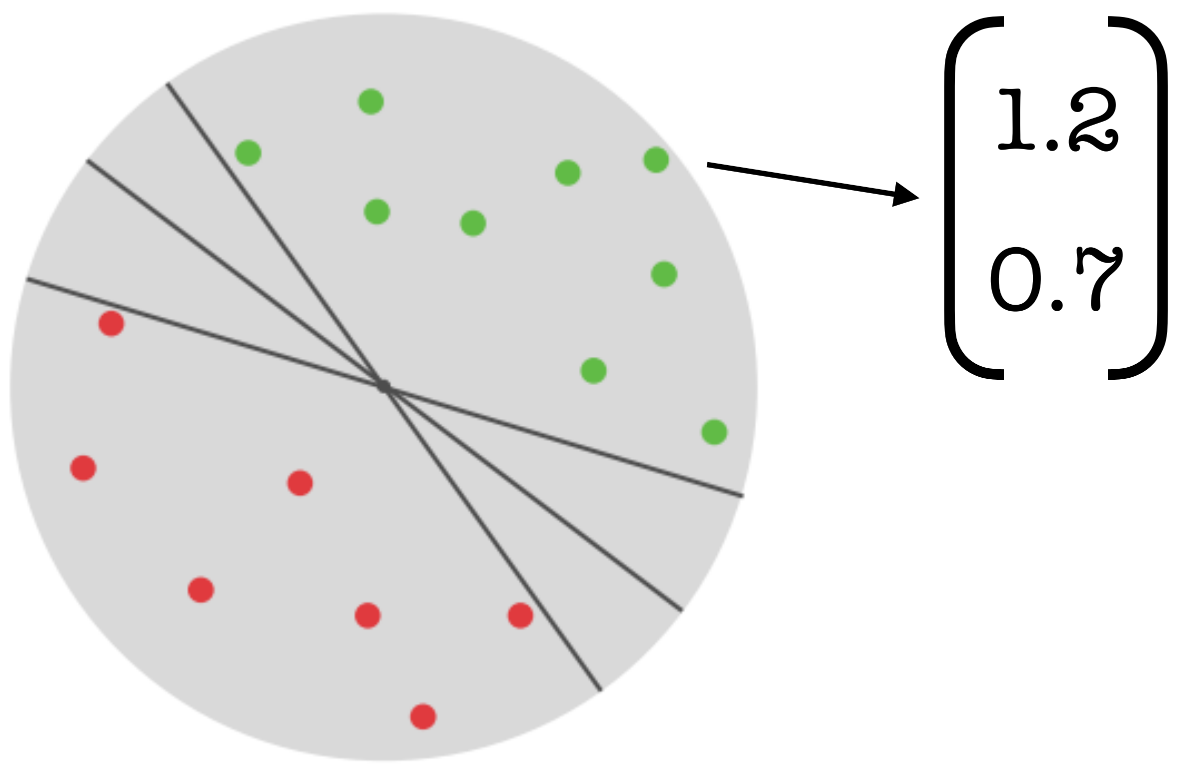
DAVID ALVAREZ-MELIS, MICROSOFT RESEARCH

CS 182 GUEST LECTURE: LANGUAGE MODELS AND NLP

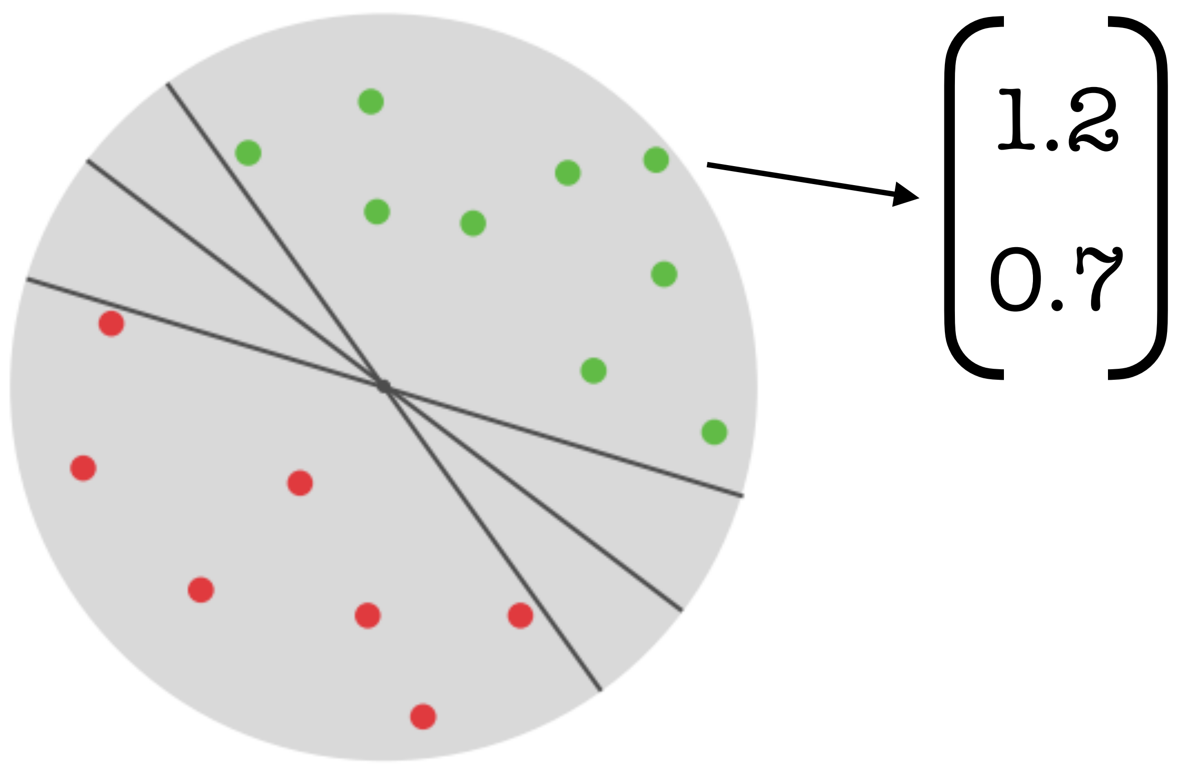
THE CHALLENGE WITH LANGUAGE

- So far: data has been assumed to be vectors:
  - fixed dimension
  - continuous



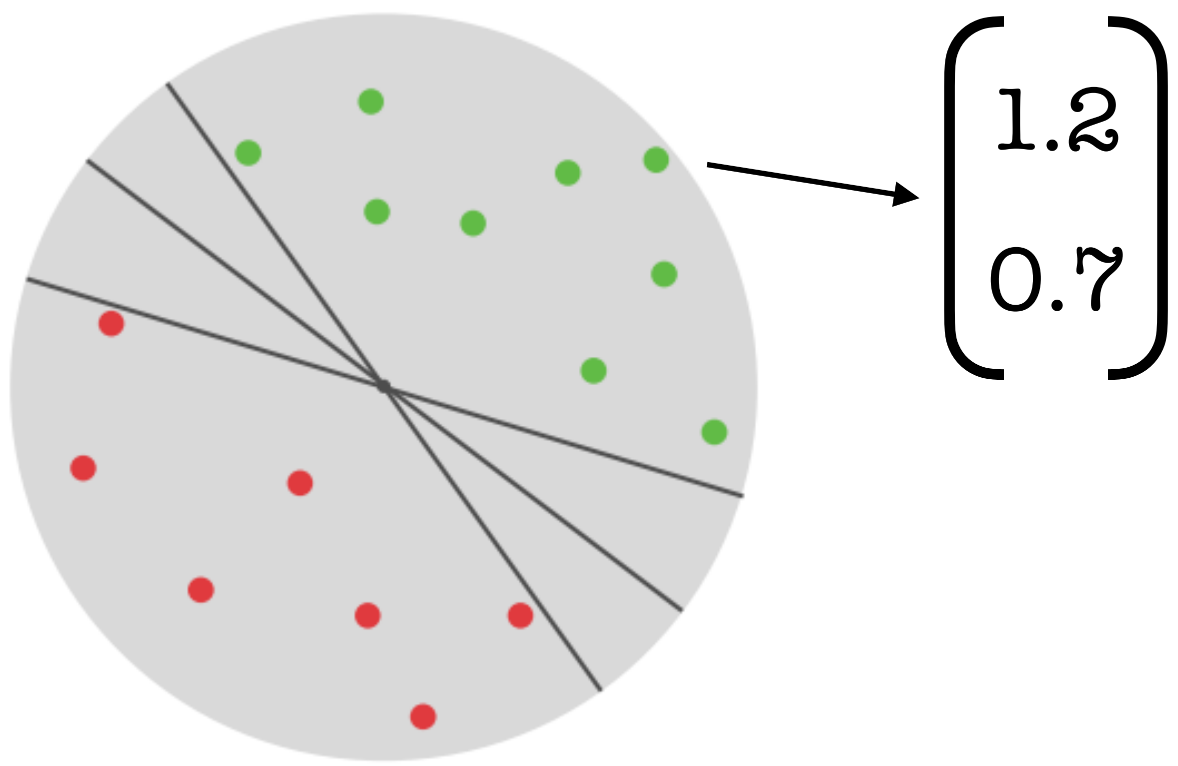
THE CHALLENGE WITH LANGUAGE

- So far: data has been assumed to be vectors:
  - fixed dimension
  - continuous
- What if the input is a sentence? Or a document?



