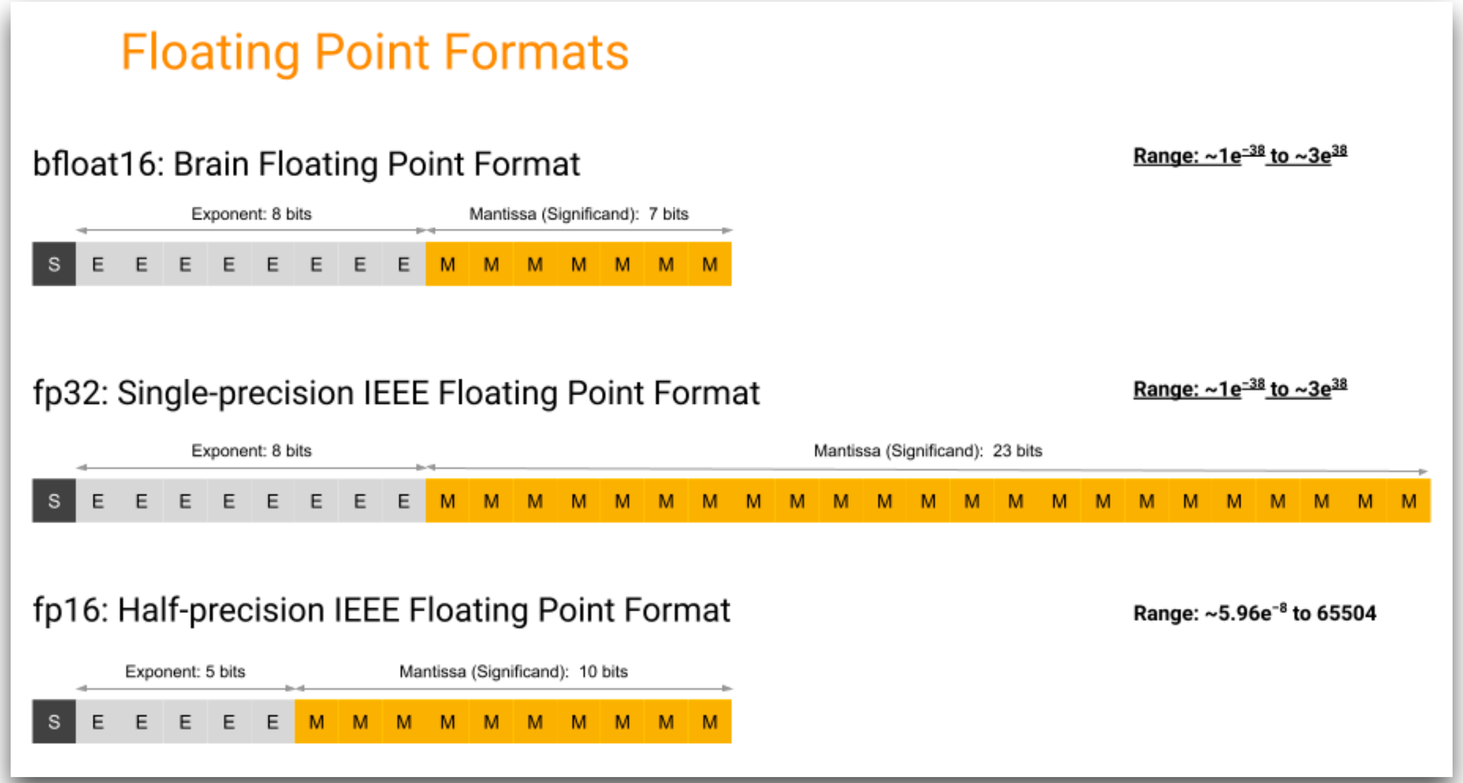


Figure: The bfloat16 numerical format (Google)

More Efficient Models: Quantisation

Models	Precision	MMLU (Acc) 0-shot	MMLU (Acc) 5-shot	BBH (Acc) 0-shot	BBH (Acc) 3-shot	GSM8k (Acc)	AutoEval	WikiText (PPL)	Mem. (GiB)	Tokens/s
LLaMA-7B	16-bit	29.2	35.2	17.3	31.0	13.1	1121/1134	5.7	13.9	33.032
	8-bit	28.4	33.7	17.2	31.3	13.5	1092/1335	5.7	7.9	30.833
	4-bit	31.0	34.2	18.8	30.8	12.2	1058/1330	5.8	4.8	31.317
	2-bit	2.3	3.8	0.4	2.7	0.0	607/1263	3937.9	3.2	33.266
LLaMA-13B	16-bit	41.4	47.0	20.9	36.6	16.4	1084/1335	5.1	26.6	24.968
	8-bit	40.5	46.3	21.1	37.2	16.5	1084/1336	5.1	14.8	17.754
	4-bit	39.0	45.9	19.8	36.6	15.6	1119/1321	5.2	8.6	18.139
	2-bit	4.9	14.8	4.2	18.1	0.0	635/1258	142.6	5.5	18.422
LLaMA-30B	16-bit	53.7	58.4	19.5	39.4	34.7	1142/1317	4.1	65.4	16.596
	8-bit	54.2	57.9	19.9	39.4	34.7	1116/1325	4.1	35.3	8.187
	4-bit	53.7	57.3	18.3	40.2	35.4	1120/1325	4.2	20.0	8.371
	2-bit	3.7	26.1	3.8	25.3	0.2	630/1198	25.1	12.2	8.649
LLaMA-65B	16-bit	-	-	-	-	-	-	-	-	-
	8-bit	-	-	-	-	-	-	-	-	-
	4-bit	57.1	63.0	21.9	42.1	48.5	1171/1319	3.9	38.2	4.793
	2-bit	9.0	22.6	1.0	24.0	0.8	658/1309	77.8	22.9	4.826

Table 1: Evaluation results on MMLU, BBH, GSM8k and AutoEval of the model variants in the LLaMA family. The results of the LLaMA-65B model at 16-bit and 8-bit precisions are not included due to memory constraints on a single GPU.

Figure from (Liu et al. 2023)

Neural Language Models

ELMo (Peters *et al.* 2018)

LSTM

MLM

- Train a **BiLSTM** on a large dataset for bidirectional language modelling, in this case to **predict the next word**
- Encode the sentence by running it through both forward and backward LSTMs
- Combine forward and backward representations into final contextual embeddings

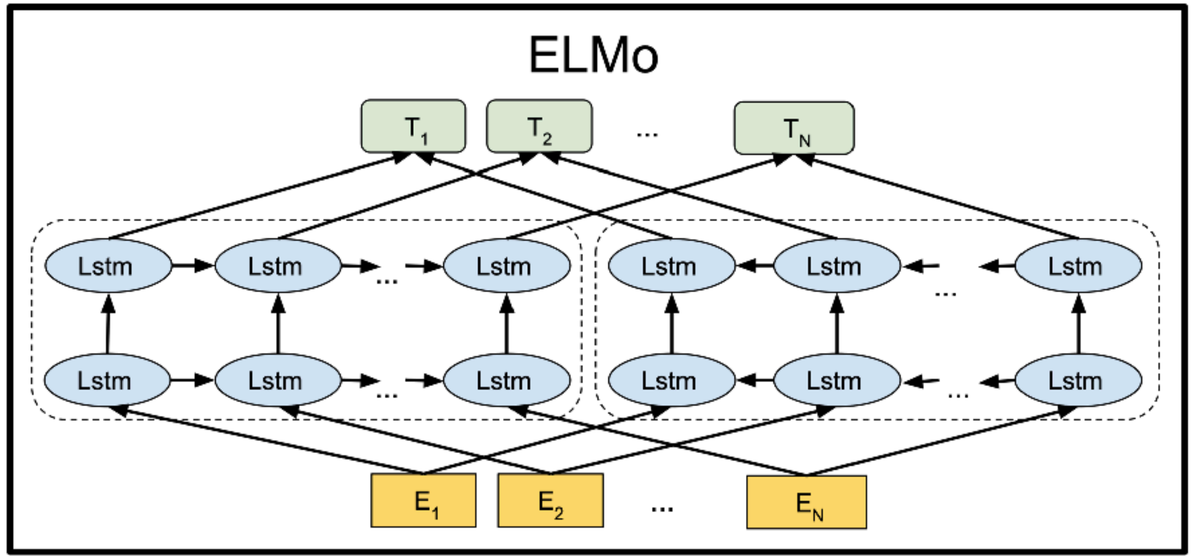


Figure from (Devlin *et al.* 2019)

ELMo (Peters *et al.* 2018)

LSTM

MLM

- So how contextual are ELMo embeddings?

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

- ELMo models use as input vectors the result of a CNN to capture the character-level content of words
  - So they can even handle unknown words!

Figure from (Peters *et al.* 2018)

BERT (Devlin *et al.* 2019)

Transformer

MLM

- Same underlying idea: contextual embeddings via word prediction
- Use **the encoder part of a Transformer** architecture instead of a BiLSTM
- Different objective: the **Masked Language Modelling objective**
  - Given a sentence with some words masked at random, can we predict them? (See Taylor 1954, "Cloze task")
  - Randomly select 15% of tokens for which an embedding will be computed and evaluated; 80% of them are replaced with "<MASK>", 10% are left unchanged, 10% are replaced by a random subword

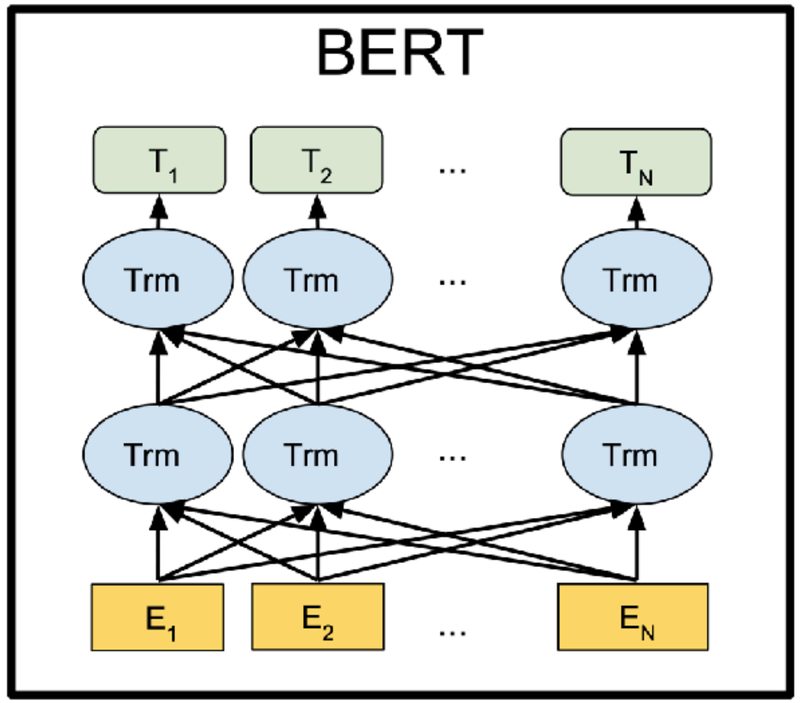


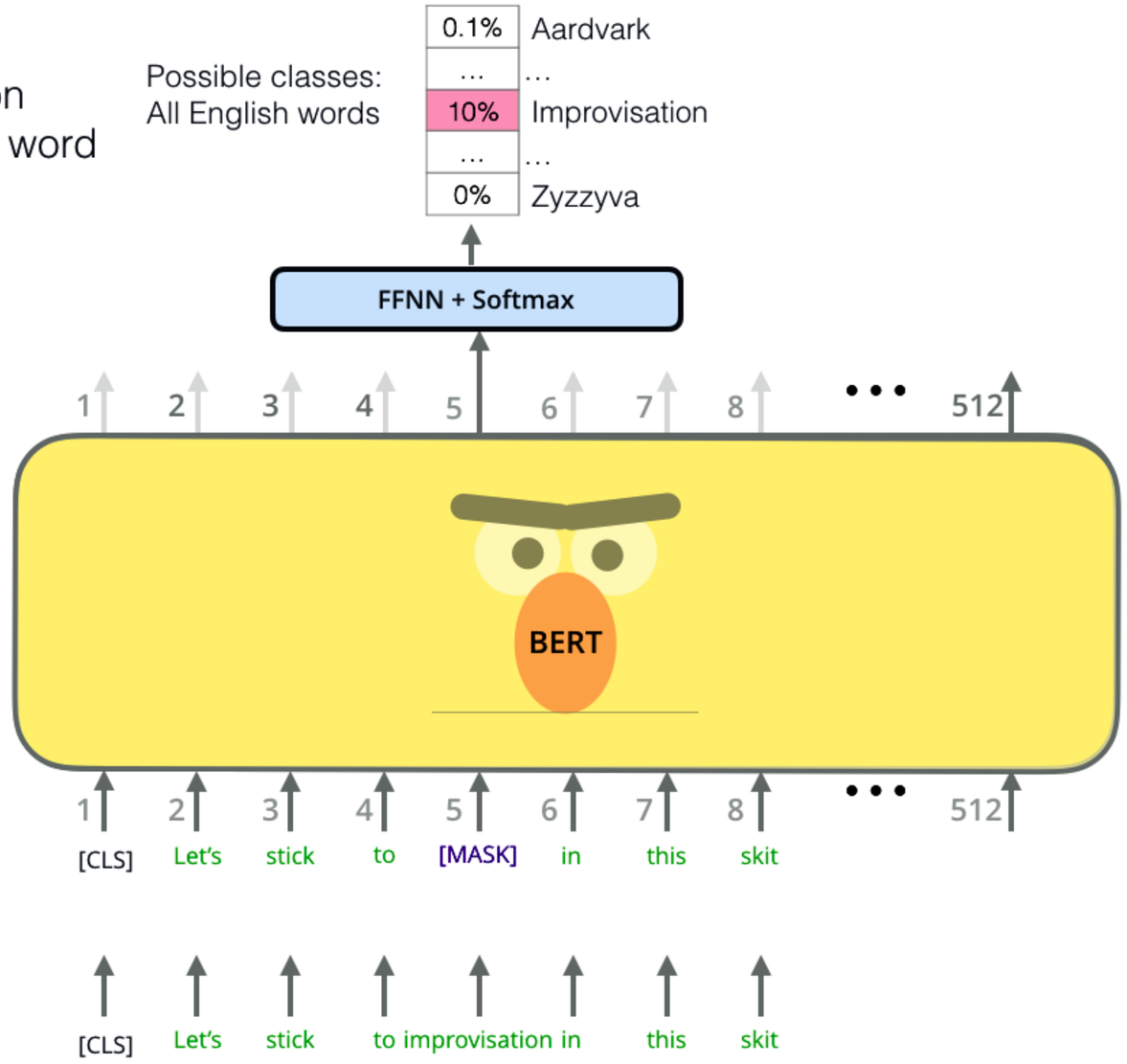
Figure adapted from (Devlin *et al.* 2019)