THE CHALLENGE WITH LANGUAGE

- So far: data has been assumed to be vectors:
  - fixed dimension
  - continuous



- What if the input is a sentence? Or a document?

- Key questions:
  - how to **represent** text data while preserving its meaning
  - how to process it /**compute** with it efficiently

THE CHALLENGE WITH LANGUAGE

Natural Language Processing

THE CHALLENGE WITH LANGUAGE

$\underbrace{\text{Natural Language}}_{\text{i.e., not synthetic/constructed}}$ Processing

THE CHALLENGE WITH LANGUAGE

Natural Language Processing

i.e., not synthetic/constructed basically, "computing"

THE CHALLENGE WITH LANGUAGE

Natural Language Processing

THE CHALLENGE WITH LANGUAGE

Speech and Language Processing

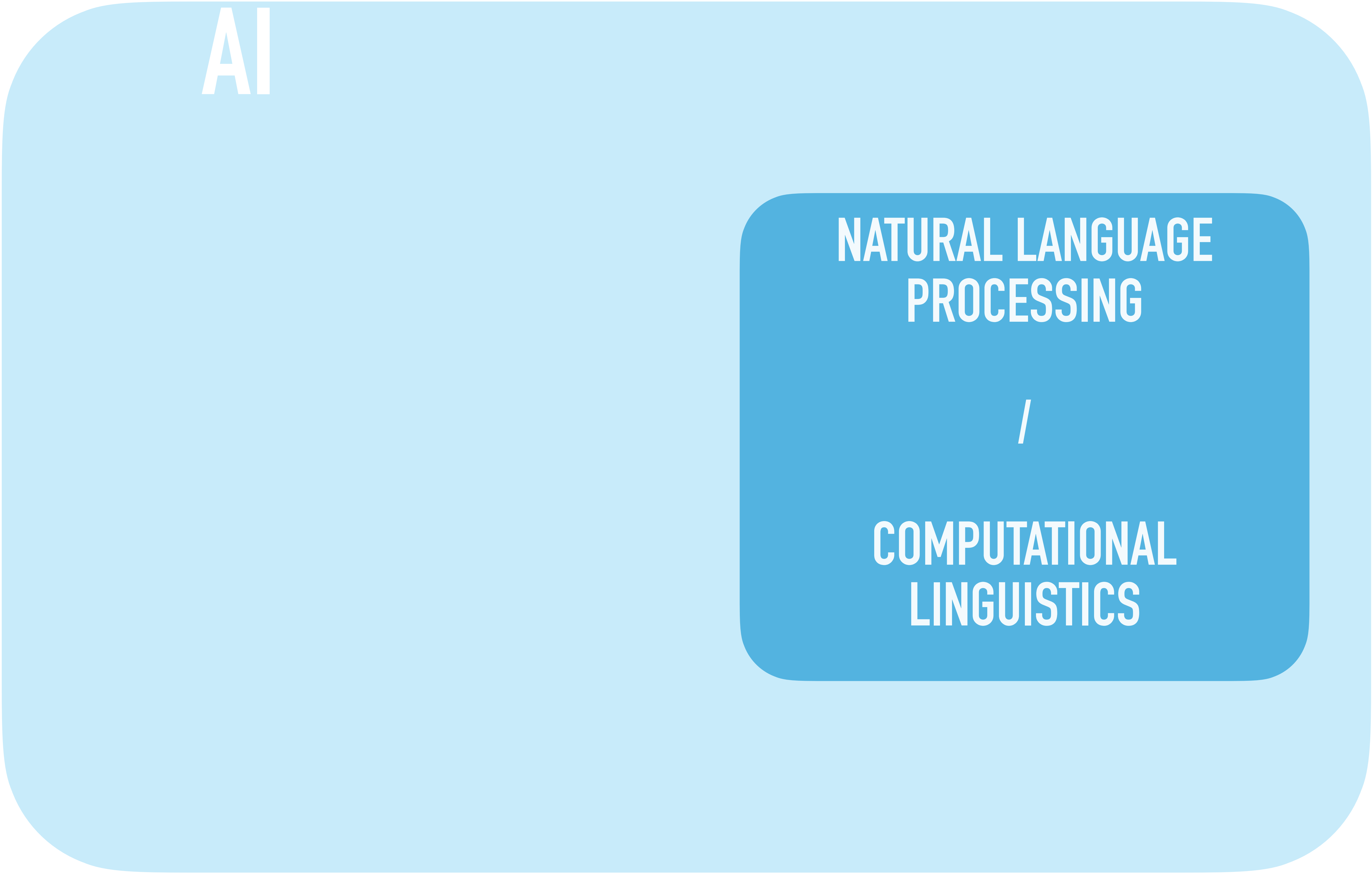
Human Language Technologies

Natural Language Processing

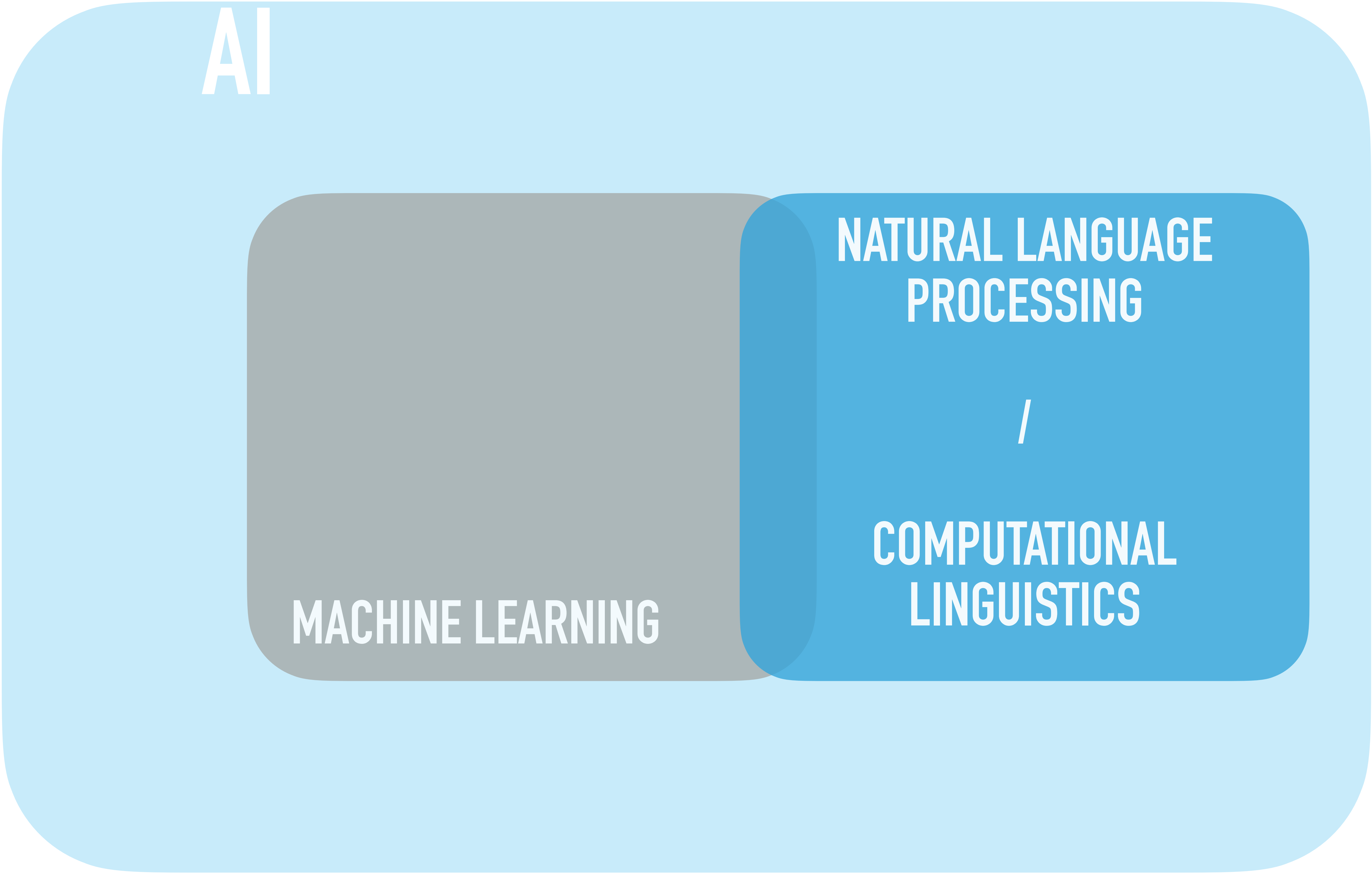
Natural Language Understanding

Computational Linguistics

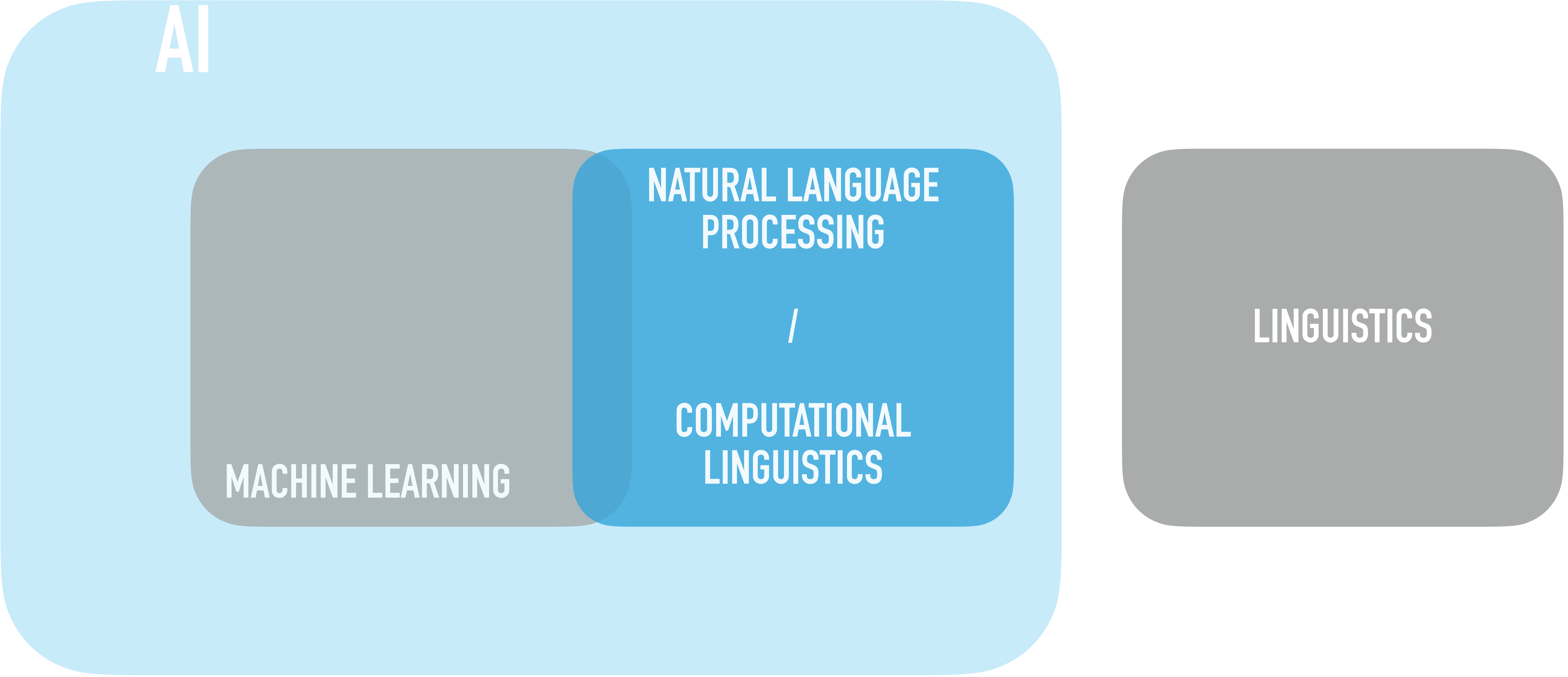
NATURAL LANGUAGE PROCESSING



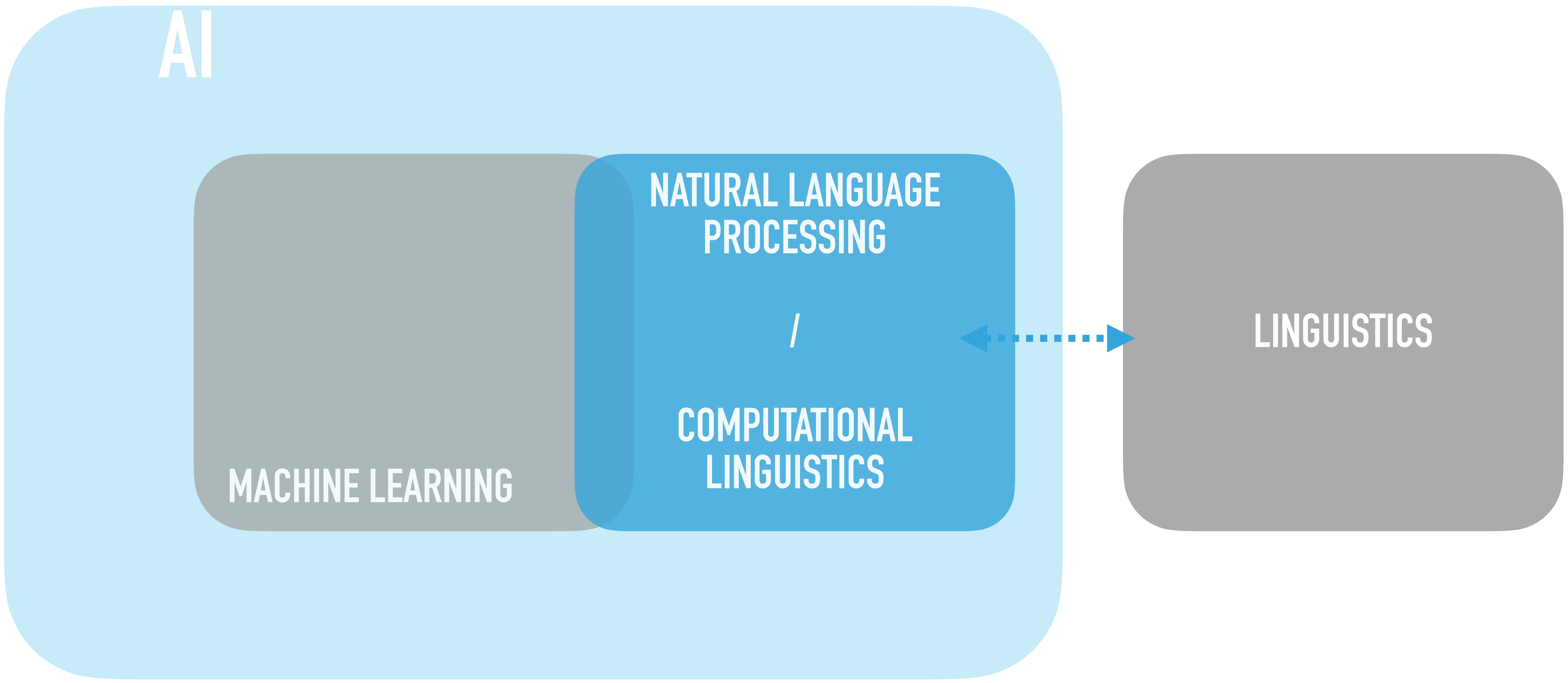
NATURAL LANGUAGE PROCESSING



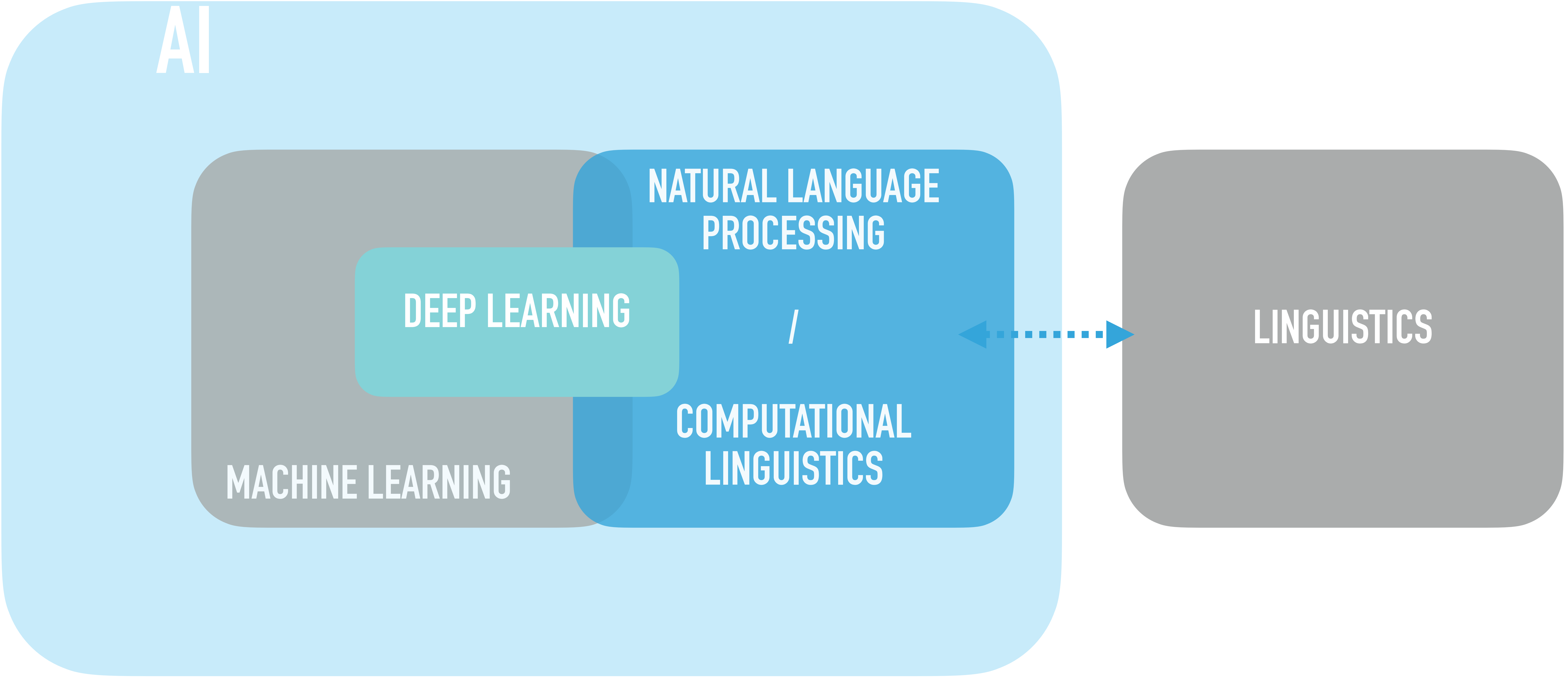
NATURAL LANGUAGE PROCESSING



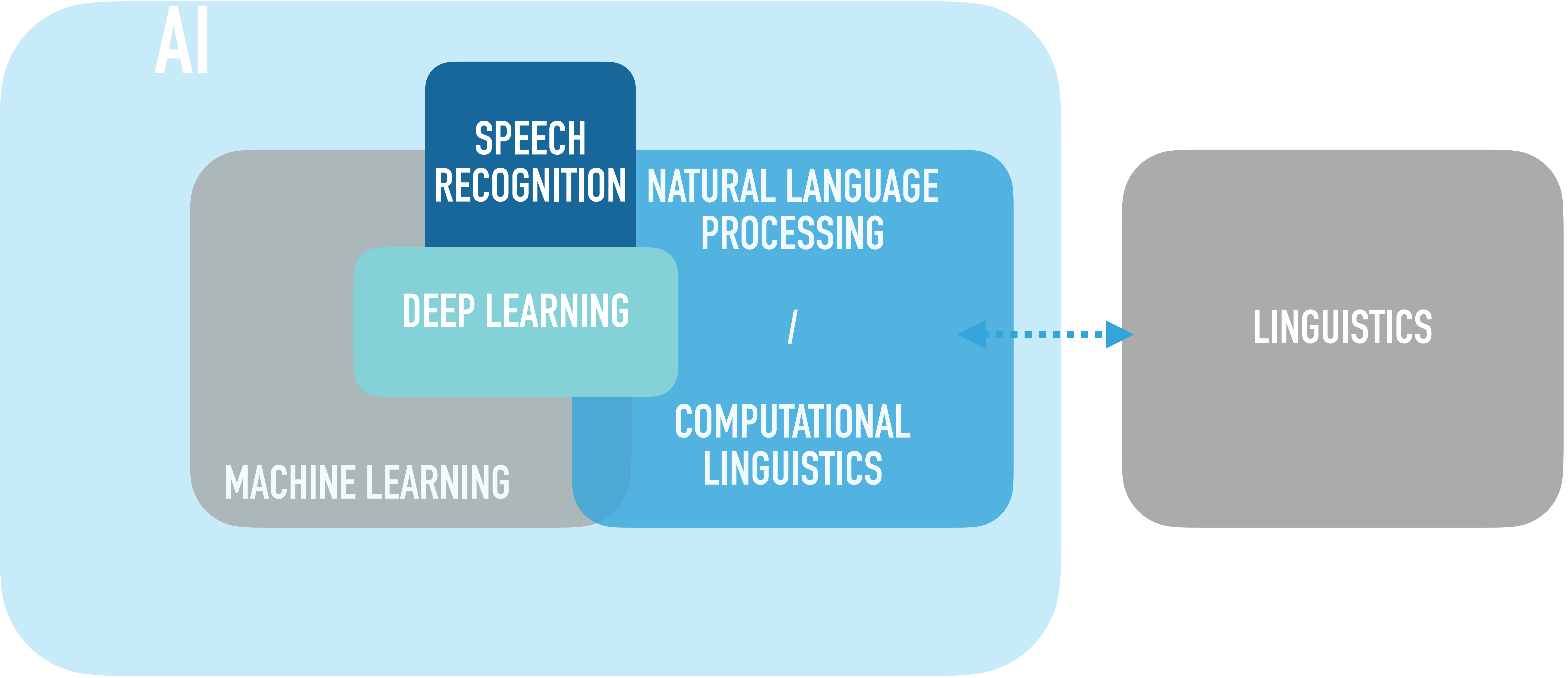
NATURAL LANGUAGE PROCESSING



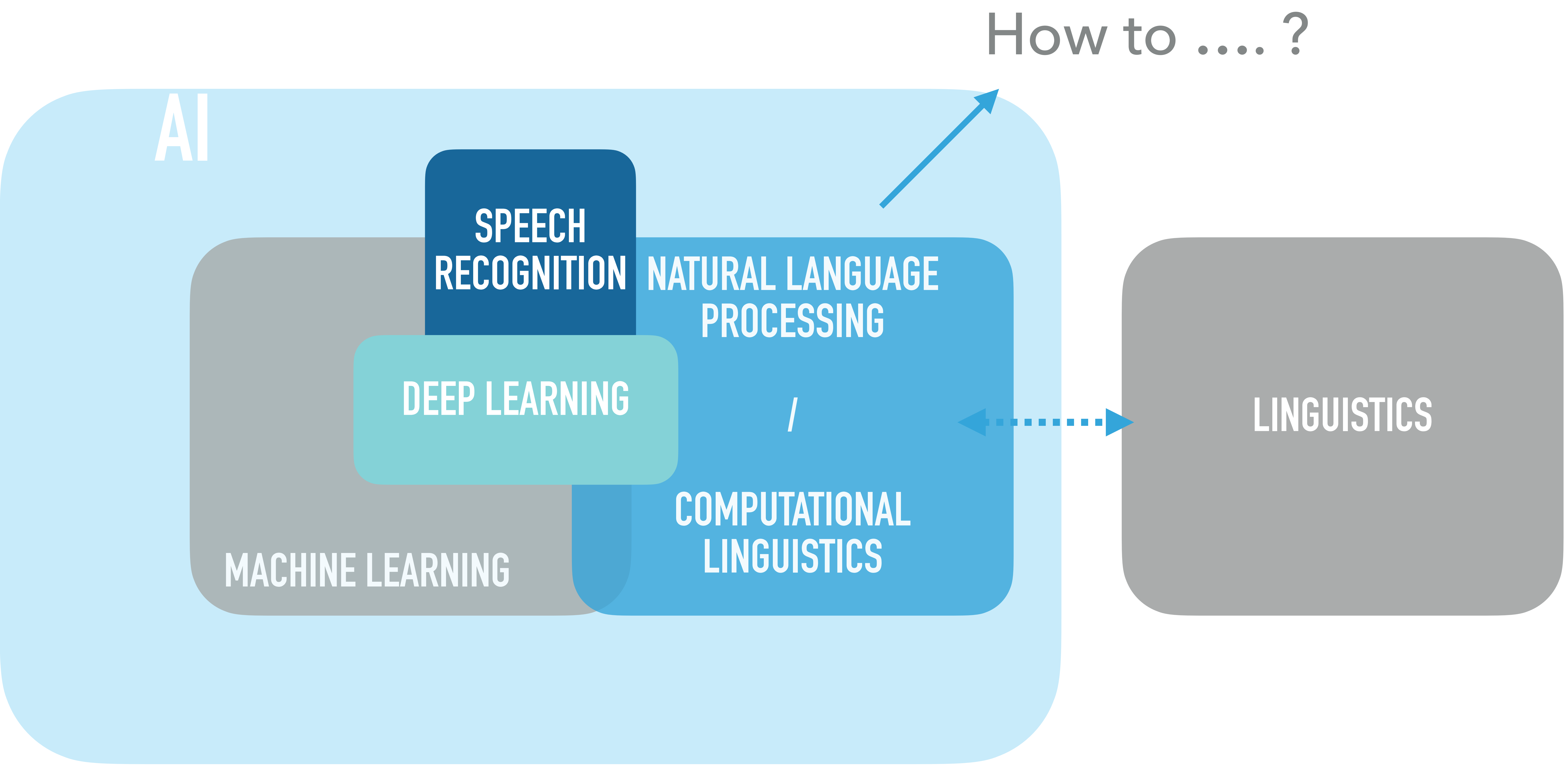
NATURAL LANGUAGE PROCESSING



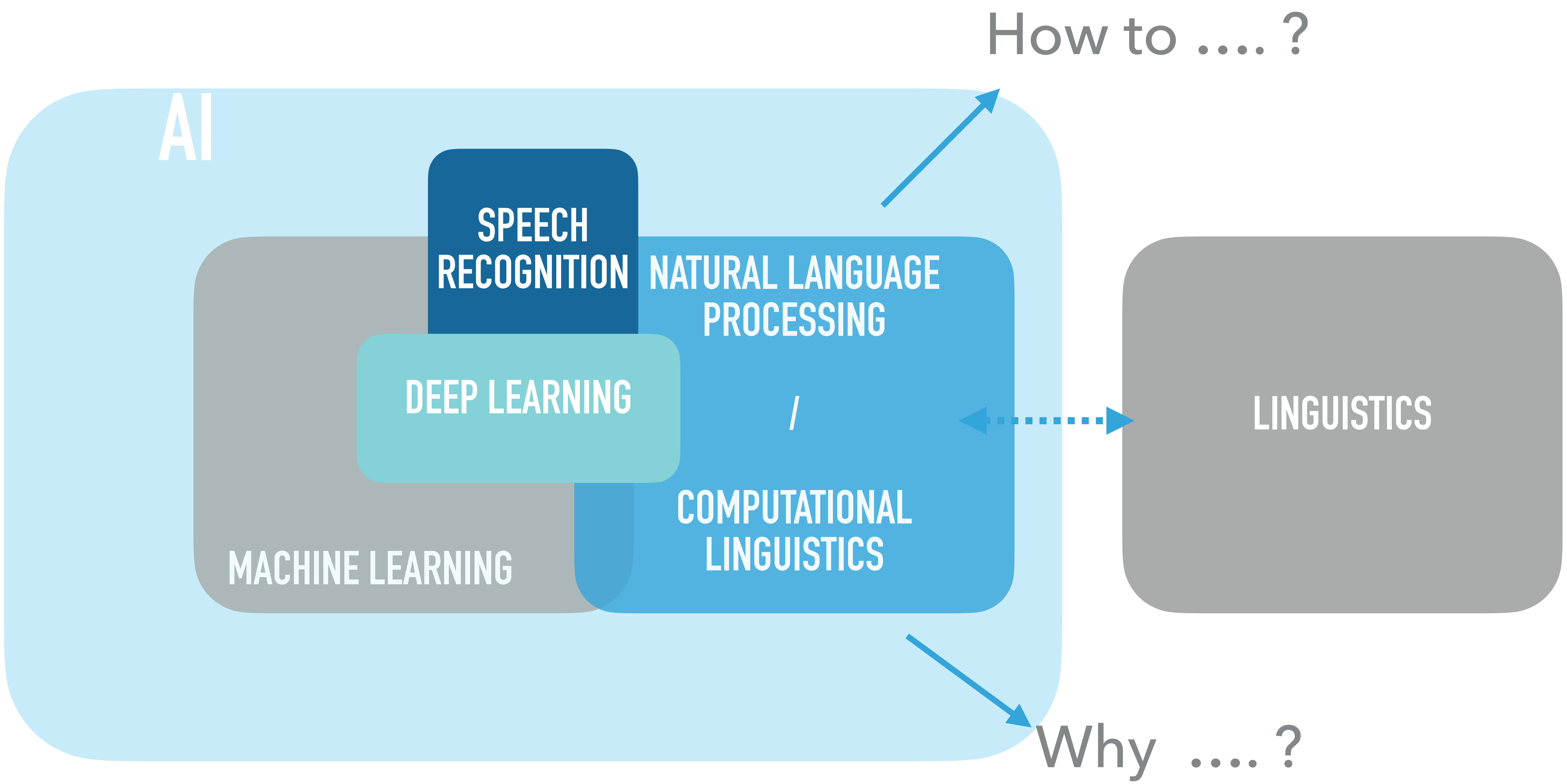
NATURAL LANGUAGE PROCESSING



NATURAL LANGUAGE PROCESSING



NATURAL LANGUAGE PROCESSING



NATURAL LANGUAGE PROCESSING

DIMENSIONS OF NLP

PROBLEMS	ASPECTS	METHODS
Machine translation	Semantics	Probabilistic
Summarization	Syntax	Symbolic
Text classification	Morphology	Bayesian
Parsing	Phonology	Kernel-Based
Language Modeling	Pragmatics	Deep Learning

LINGUISTICS CHEAT SHEET

LINGUISTICS CHEAT SHEET