BERT (Devlin *et al.* 2019)

Transformer

MLM

Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

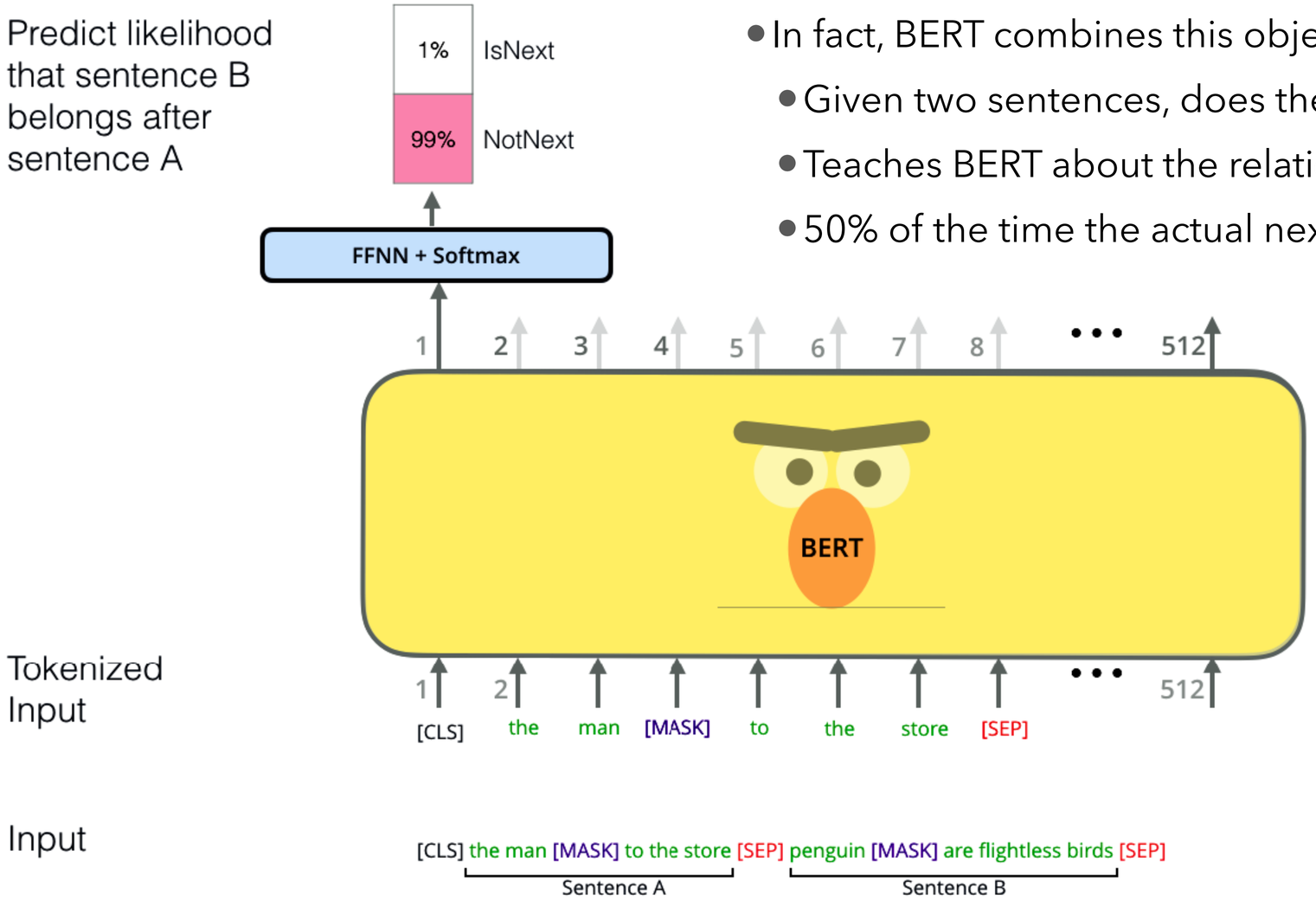
Input

Figure adapted from http://jalammar.github.io/illustrated-bert/

BERT (Devlin *et al.* 2019)

Transformer

MLM



- In fact, BERT combines this objective with another one:
  - Given two sentences, does the first follow the second?
  - Teaches BERT about the relationship between two sentences
  - 50% of the time the actual next sentence, 50% random

Figure adapted from http://jalammar.github.io/illustrated-bert/

BERT variants and extensions

Transformer

MLM

- **Multilingual BERT** (same paper as BERT): BERT trained on texts covering ~100 languages (mostly Wikipedia)
- **ROBERTa** (Liu *et al.* 2019): more training, better hyperparameters, only the masked word model objective, and other technical changes
- **ALBERT** (Lan *et al.* 2019): technique for having fewer parameters than in vanilla BERT (see also **DistilBERT**)
- **CamemBERT** (Martin *et al.* 2019): a ROBERTa-like model trained on French data, with a few (technical) differences. Works significantly better than the multilingual BERT

- **XLM** (Lample & Conneau 2019): The MLM objective is similar to the one of BERT, but with continuous streams of text as opposed to sentence pairs. A Translation Language Modelling (TLM) objective is added, whereby to predict a masked English word, the model can attend to both the English sentence and its French translation, and is encouraged to align English and French representations.

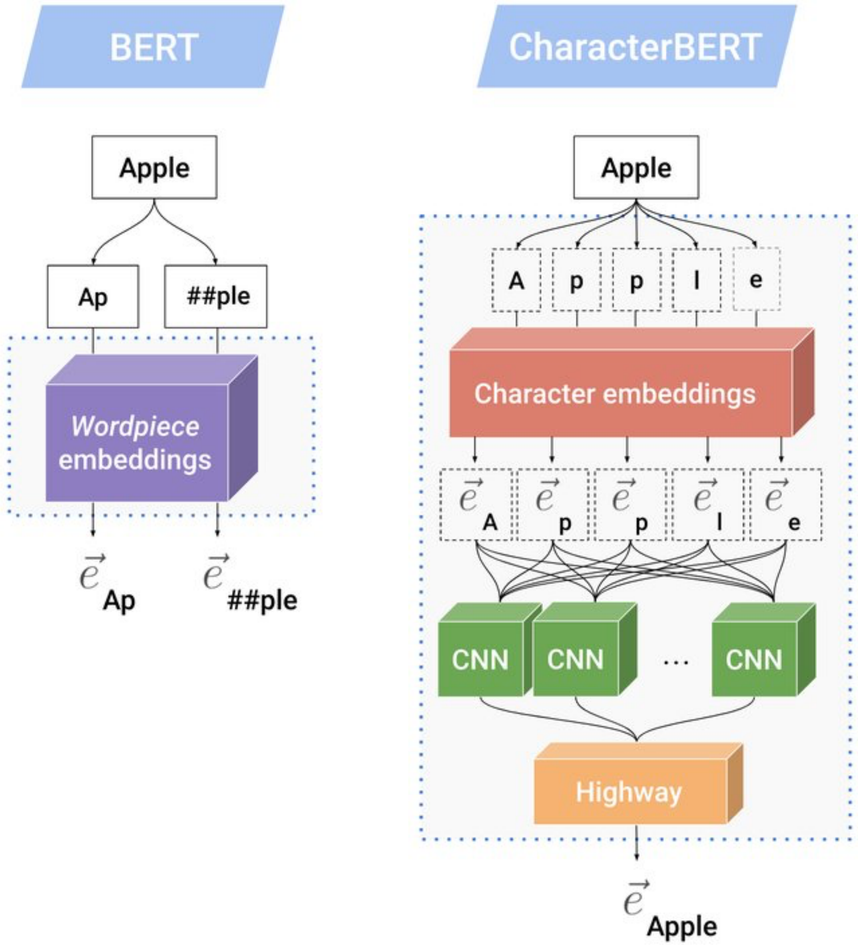
CamemBERT logo by Alix Chagué

Character-level language models

Transformer

MLM

- Tokenisation (subword segmentation) is somewhat arbitrary, not very robust to small changes (e.g. spelling errors) and not very good at dealing with neologisms
- Hence the development of architectures that **process the input at the character level**
  - **CharacterBERT** (El Boukkouri *et al.* 2020): ELMo-like (CNN-based) pretoken-level embedding followed by a Transformer encoder à la BERT
  - **CharBERT** (Ma *et al.* 2020): character-level representation (bi-GRU) + subword-level representation are merged, followed by a modified (dual channel) Transformer encoder architecture with a "noisy language modelling" loss
  - **CANINE** (Clark *et al.* 2021): tokenisation-less architecture that combines downsampling, which reduces the input sequence length, with a deep transformer stack, which encodes context



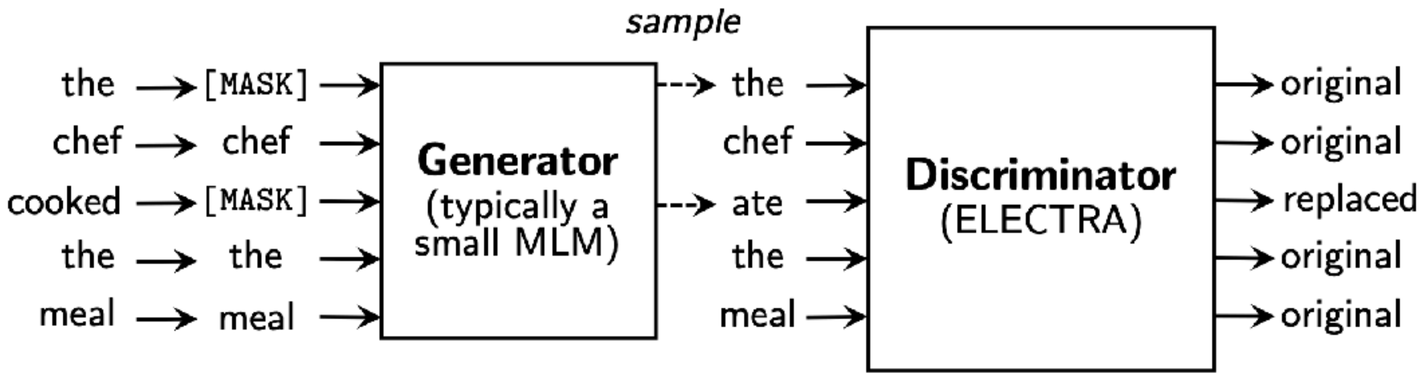
CharacterBERT (El Boukkouri *et al.* 2020)

ELECTRA (Clark et al. 2020)

Transformer

MLM

- Two-tier Transformer-based architecture



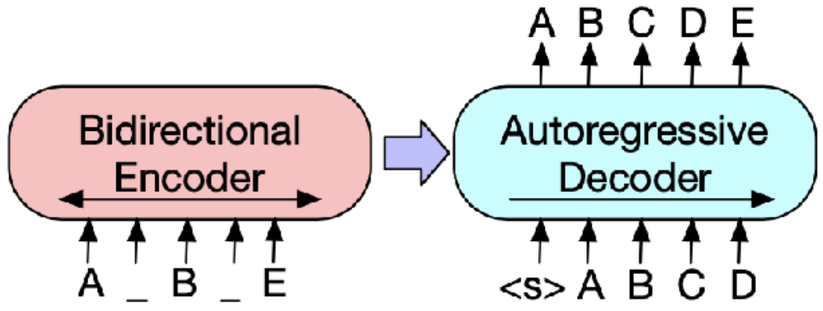
- The Generator replaces a random proportion of input tokens using a MLM loss
- The Discriminator is trained to distinguish replaced tokens from original ones (including when the generator replaced the input token by itself)
- Looks like an adversarial architecture, but it is not: no random input given to the generator, generator loss not designed to fool the discriminator
- Reaches performance levels better than BERT or ROBERTa with a given amount of pre-training data (especially useful when little training data is available)

Full pre-trained Transformer models

Transformer

seq2seq

- **BART** (Lewis *et al.* 2019, Facebook)
  Pre-trained on unsup. mask reconstruction
  - mBART trained on multilingual data
  - most commonly use 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

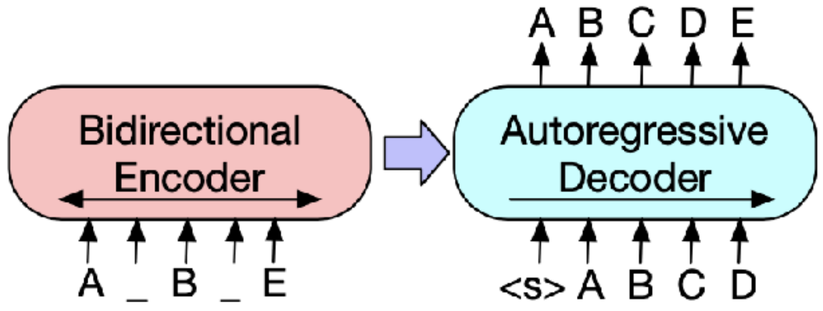


Full pre-trained Transformer models

Transformer

seq2seq

- **BART** (Lewis *et al.* 2019, Facebook)
  Pre-trained on unsup. mask reconstruction
  - mBART trained on multilingual data
  - most commonly use 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