- **Semantics:** pertaining to the meaning of a word, phrase, sentence, or text

se·man·tics

/səˈman(t)iks/

noun

- the meaning of a word, phrase, sentence, or text.

  plural noun: **semantics**

  "such quibbling over semantics may seem petty stuff"

LINGUISTICS CHEAT SHEET

- **Semantics:** pertaining to the meaning of a word, phrase, sentence, or text

- **Syntax:** arrangement of words and phrases to create wellformed sentences

se·man·tics

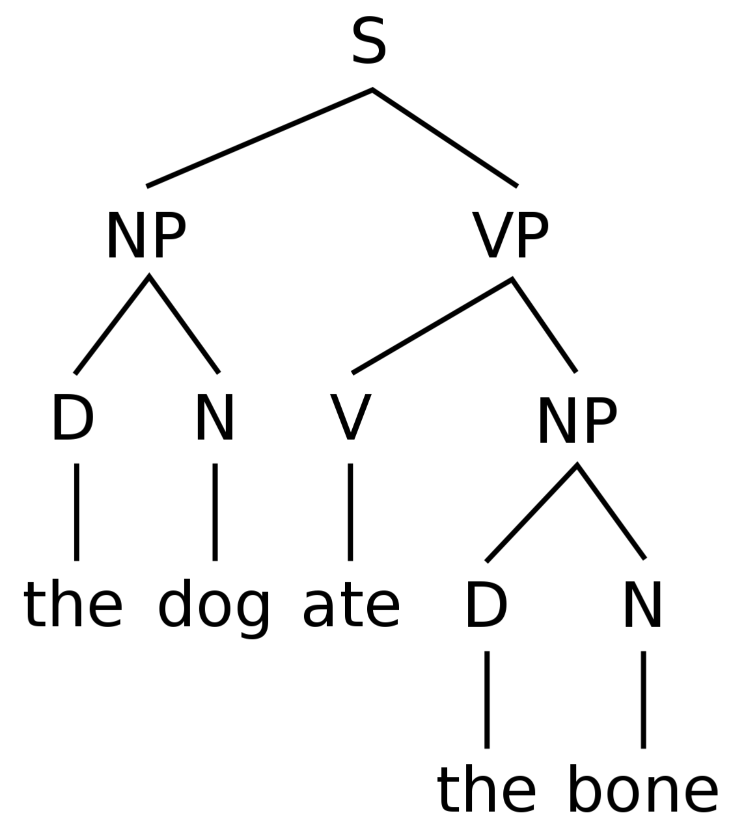
/səˈman(t)iks/

noun

- the meaning of a word, phrase, sentence, or text.

  plural noun: **semantics**

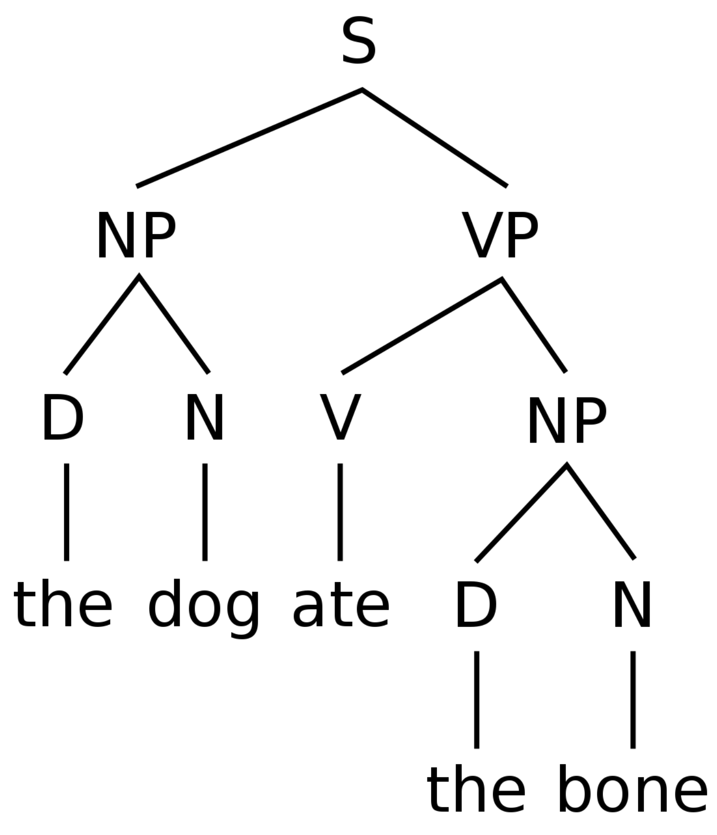
  "such quibbling over semantics may seem petty stuff"



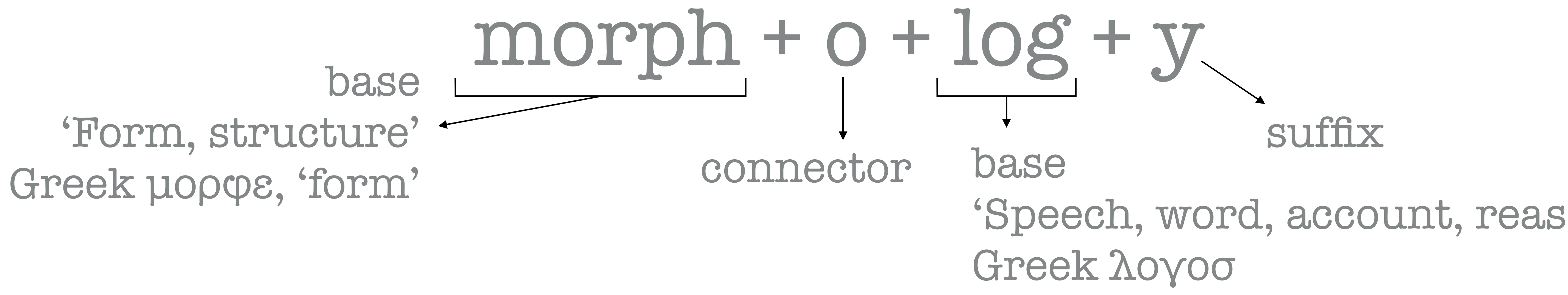
LINGUISTICS CHEAT SHEET

- **Semantics:** pertaining to the meaning of a word, phrase, sentence, or text

- **Syntax:** arrangement of words and phrases to create wellformed sentences

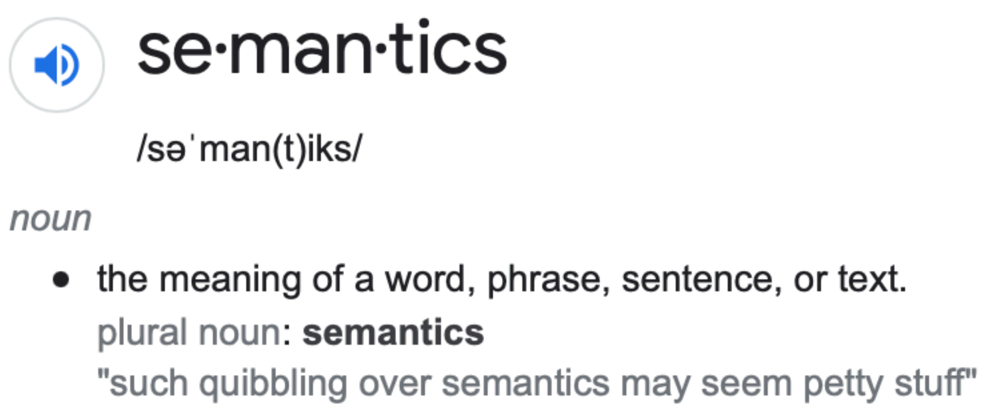


- **Morphology:** pertaining to the structure or form of words, e.g., their parts

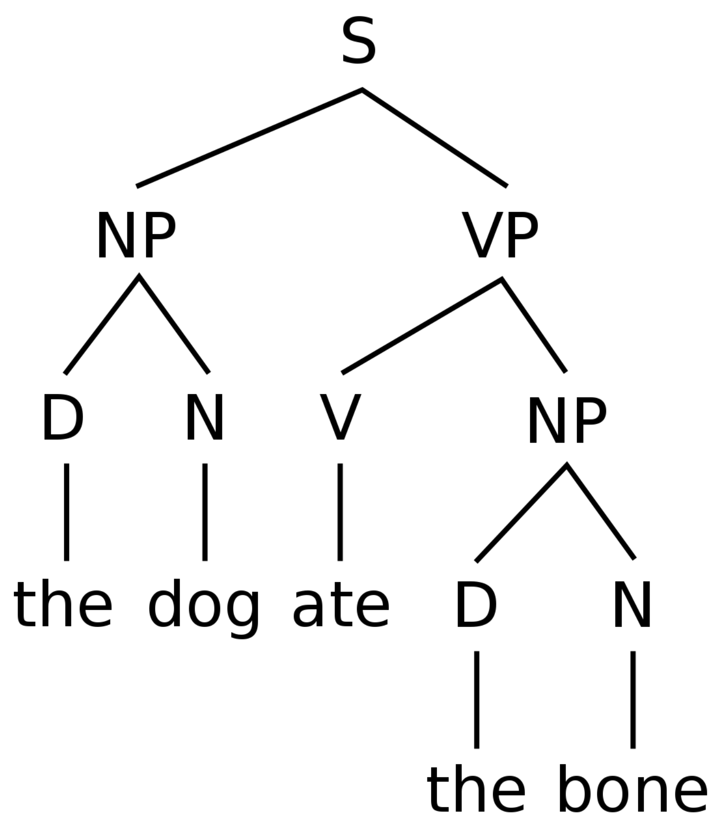


LINGUISTICS CHEAT SHEET

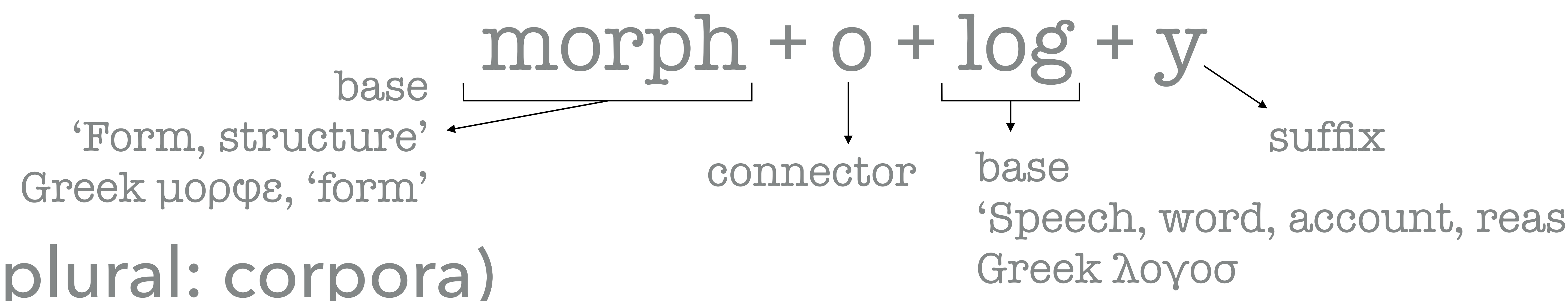
- **Semantics:** pertaining to the meaning of a word, phrase, sentence, or text



- **Syntax:** arrangement of words and phrases to create wellformed sentences



- **Morphology:** pertaining to the structure or form of words, e.g., their parts



- **Corpus:** a collection of text data (plural: corpora)

OUTLINE FOR TODAY

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP
- **Part I:** how do we encode meaning from text data (representation)

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP

- **Part I:** how do we encode meaning from text data (representation)
  - deep dive into word2vec for word embedding

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP

- **Part I:** how do we encode meaning from text data (representation)
  - deep dive into word2vec for word embedding

- **Part II:** how do we use encoded text to solve NLP tasks? (prediction)

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP
- **Part I:** how do we encode meaning from text data (representation)
  - deep dive into word2vec for word embedding
- **Part II:** how do we use encoded text to solve NLP tasks? (prediction)
  - deep dive into recurrent neural nets (vanilla and LSTM)