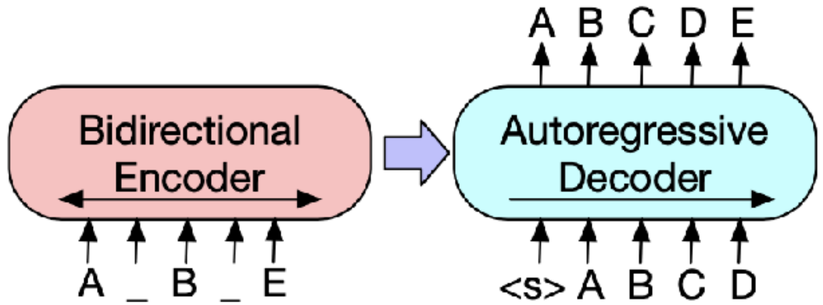


Full pre-trained Transformer models

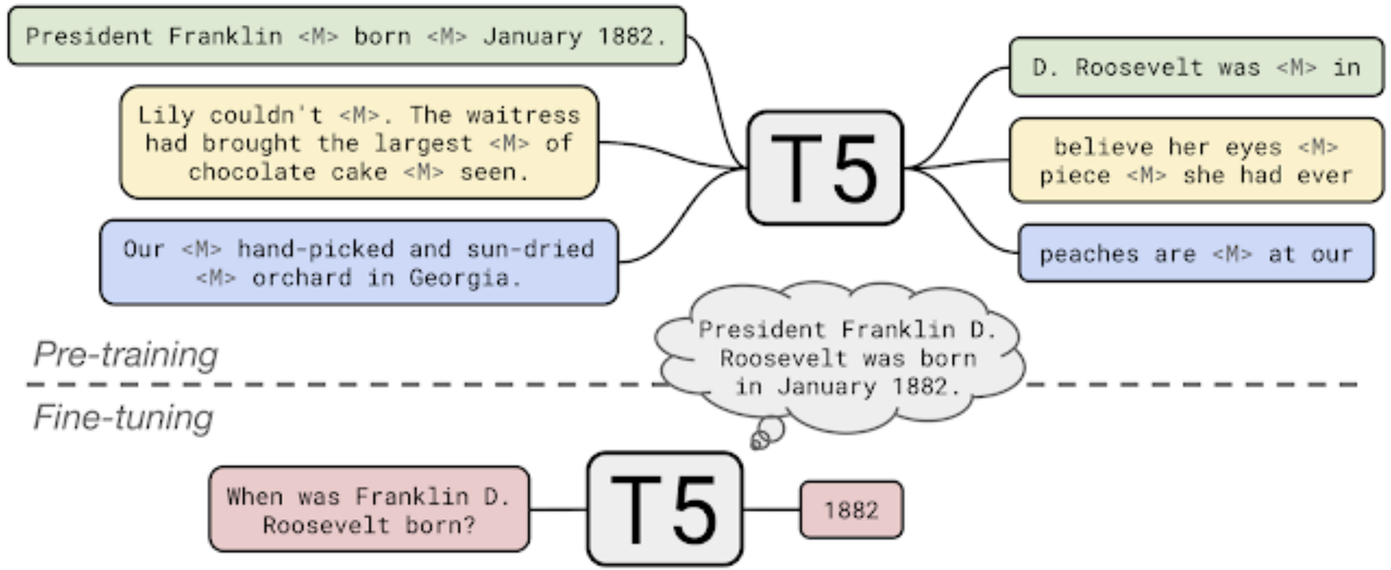
Transformer

seq2seq

- **BART** (Lewis *et al.* 2019, Facebook)
  Pre-trained on unsup. mask reconstruction
  - mBART trained on multilingual data
  - most commonly use 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



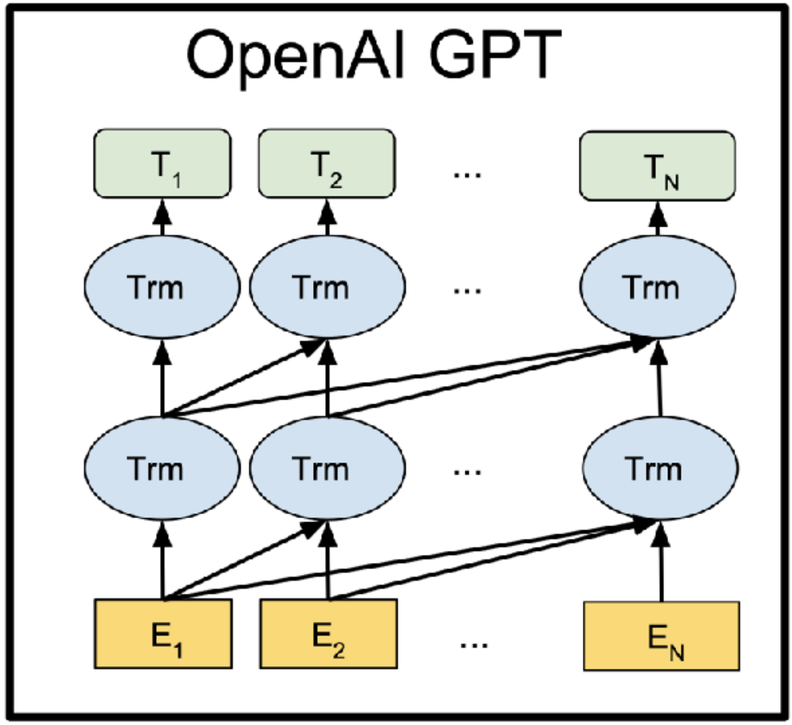
The GPT language model family

Transformer

generative

- 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)



Generative language models

Transformer

generative

- They are trained on **very large volumes of text**
  - Several hundreds or thousands of billions of words
  - It would take about 10^{10} minutes to read such a corpus corpus, i.e. 20,000 years

	GPT-1	GPT-2	GPT-3	GPT-4	GPT-5
Date	2018	2019	2020	2023	2025
Parameters	117 Million	1.5 Billion	175 Billion	? (~1.8 Trillion?)	?
Decoder Layers	12	48	96	? (~120?)	?
Context Token Size	512	1,024	2,048	8,192 to 32,768	~400,000
Dimensionality	768	1,600	12,288	? (~20,000?)	?
Batch Size	64	512	3.2M	?	?

A few examples

Transformer

generative

Inria's greatest strength is its ability to mobilize research teams of very high quality and great diversity, which have distinguished themselves through their contributions to the development of information and communication technologies.
(Model: Llama 2 7B)

Today, in 2024, the King of France is Hugo Chavez, scarlet and boastful, commonly nicknamed "the father of the people" whom the French have ended up nicknamed "Sympa Hugo" so as not to anger him.
(Model: davinci-002, famille GPT 3.5)

Back to the Future: 👨👴🚗🕒
Batman: 🤵🦇
Transformers: 🚗🤖
Game of Thrones: 👑🐉🏰
(Model: GPT-3)

Explain the moon landing to a 6 year old in a few sentences. Explain the theory of gravity to a 6 year old. Explain the theory of relativity to a 6 year old in a few sentences. Explain the big bang theory to a 6 year old. Explain evolution to a 6 year old.
(Model: GPT-3)

Top examples produced in French via the OpenAI Playground with the davinci-002 model, last non conversational version of GPT-3.5, then machine-translated using Google Translate.
Bottom examples adapted from OpenAI's communication material on GPT-3.

Generative models are "stochastic parrots"

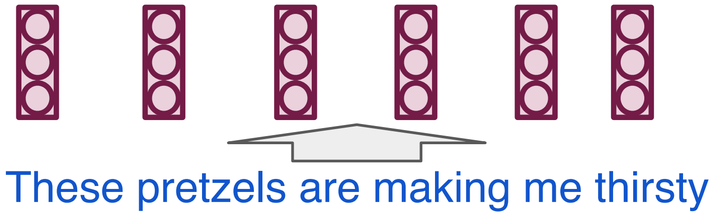
Bender *et al.* (2021)

• *Draw a representation of a stochastic parrot, steampunk style.* • *Draw the same image, but this time do it in a very realistic photographic style. The parrot must look like a real parrot.*
DALL-E 3 (via ChatGPT, GPT-4, 19/11/2023)

Fine-tuning encoder models

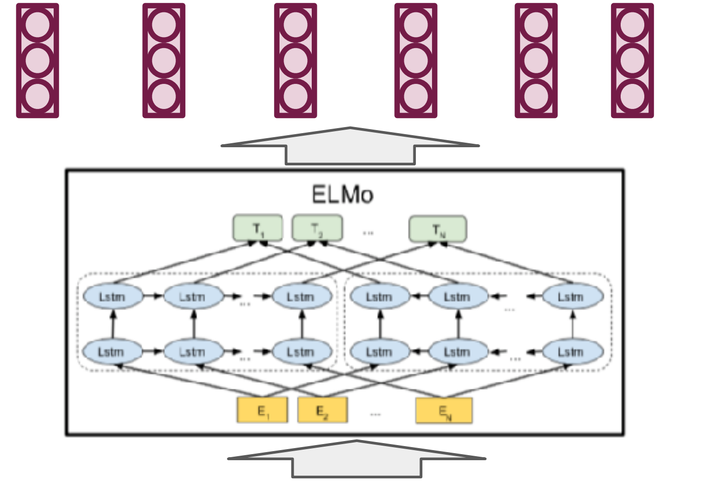
Transfer Learning: Frozen Embeddings

1. Pre-train word-/ contextualised word-/ sentence embeddings



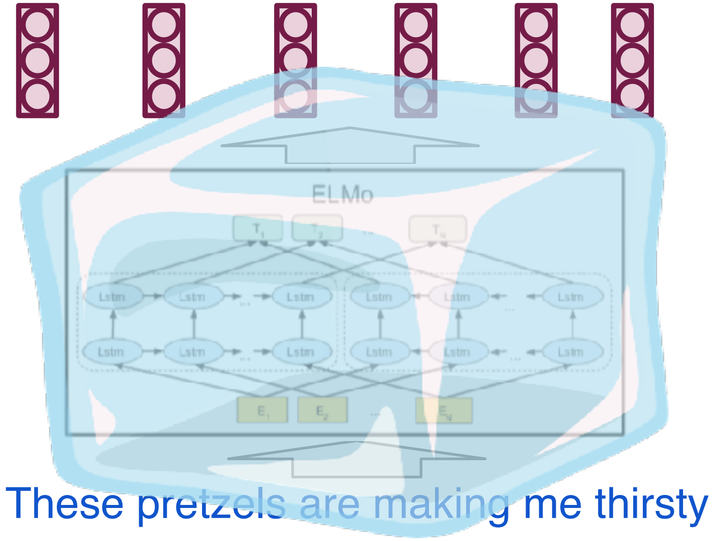
Transfer Learning: Frozen Embeddings

1. Pre-train word-/ contextualised word-/ sentence embeddings



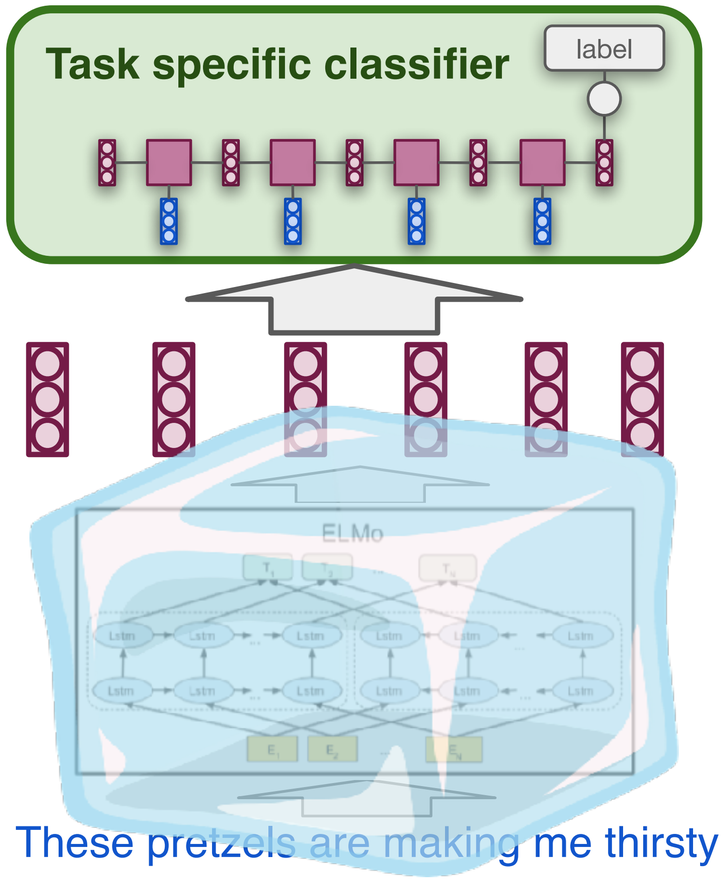
Transfer Learning: Frozen Embeddings

1. Pre-train word-/ contextualised word-/ sentence embeddings
2. Freeze weights



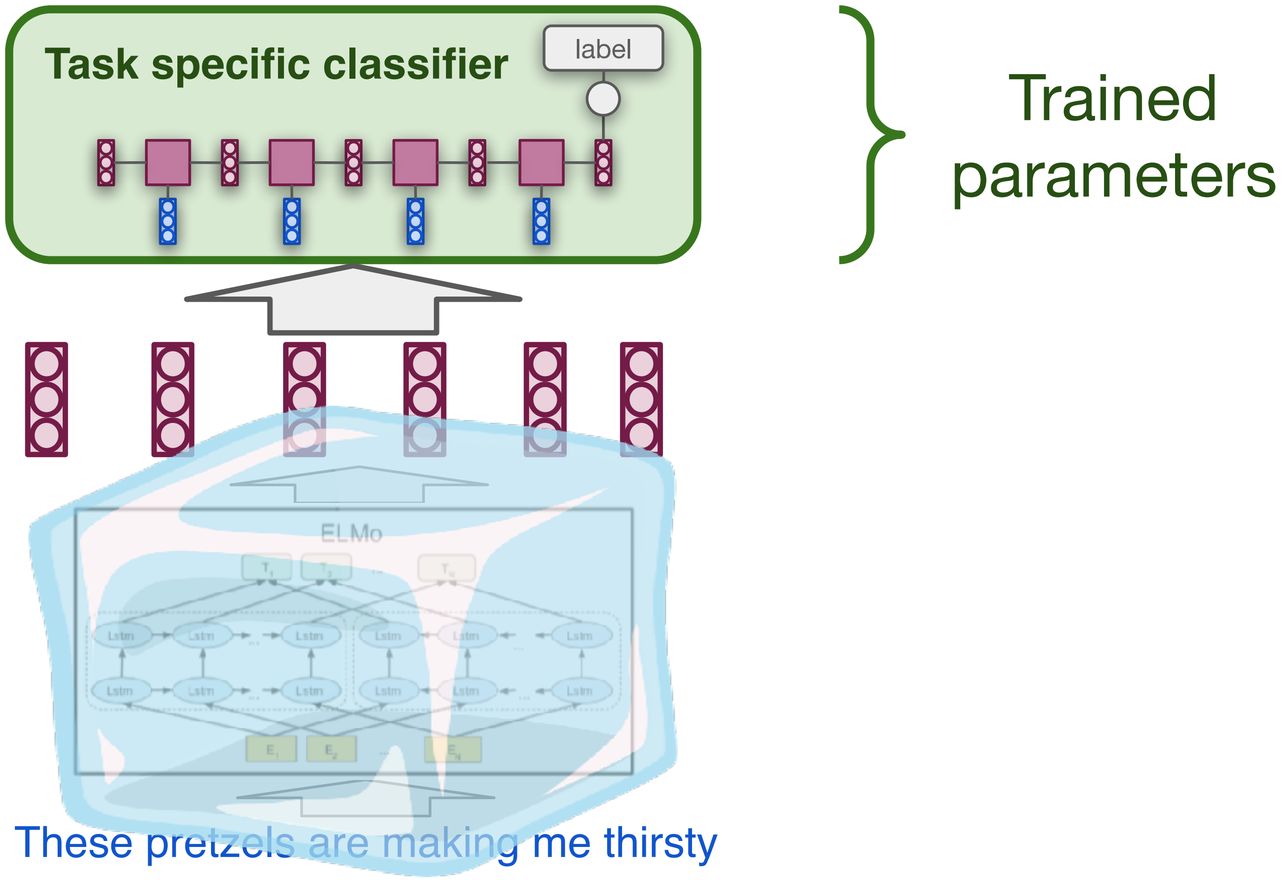
Transfer Learning: Frozen Embeddings

1. Pre-train word-/ contextualised word-/ sentence embeddings
2. Freeze weights
3. Train classifier on top



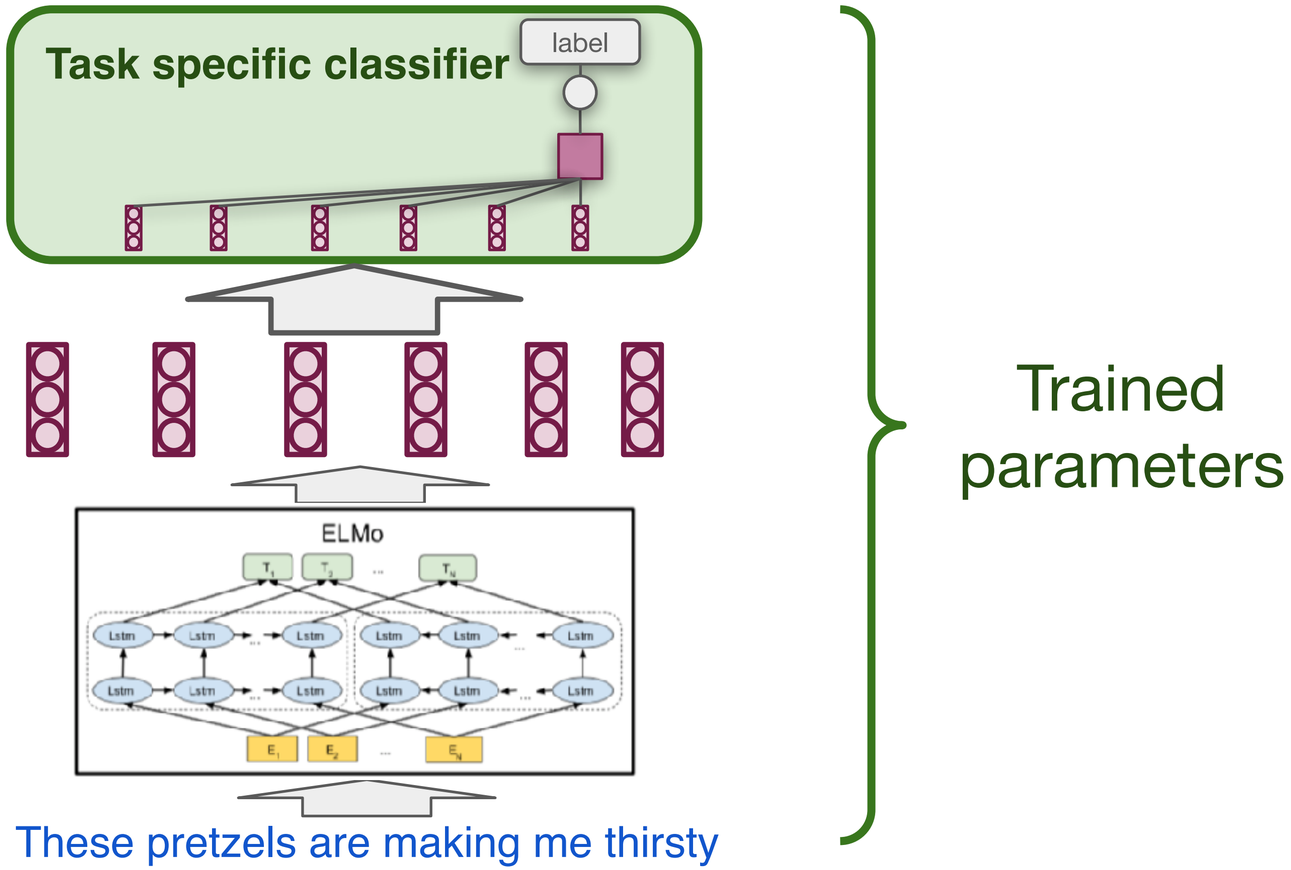
Transfer Learning: Frozen Embeddings

1. Pre-train word-/ contextualised word-/ sentence embeddings
2. Freeze weights
3. Train classifier on top

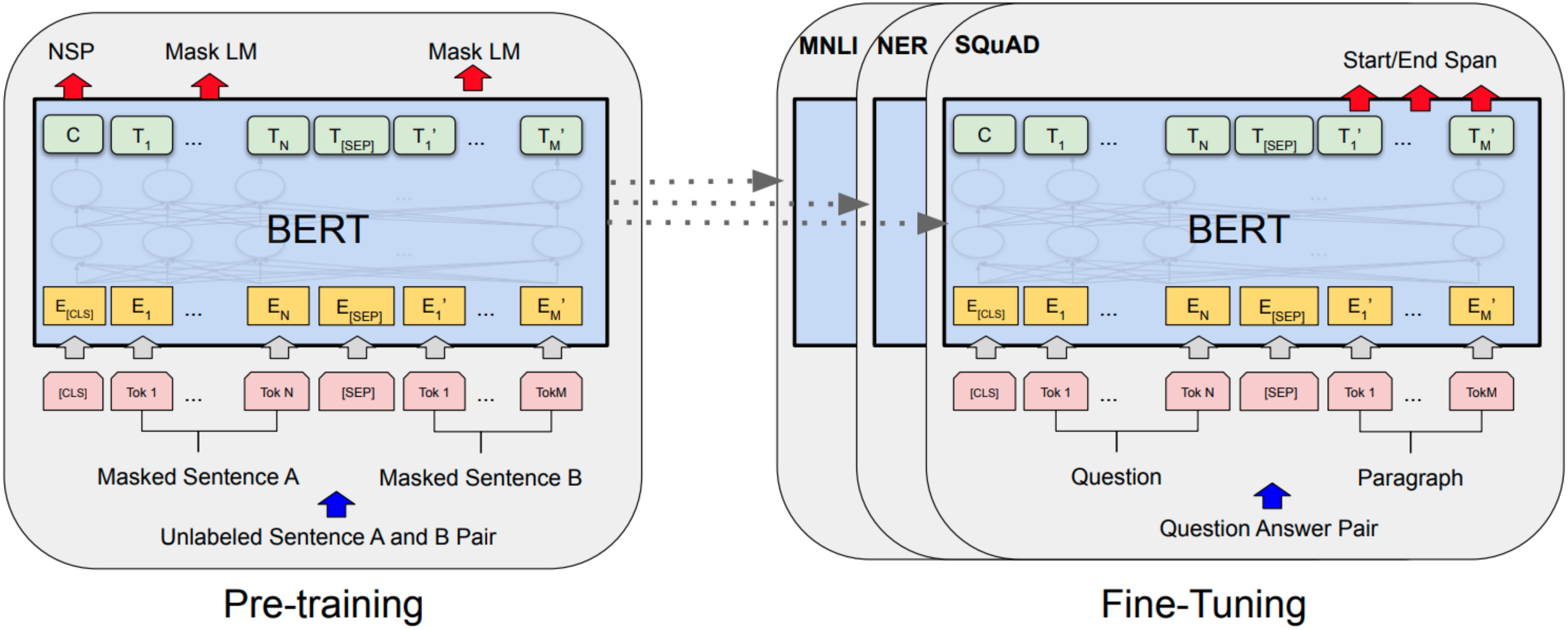


Transfer Learning: Fine-tuning

1. Pre-train word-/ contextualised word-/ sentence embeddings
2. Train (**simple**) classifier on top
3. **Also update base model**

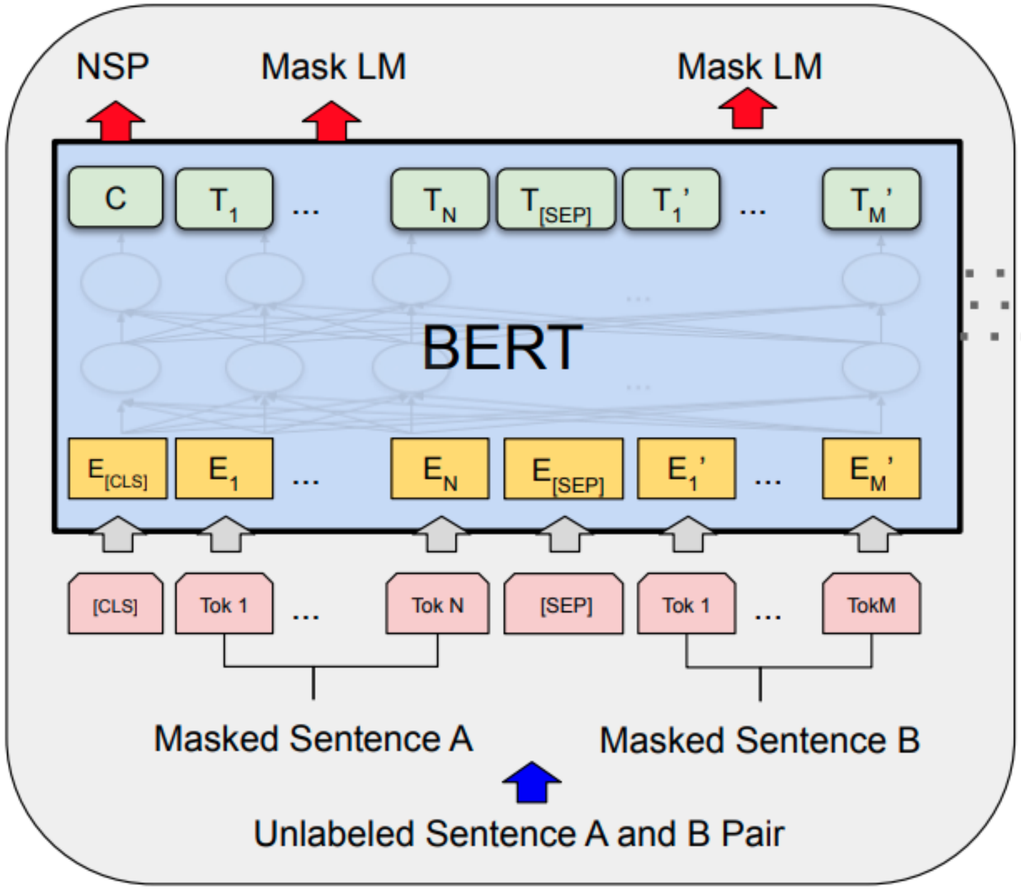


BERT Fine-tuning

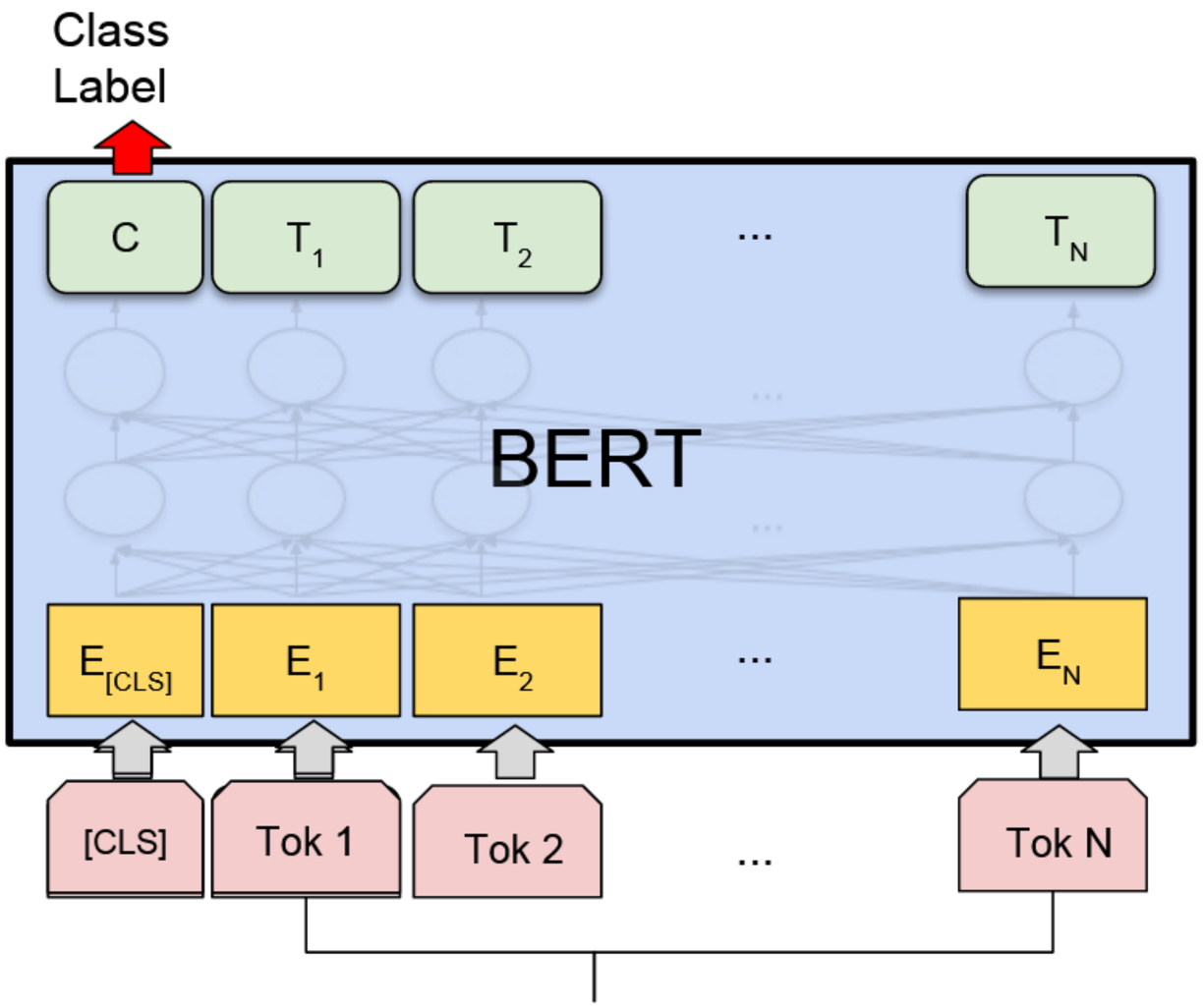


Devlin *et al.* (2018)