OUTLINE FOR TODAY

- **Goal:** overview of the main ideas and concepts behind modern NLP
- **Part I:** how do we encode meaning from text data (representation)
  - deep dive into word2vec for word embedding
- **Part II:** how do we use encoded text to solve NLP tasks? (prediction)
  - deep dive into recurrent neural nets (vanilla and LSTM)
- **Part III:** (time permitting) very large neural language models

PART 1:

ENCODING MEANING

THROUGH WORD EMBEDDINGS

WORD VECTOR REPRESENTATION: FIRST IDEA

Word representation:

house $= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \ \ldots]$

apartment $= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \ \ldots]$

nice $= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \ \ldots]$

WORD VECTOR REPRESENTATION: FIRST IDEA

each dimension
corresponds to a word

Word representation:

house $= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \ \ldots]$

apartment $= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \ \ldots]$

nice $= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \ \ldots]$

WORD VECTOR REPRESENTATION: FIRST IDEA

each dimension
corresponds to a word

Word representation:

house $= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \; \ldots]$

apartment $= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \; \ldots]$

nice $= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \; \ldots]$

Sentence/Document representation:

"the house is nice, the apartment is nice"

$$= [2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \; \ldots]$$

WORD VECTOR REPRESENTATION: FIRST IDEA

Word representation:

each dimension corresponds to a word

$$\begin{aligned} \text{house} &= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \; \ldots] \\ \text{apartment} &= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \; \ldots] \\ \text{nice} &= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \; \ldots] \end{aligned}$$

Sentence/Document representation:

“the house is nice, the apartment is nice”

$$= [2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \; \ldots]$$

Two crucial issues:

WORD VECTOR REPRESENTATION: FIRST IDEA

each dimension corresponds to a word

Word representation:

house $= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \ \ldots]$

apartment $= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \ \ldots]$

nice $= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \ \ldots]$

Sentence/Document representation:

"the house is nice, the apartment is nice"

$$= [2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \ \ldots]$$

Two crucial issues:

Vector size: # words in vocabulary (potentially huge)

WORD VECTOR REPRESENTATION: FIRST IDEA

each dimension corresponds to a word

Word representation:

house $= [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \ \ldots]$

apartment $= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \ \ldots]$

nice $= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \ \ldots]$

Sentence/Document representation:

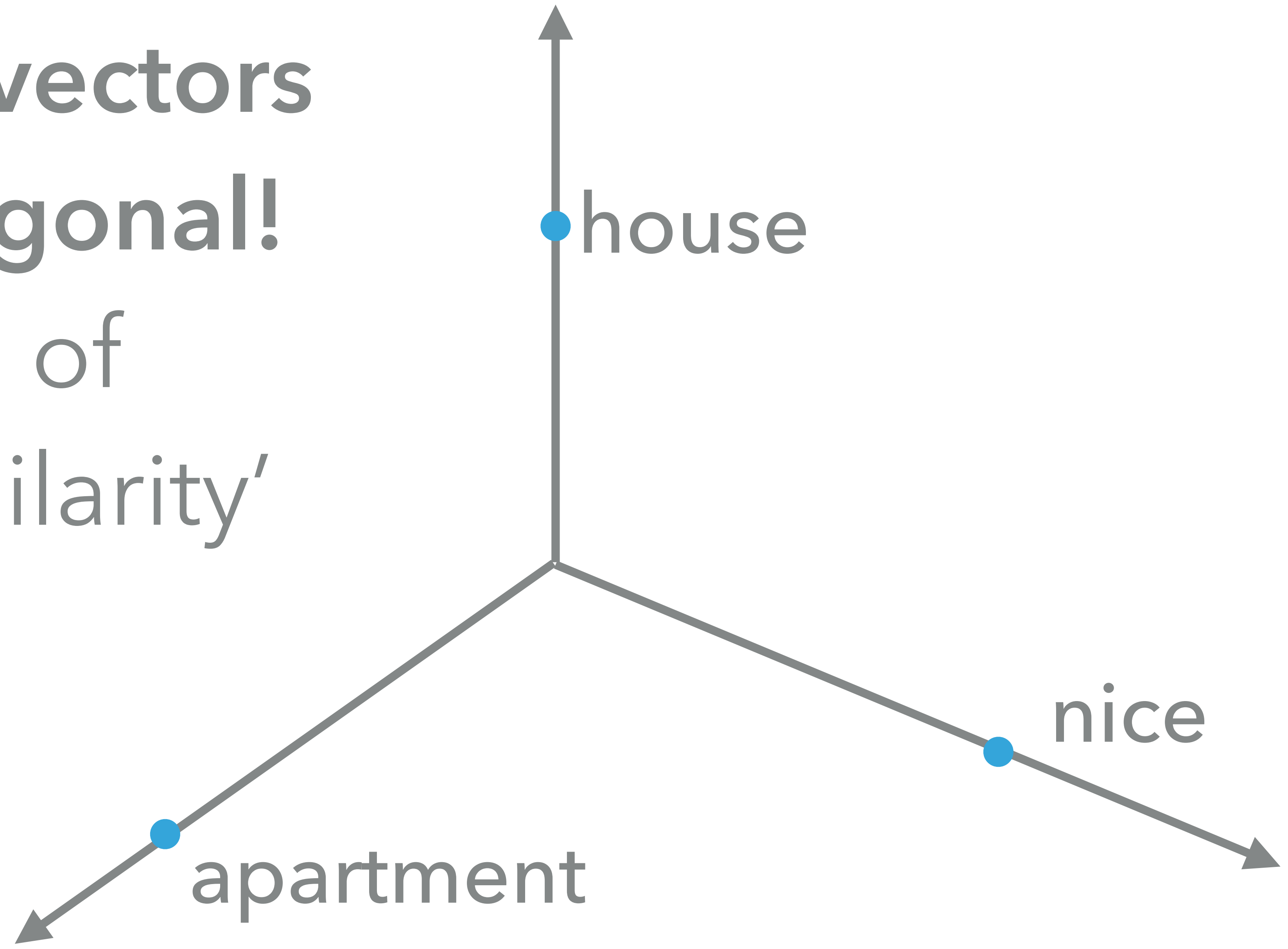
"the house is nice, the apartment is nice"

$$= [2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \ \ldots]$$

Two crucial issues:

Vector size: # words in vocabulary (potentially huge)

All word vectors are orthogonal!
no notion of word 'similarity'



WORD VECTOR REPRESENTATION

What we really want:

$$\begin{aligned}
\text{house} &= [0.23 \quad -1.52 \quad 3.22 \quad 0.01 \quad 2.45 \quad -1.32 \ \ldots] \\
\text{apartment} &= [-1.32 \quad 0.78 \quad 1.34 \quad 0.34 \quad -1.11 \quad 5.32 \ \ldots] \\
\text{nice} &= [0.98 \quad 0.32 \quad -3.34 \quad 8.23 \quad 1.01 \quad -2.68 \ \ldots]
\end{aligned}$$

WORD VECTOR REPRESENTATION

What we really want:

Vector size: fixed, not too large

house $= [0.23 \quad -1.52 \quad 3.22 \quad 0.01 \quad 2.45 \quad -1.32 \ \ldots]$

apartment $= [-1.32 \quad 0.78 \quad 1.34 \quad 0.34 \quad -1.11 \quad 5.32 \ \ldots]$

nice $= [0.98 \quad 0.32 \quad -3.34 \quad 8.23 \quad 1.01 \quad -2.68 \ \ldots]$

WORD VECTOR REPRESENTATION

What we really want:

Vector size: fixed, not too large

$$\text{house} = [0.23 \quad -1.52 \quad 3.22 \quad 0.01 \quad 2.45 \quad -1.32 \ \ldots]$$
$$\text{apartment} = [-1.32 \quad 0.78 \quad 1.34 \quad 0.34 \quad -1.11 \quad 5.32 \ \ldots]$$
$$\text{nice} = [0.98 \quad 0.32 \quad -3.34 \quad 8.23 \quad 1.01 \quad -2.68 \ \ldots]$$