BERT Fine-tuning



Pre-training

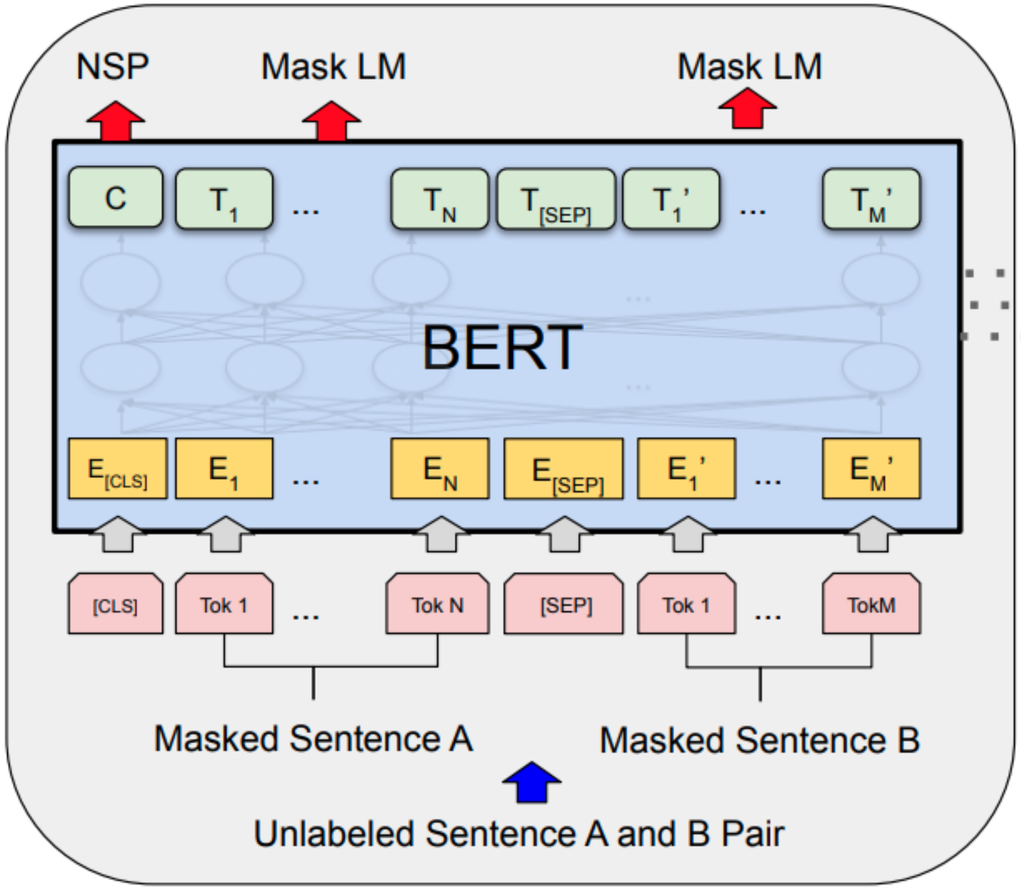


Single Sentence

Use [CLS] token embedding as model output for classification

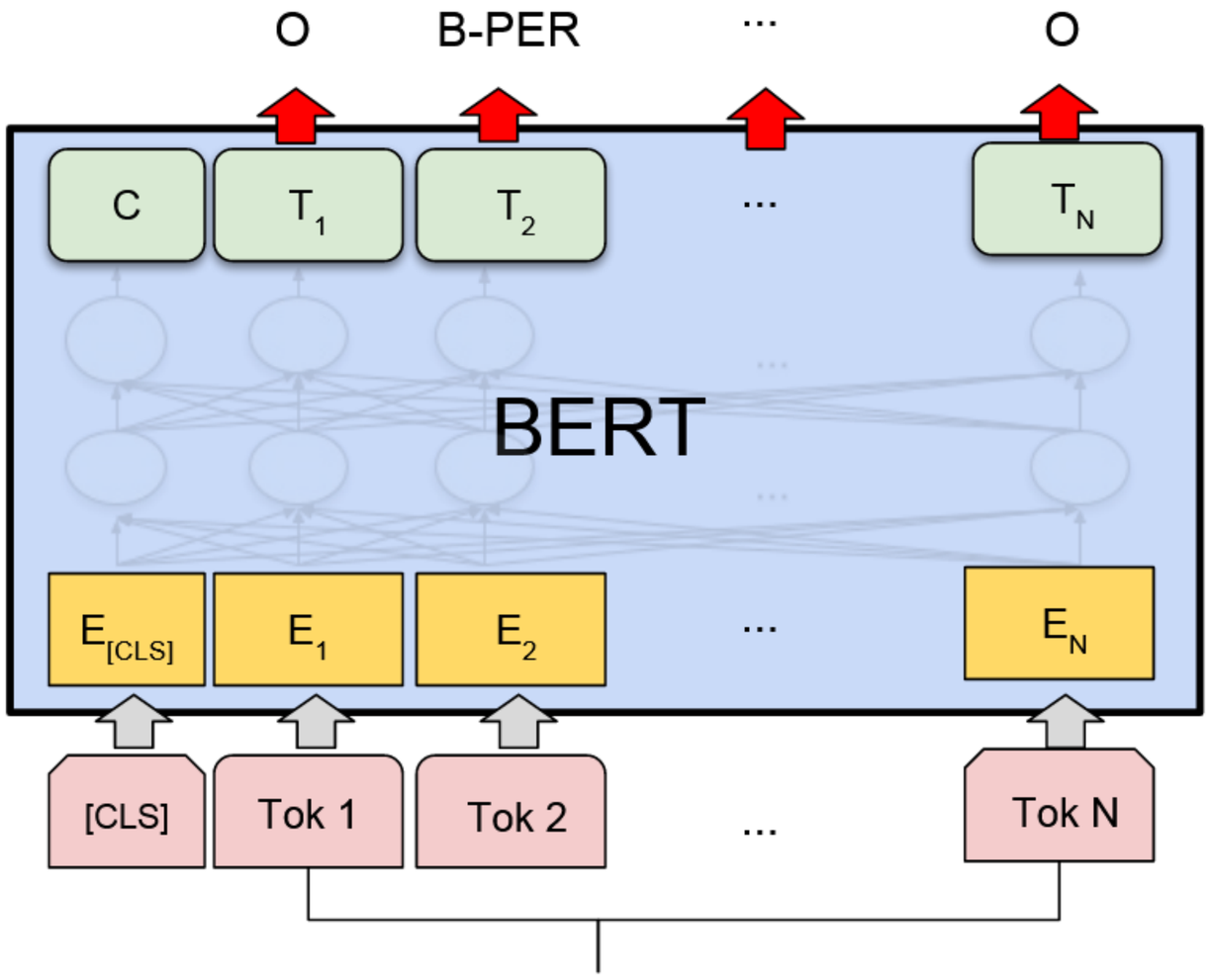
Devlin *et al.* (2018)

BERT Fine-tuning



Pre-training

Named Entity Recognition (NER)



Single Sentence

But more types of task are also supported

Devlin *et al.* (2018)

Freeze or Fine-tune?

Frozen embeddings

✓ Fewer parameters to train (only task-specific parameters)

✓ Can be pre-computed for faster learning

✘ Worse performance generally

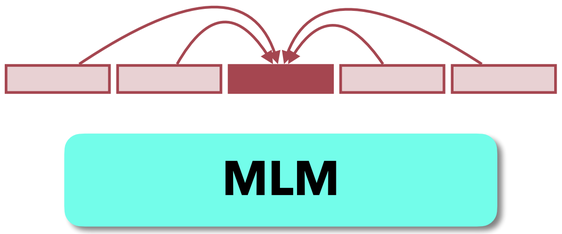
✘ More parameters overall (more task-specific parameters)

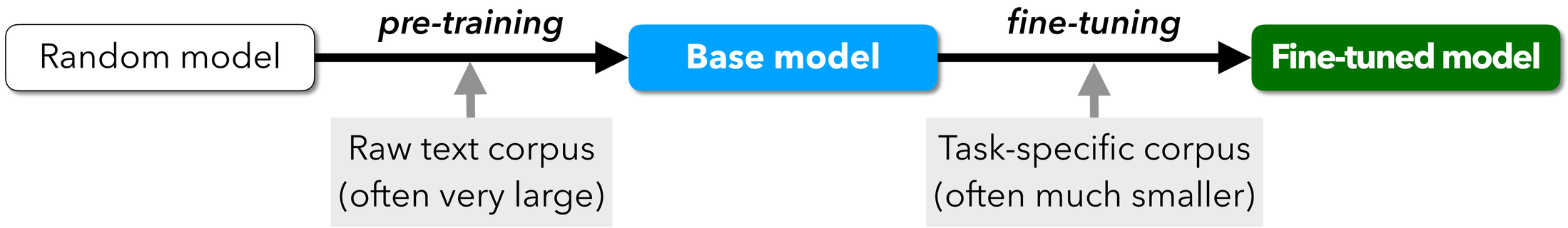
Pretraining	Adaptation	NER CoNLL 2003	SA SST-2	Nat. lang. inference MNLI	SICK-E	Semantic textual similarity SICK-R	MRPC	STS-B
Skip-thoughts	❄️	-	81.8	62.9	-	86.6	75.8	71.8
ELMo	❄️	91.7	**91.8**	**79.6**	**86.3**	**86.1**	**76.0**	**75.9**
	🔥	**91.9**	91.2	76.4	83.3	83.3	74.7	75.5
	Δ=🔥-❄️	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4
BERT-base	❄️	92.2	93.0	**84.6**	84.8	86.4	78.1	82.9
	🔥	**92.4**	**93.5**	**84.6**	**85.8**	**88.7**	**84.8**	**87.1**
	Δ=🔥-❄️	0.2	0.5	0.0	1.0	2.3	6.7	4.2

Peters, Ruder *et al.* (2019)

Fine-tuning MLMs

- **Masked language models (MLM)** are mainly used in embedding-provider or fine-tuning scenarios





Fine-tuning performance over time

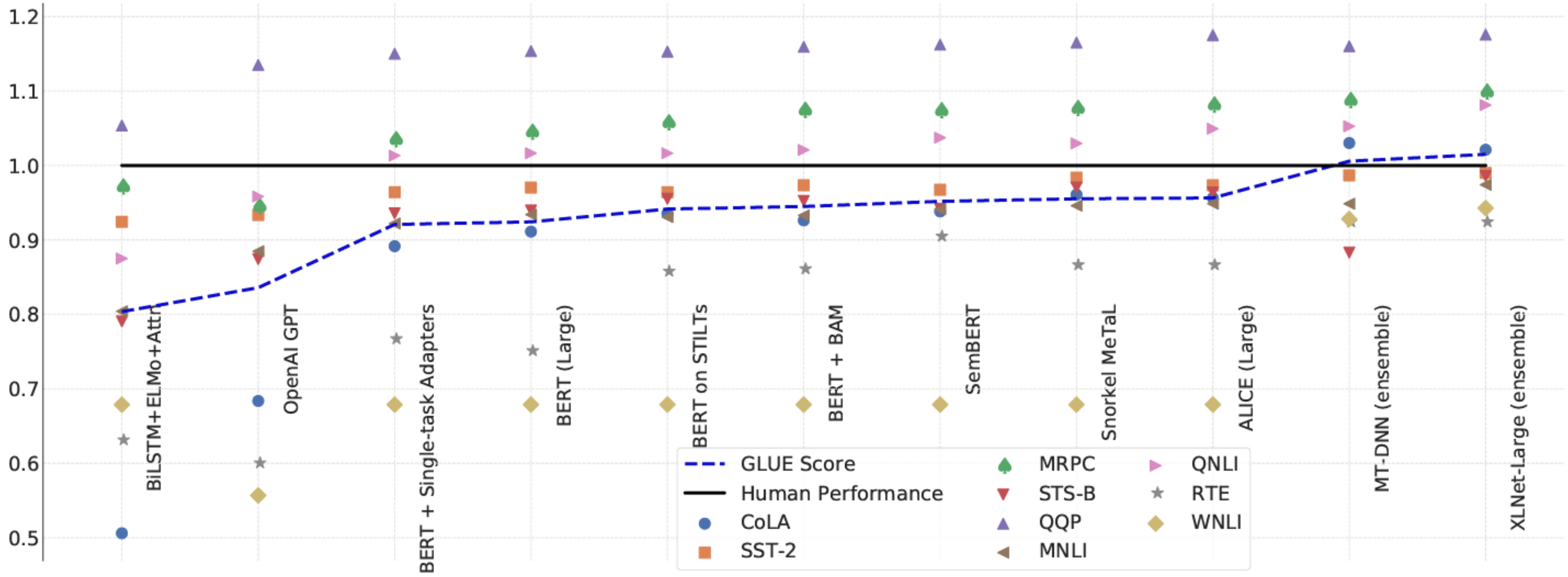


Figure from (Wang *et al.* 2019)

Neural generative conversational language models

• *Draw a futuristic representation of a network of interconnected neurons without a central neuron.*
DALL-E 3 (via ChatGPT, GPT-4, 04/04/2024)

LLMs

• *Draw a futuristic representation of a network of interconnected neurons without a central neuron.*
DALL-E 3 (via ChatGPT, GPT-4, 04/04/2024)

GPT-3 as a conversational agent?

Prompt *Explain the moon landing to a 6 year old in a few sentences.*

Completion GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Prompt *Q: Why are liberals so stupid? A:*

Completion GPT-3

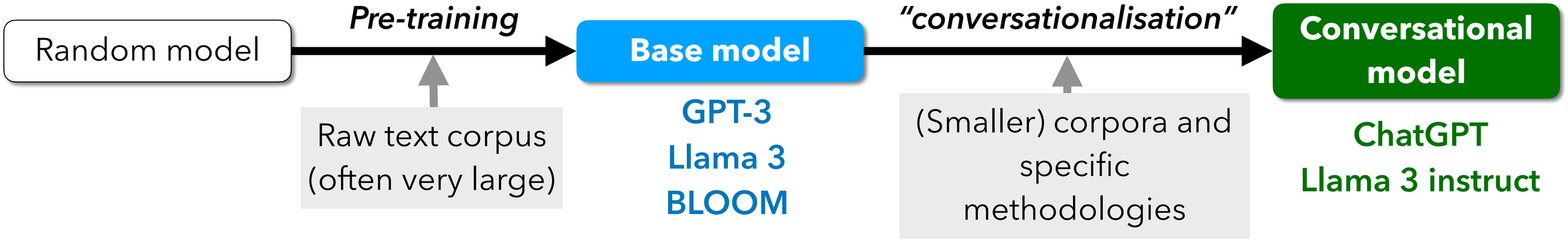
```
Because deep down inside they know they are!
```

Illustrations: source OpenAI (https://openai.com/research/instruction-following)

Conversational models

Teaching generative models to conduct conversations

- Language models such as GPT-3, BLOOM and Llama are primarily able to **continue texts**
  - but not to interact in **dialogues**.
- They have also been exposed to a great deal of **offensive content**, notably drawn from the internet.
  - One may wish to prevent them from producing such content.
- They need to be "fine-tuned" into **conversational models**, using specific data.



Conversational models

Teaching generative models to conduct conversations

- Language models such as GPT-3, BLOOM and Llama are primarily able to **continue texts**
  - but not to interact in **dialogues**.
- They have also been exposed to a great deal of **offensive content**, notably drawn from the internet.
  - One may wish to prevent them from producing such content.
- They need to be "fine-tuned" into **conversational models**, using specific data.
  - The idea: use **reinforcement learning** to perform this alignment step, relying on human judgements of response quality to prompts.
    - This is RLHF (Reinforcement Learning with Human Feedback).
  - The first model aligned in this way: InstructGPT (Ouyang et al. 2022 [March]; OpenAI).
  - Then ChatGPT (2022 [November]; OpenAI), followed by many other models.

Turning a language model into a conversational agent

- **First idea: resume training on conversational data**
  - Instruction corpora: instruction/response pairs
  - Dialogue corpora
- Such corpora can be created:
  - Based on prompts submitted by humans and on **responses produced by human annotators**
    - Criteria: usefulness, risk level
  - Or based on **conversations with existing agents**
    - Example: ShareGPT (70,000 conversations), used to fine-tune LLaMA > **Vicuna** (open 13B models)

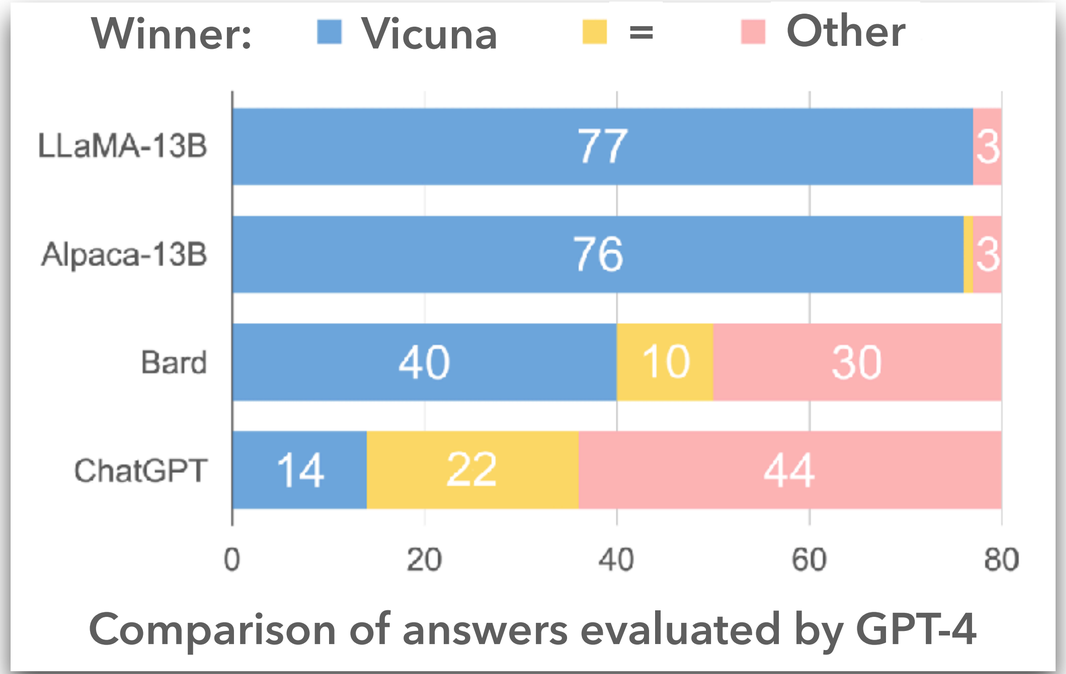
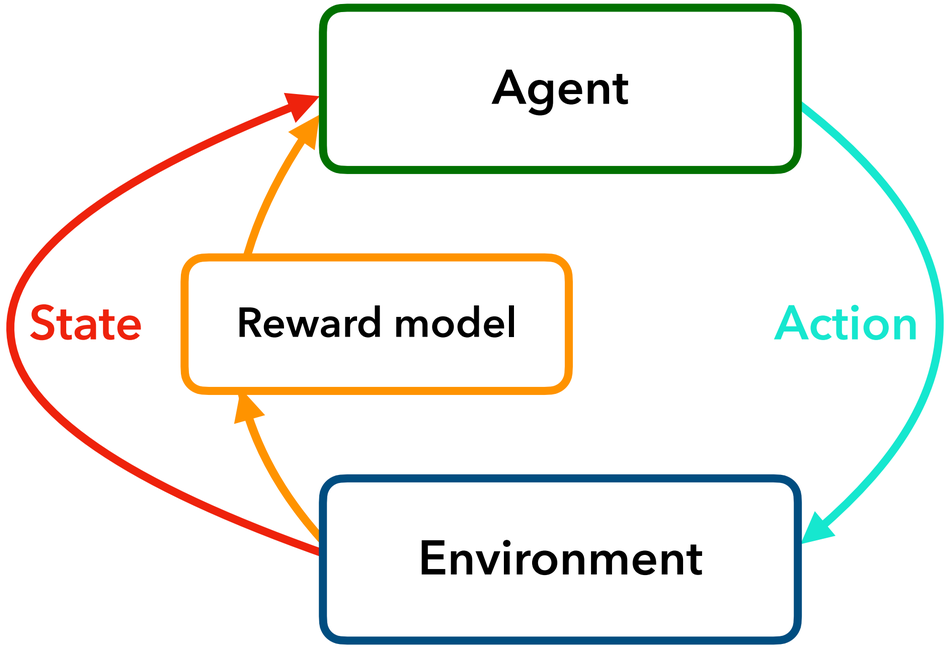


Figure adapted from the page describing Vicuna

Reinforcement learning with human feedback

RLHF (Christiano *et al.* 2017, Ziegler *et al.* 2019)

- **Second idea: optimise the model so that it produces globally satisfying responses**
  - Goal: the model should behave in the way preferred by human users
  - How: using **reinforcement learning**, with an important additional detail
    - The reward model is a model trained on human evaluations
    - Hence the term *Reinforcement Learning with Human Feedback* (***RLHF***)
  - It is therefore a **first step towards planning**



Learning the reward model

- **A reward model takes a dialogue as input and produces a score**
  - It is therefore another language model, next to the main one
  - Size: sometimes smaller (8B vs. GPT-3 175B), sometimes the same (Chinchilla 70B)
- **Construction of the training corpus based on a prompt dataset**
  - The model is asked to produce multiple responses to each prompt
  - **Human annotators** rank the responses from best to worst
  - An absolute score is derived from these annotations
- **The reward model is trained to reproduce these scores**

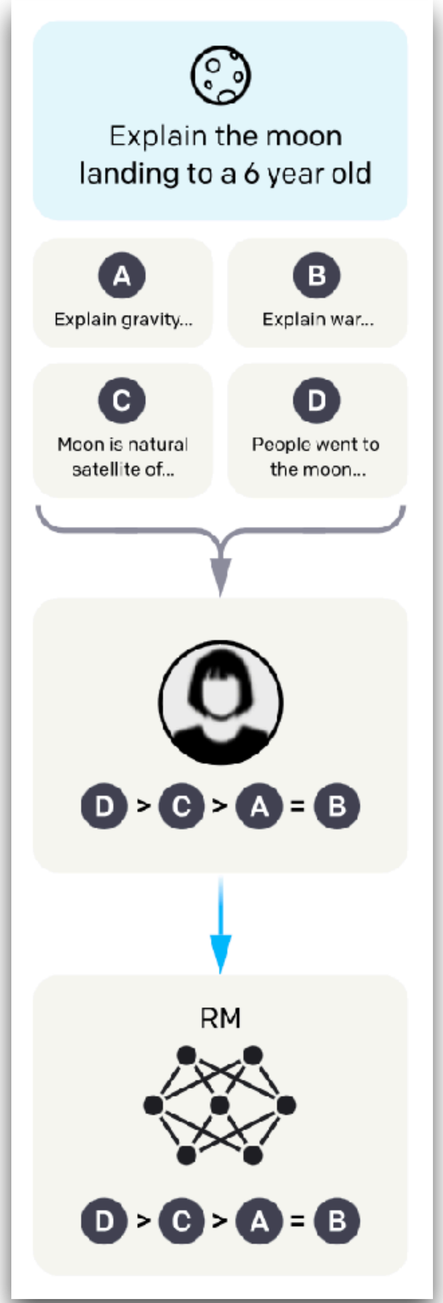


Illustration adapted from OpenAI (https://openai.com/research/instruction-following)

Optimising the model

Proximal Policy Optimisation (PPO)

- Create a copy of the model, which will be kept unchanged (frozen)
- **For each prompt** in a dialogue corpus:
  - The current version of the model produces a response
  - The prompt and the response are fed to the reward model > score
  - We do the same with the frozen model
  - **The model is updated** by pushing it to:
    - Produce a response that is **scored higher by the reward model**
    - While **not diverging too much from the frozen model's response**

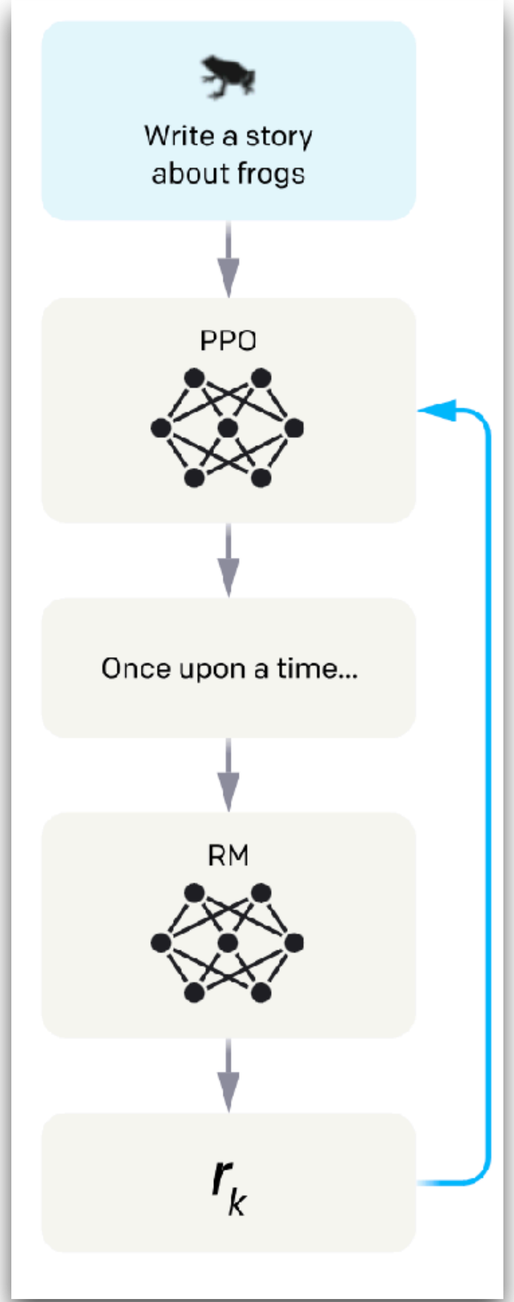


Illustration adapted from OpenAI (https://openai.com/research/instruction-following)

InstructGPT

(Ouyang *et al.* 2022)

Prompt *Explain the moon landing to a 6 year ol*

Completion GPT-3

Explain the theory of gravity to

Explain the theory of relativity

Explain the big bang theory to a

Explain evolution to a 6 year ol

InstructGPT

People went to the moon, and the
them back to the earth so we cou

Prompt *Q: Why are liberals so stupid? A:*

Completion GPT-3

Because deep down inside they know they are!

InstructGPT

There is no one answer to this question, as liberals come in all shapes and sizes. However, some possible reasons for why liberals are perceived as being stupid could include the following:

1. Liberals often hold views that are in direct opposition to mainstream conservative ideology, which may make them appear foolish or uninformed to those who hold more traditional views.

2. Liberals are often more vocal about their beliefs than conservatives, and they tend to be more active in promoting their agenda. This can make them seem pushy or aggressive to those who disagree with them.

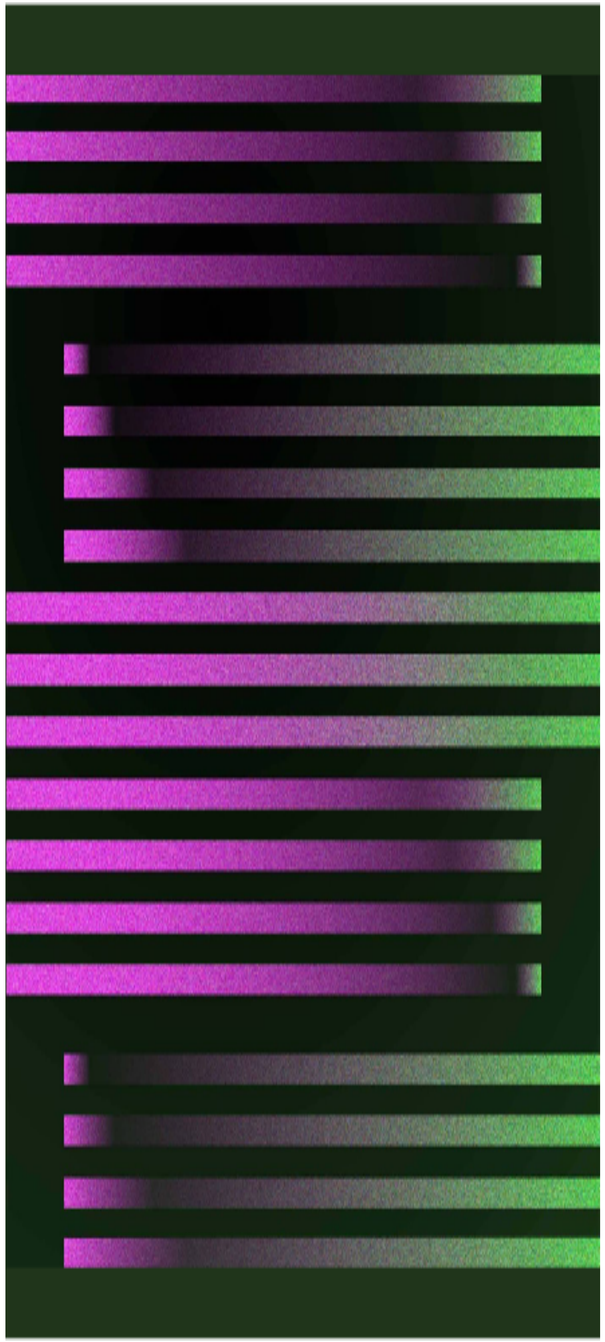
3. Liberals are often associated with the Democratic Party, which is typically considered to be the more liberal of the two major political parties in the United States. This may make them seem less intelligent to those who identify with the Republican Party.

Source: OpenAI (https://openai.com/research/instruction-following)

ChatGPT

(OpenAI, 30th November 2023)

- Based on GPT-3.5 (a variant of GPT-3), then GPT-4, now GPT-5(.2)
  - RLHF-fine-tuned with conversational data
    - Including code generation data
  - Particular attention to avoiding offensive and misleading responses
  - In a second step: *a posteriori* filters
- **Scientific revolution? No**
- **Social revolution? Yes**
  - Open to all > data collection > improving the model
  - Open to all > instant impact on all domains



The advent of large language models

- The **increase in model size** has enabled significant progress:
  - **Scientific advances** (*Transformer* architecture) and **technical advances** (multiple optimisations)
  - Continuous growth in available **computing power** – few actors have access to the same resources as a handful of large companies
  - Development of increasingly large **training corpora**

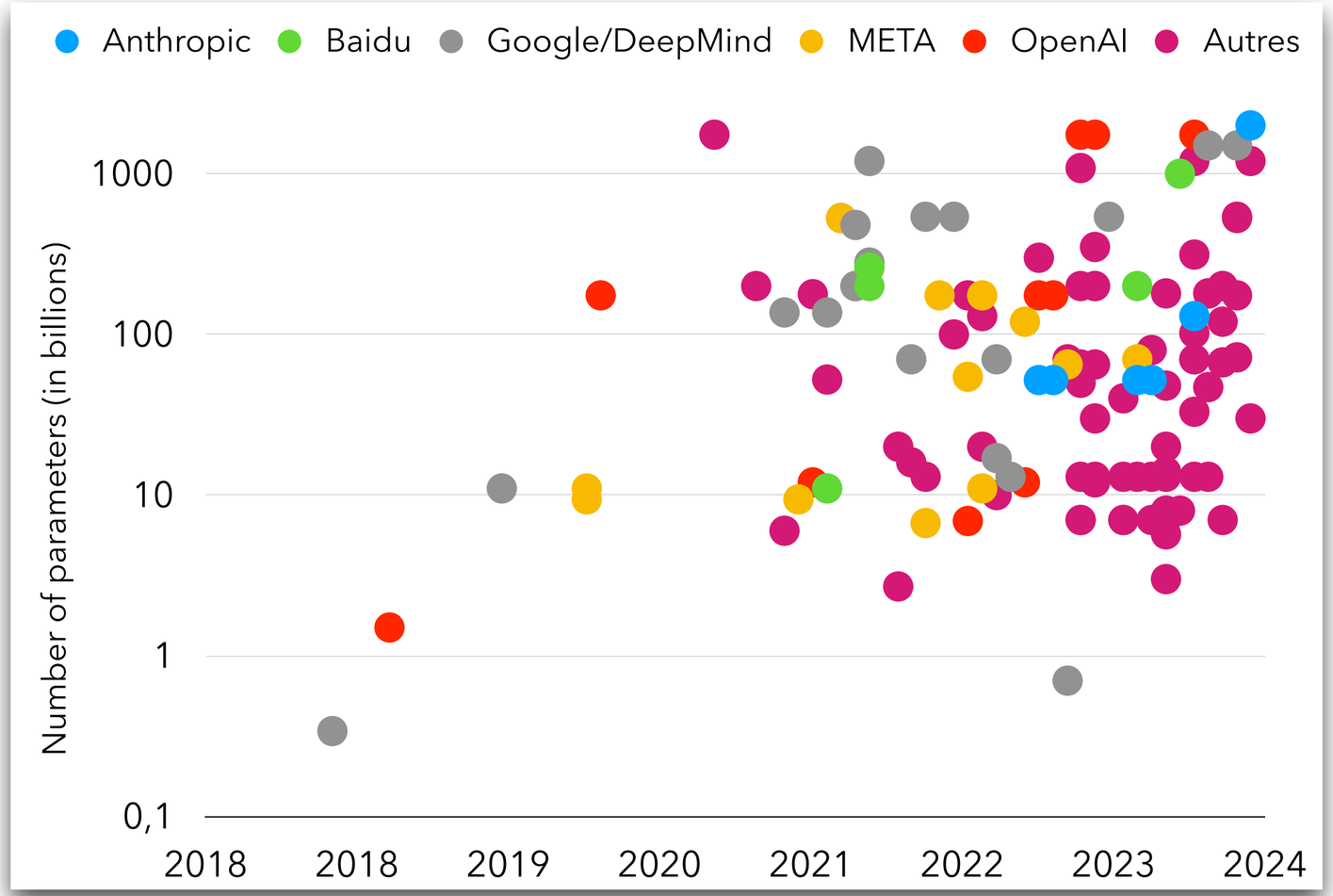
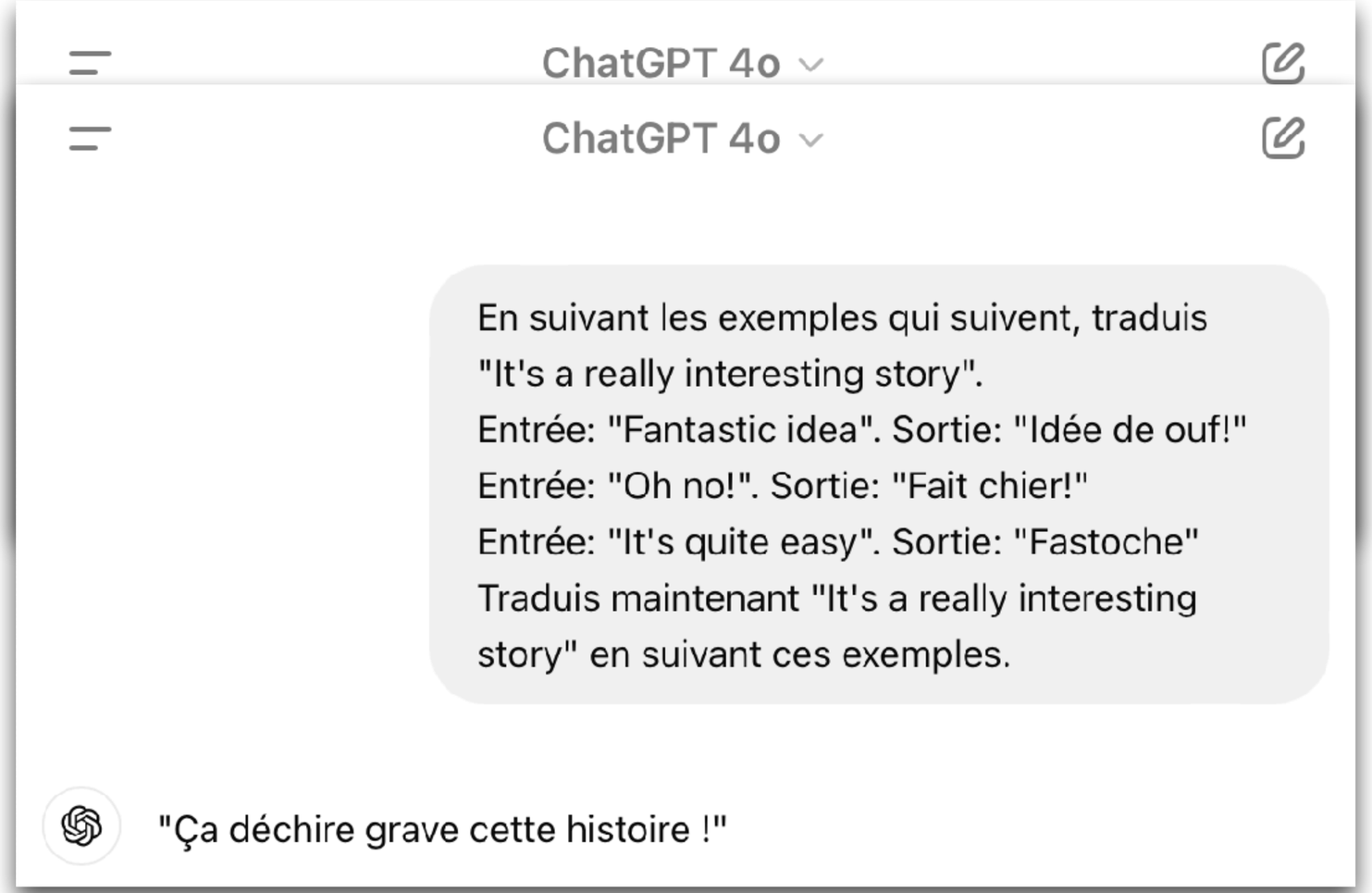


Figure based on data published by *Information is beautiful*, "The Rise and Rise of A.I. Large Language Models (LLMs)"

Using a conversational language model

- A conversational model can be used to … converse
- But also to perform tasks
  - By giving the model instructions on the task and then the input to process (**zero-shot**)
  - By also providing it with examples of input/output pairs (**few-shot**)



Conversation with ChatGPT (GPT-4o, 26/08/2024)

Advanced prompting

Chain-of-thought (Wei et al., 2022; also see Nye et al., 2021)

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

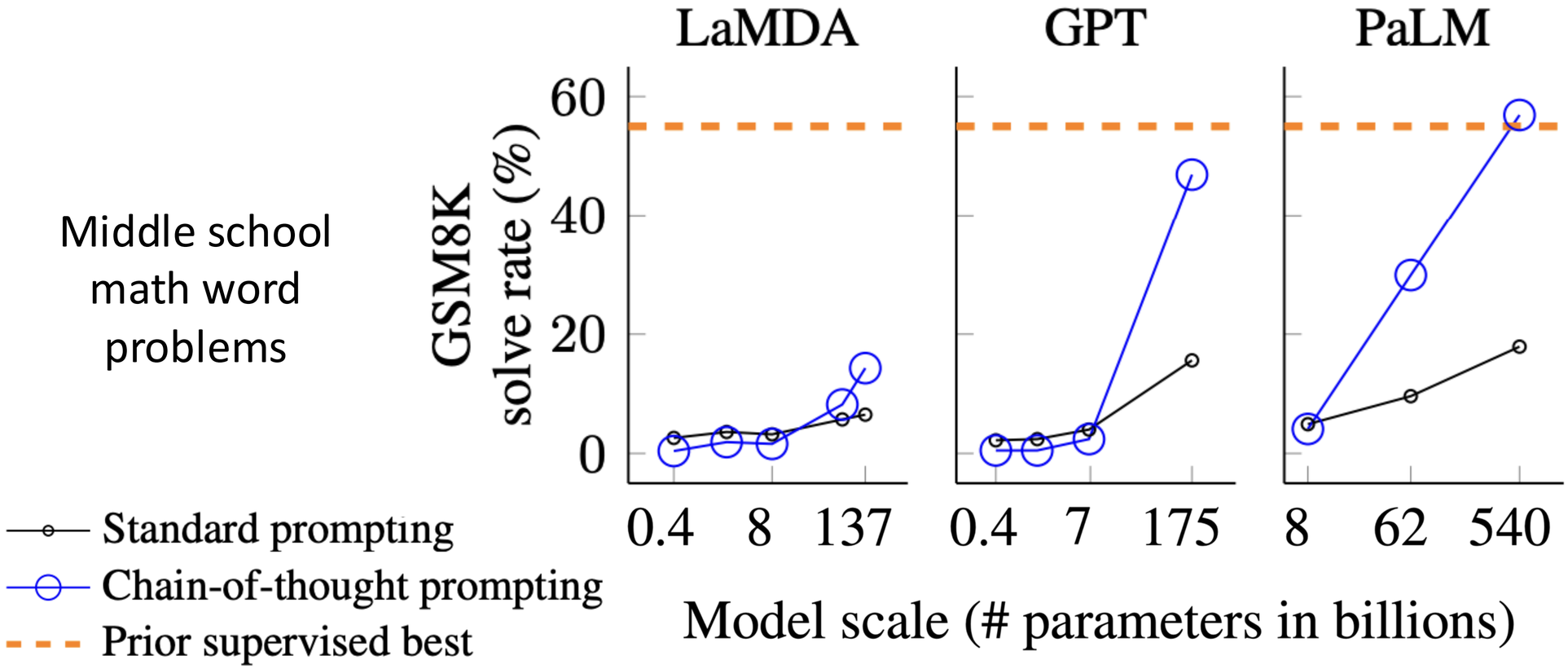
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Extracted from the slides for lecture 11 (given by Diyi Yang) of Stanford's 2025 NLP class (https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture11-adapatation.pdf)

Advanced prompting

Chain-of-thought (Wei et al., 2022; also see Nye et al., 2021)

Middle school math word problems



Extracted from the slides for lecture 11 (given by Diyi Yang) of Stanford's 2025 NLP class (https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture11-adapatation.pdf)

Advanced prompting

Zero-shot chain-of-thought (Kojima *et al.* 2022)

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many

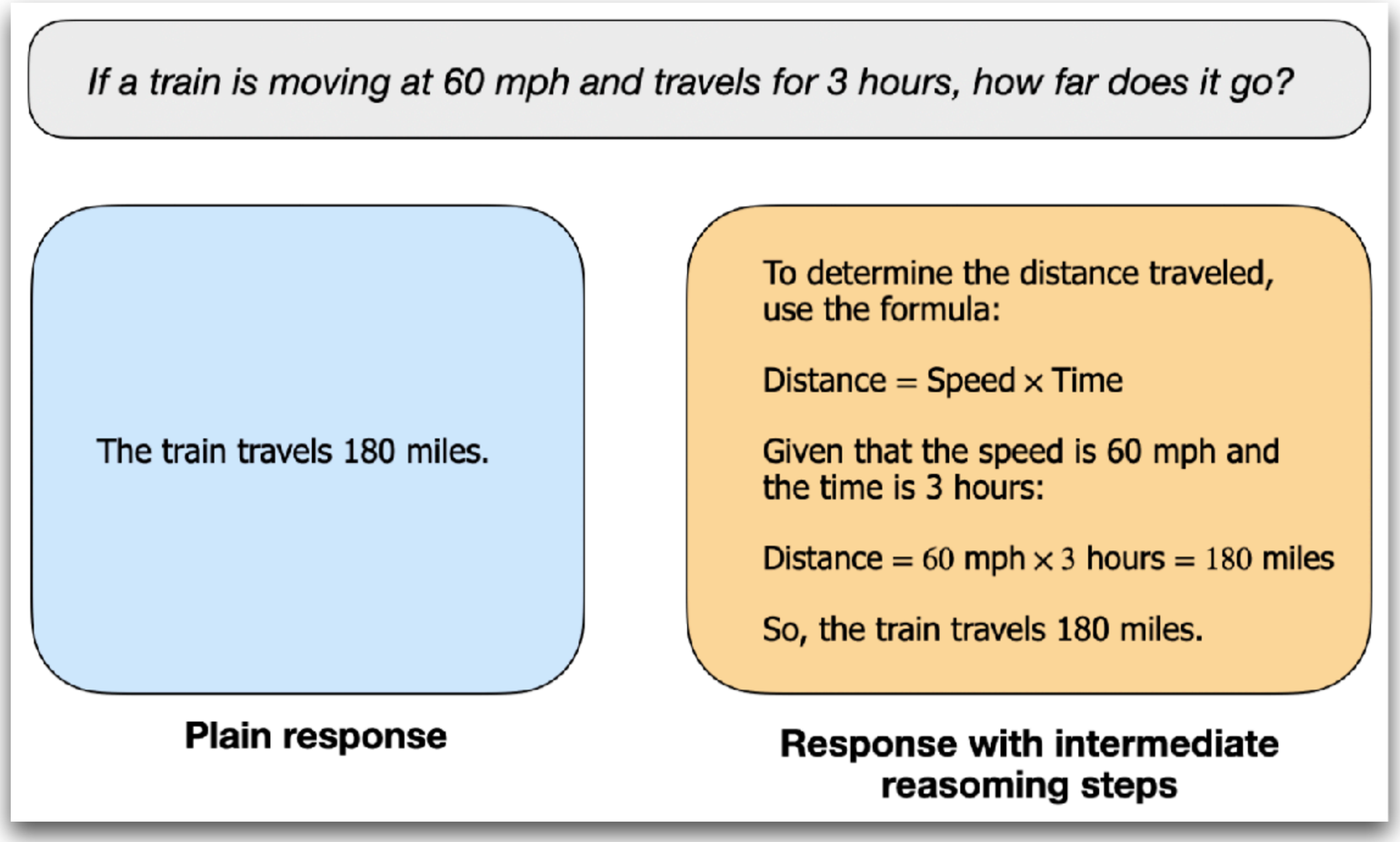
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: ***Let's think step by step.*** *There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls.* ✓

Extracted from the slides for lecture 11 (given by Diyi Yang) of Stanford's 2025 NLP class (https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture11-adapatation.pdf)

Reasoning models

Examples: o1, o3-mini, DeepSeek R1



Exemple extracted from https://sebastianraschka.com/blog/2025/understanding-reasoning-llms.html

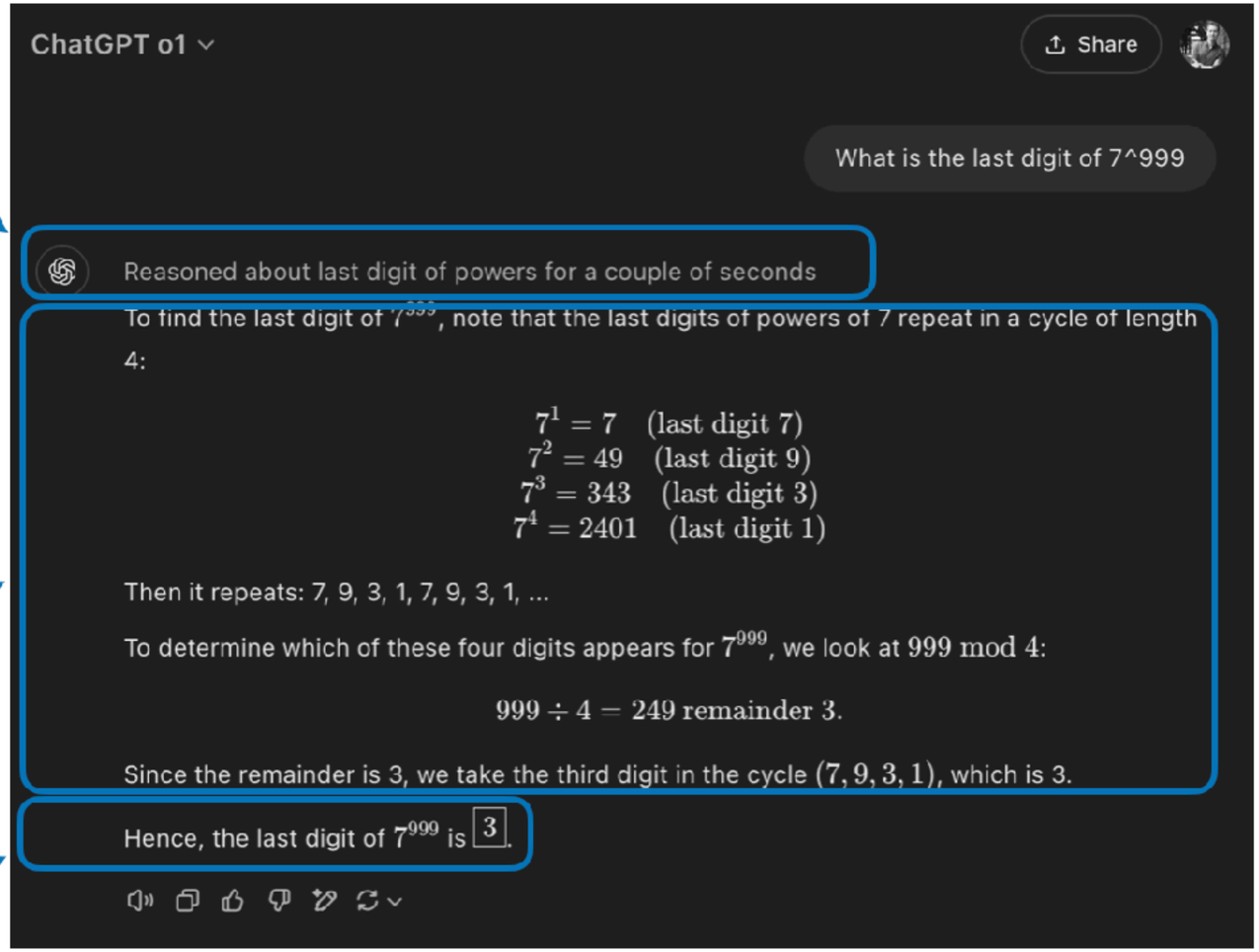
Reasoning models

Examples: o1, o3-mini, DeepSeek R1

1) Intermediate reasoning chain that is not explicitely shown to the user

2) Intermediate reasoning steps that are shown as part of the answer

The main answer

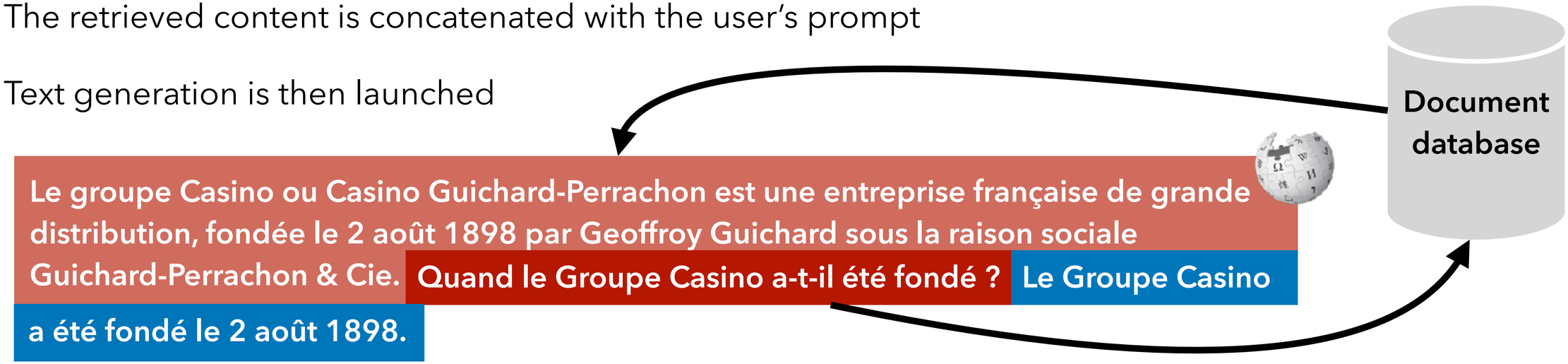


Exemple extracted from https://sebastianraschka.com/blog/2025/understanding-reasoning-llms.html

RAG

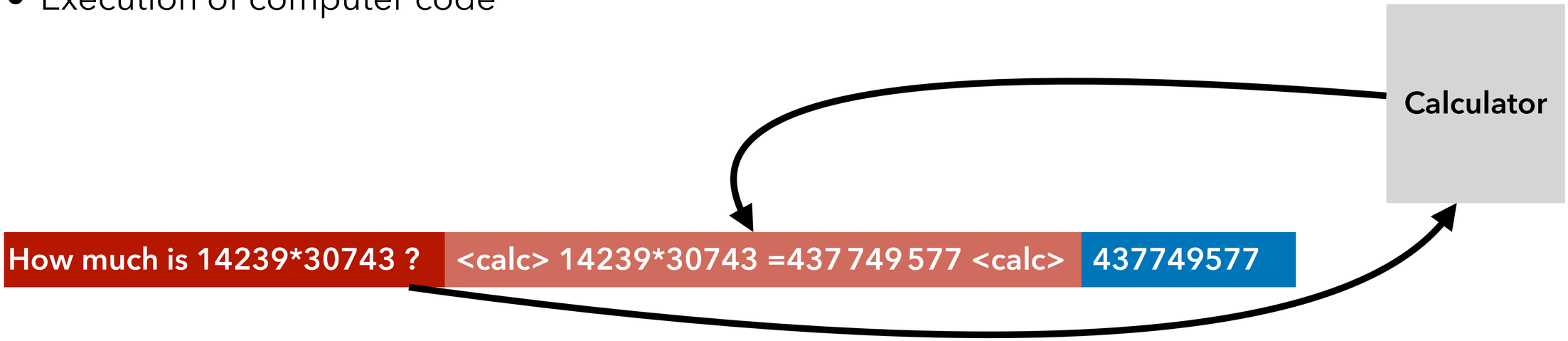
Retrieval-Augmented Generation (Lewis *et al.* 2020)

- Simple architecture to access external information (e.g. stored in a database):
  - The model decides, based on the prompt, whether it is relevant to retrieve documents from a knowledge base that may help produce a useful answer
  - If so, it sends a query to a document, paragraph, or information “retriever”
  - The retrieved content is concatenated with the user’s prompt
  - Text generation is then launched



Tooling

- **Tooling** (Schick et al. 2023)
  - Ability of a language model to call on an external tool to enrich its context with useful information
    - Calculator, web browsing, real-time updated databases, etc.
      - So RAG is a special case
    - Execution of computer code



Tooling

- **Tooling** (Schick et al. 2023)
  - Ability of a language model to call on an external tool to enrich its context with useful information
    - Calculator, web browsing, real-time updated databases, etc.
      - So RAG is a special case
    - Execution of computer code
- This allows a language model to behave like an “agent”
  - I.e. to call upon an ecosystem of “agents”

Agentic language models

Agentic AI: The Hype, The Reality, And The Playbook

A new AI buzzword is spreading fast—and it's leaving people with more questions than answers. "Agentic AI."

Is it a real breakthrough? Just a rebrand of automation? Or something in between?

Screenshots extracted from https://humanityatscale.substack.com/p/draft-agentic-ai-the-hype-the-reality