The meaning of each word is '**distributed**' across many dimensions

WORD VECTOR REPRESENTATION

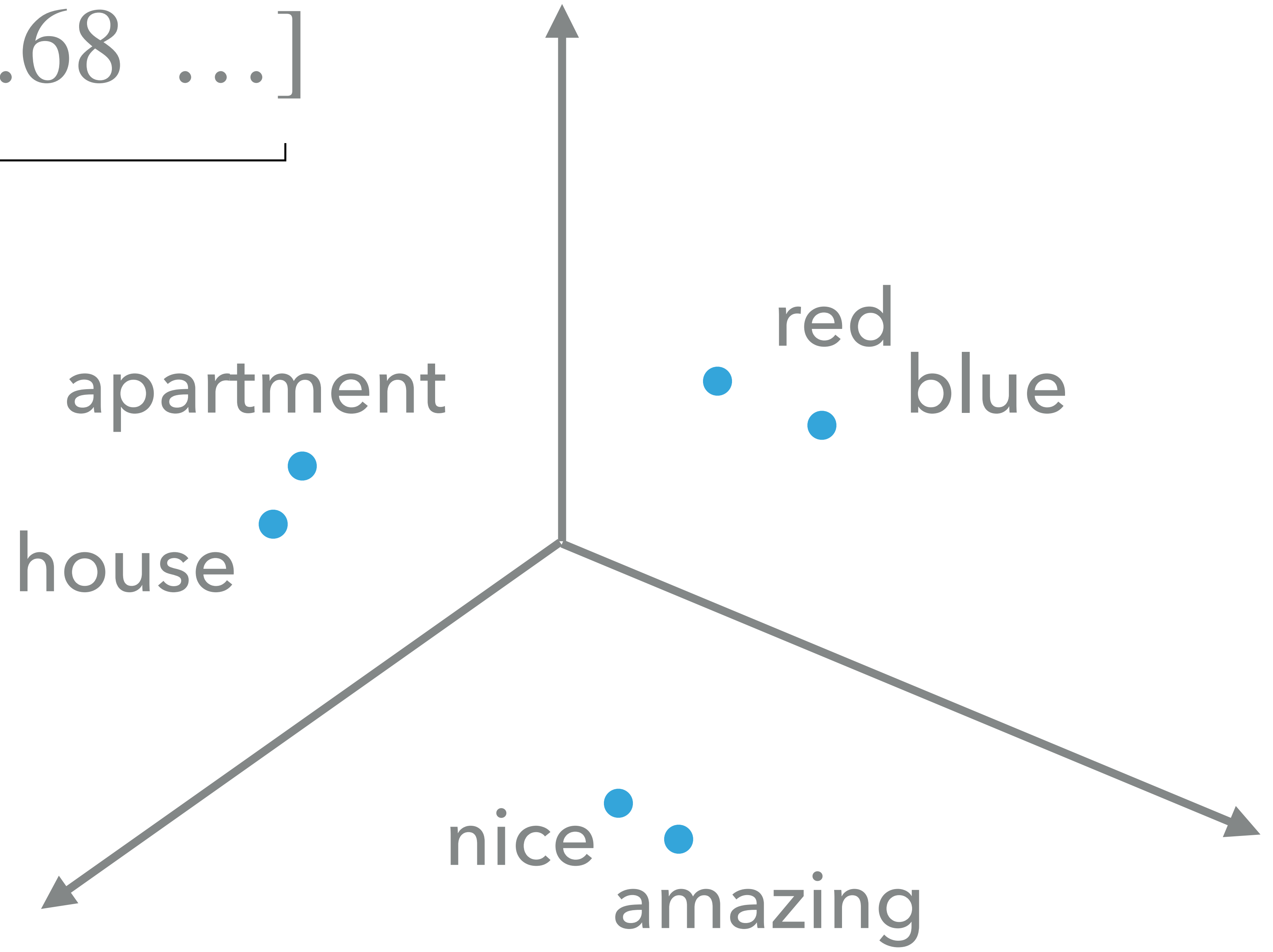
What we really want:

Vector size: fixed, not too large

$$\begin{aligned} \text{house} &= [0.23 \quad -1.52 \quad 3.22 \quad 0.01 \quad 2.45 \quad -1.32 \ \ldots] \\ \text{apartment} &= [-1.32 \quad 0.78 \quad 1.34 \quad 0.34 \quad -1.11 \quad 5.32 \ \ldots] \\ \text{nice} &= [0.98 \quad 0.32 \quad -3.34 \quad 8.23 \quad 1.01 \quad -2.68 \ \ldots] \end{aligned}$$

The meaning of each word is ‘**distributed**’ across many dimensions

Related words are closer together in vector space



WORD VECTOR REPRESENTATION

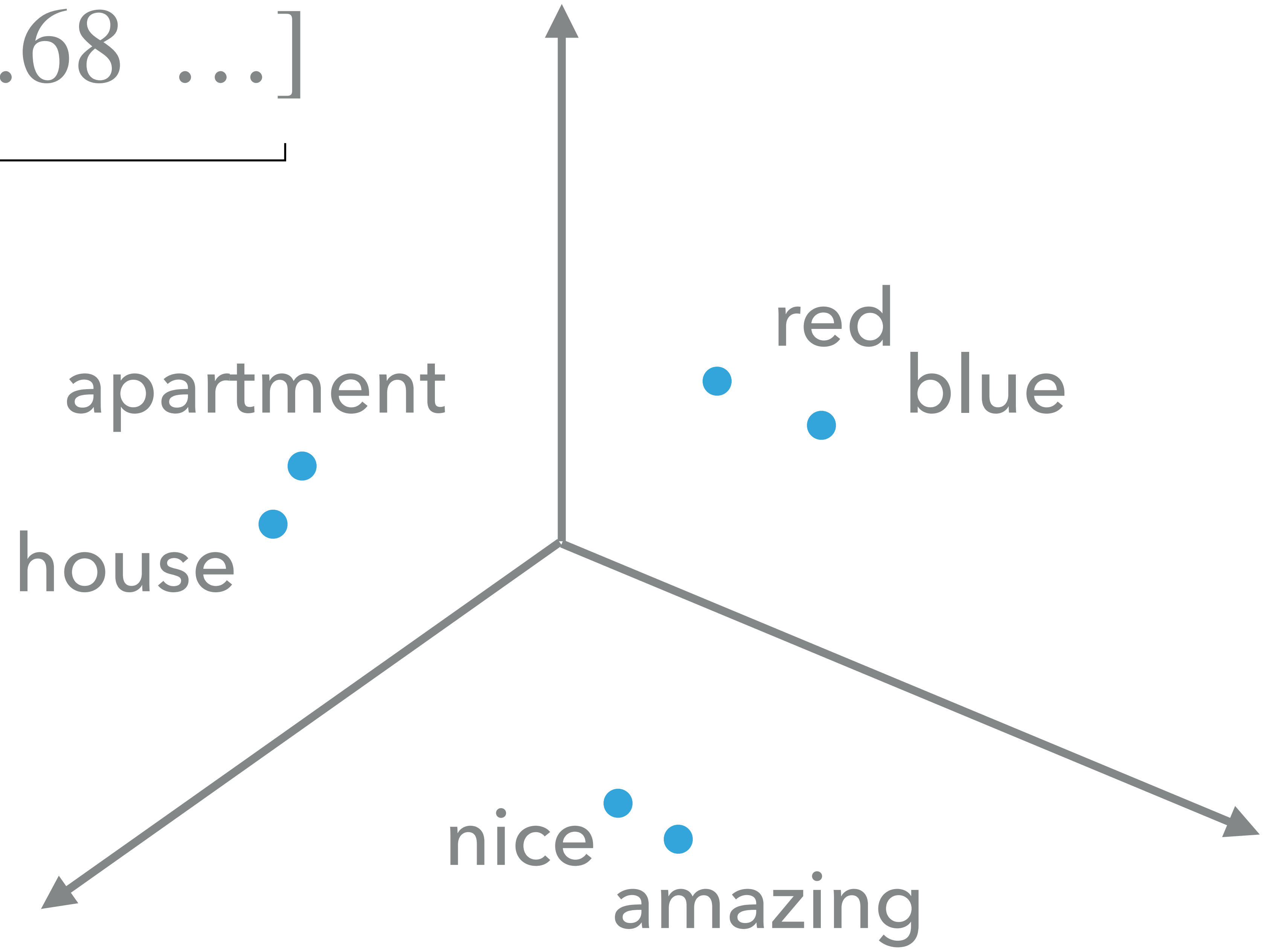
What we really want:

Vector size: fixed, not too large

$$\text{house} = [0.23 \quad -1.52 \quad 3.22 \quad 0.01 \quad 2.45 \quad -1.32 \ \ldots]$$
$$\text{apartment} = [-1.32 \quad 0.78 \quad 1.34 \quad 0.34 \quad -1.11 \quad 5.32 \ \ldots]$$
$$\text{nice} = [0.98 \quad 0.32 \quad -3.34 \quad 8.23 \quad 1.01 \quad -2.68 \ \ldots]$$

How do we achieve this?

How to turn  into $\begin{bmatrix}1.2\\0.7\\3.3\end{bmatrix}\begin{bmatrix}3.3\\1.5\\7.2\end{bmatrix}$ that carry **meaning**?



THE DISTRIBUTIONAL HYPOTHESIS

THE DISTRIBUTIONAL HYPOTHESIS

"YOU SHALL KNOW A WORD BY THE COMPANY IT KEEPS"

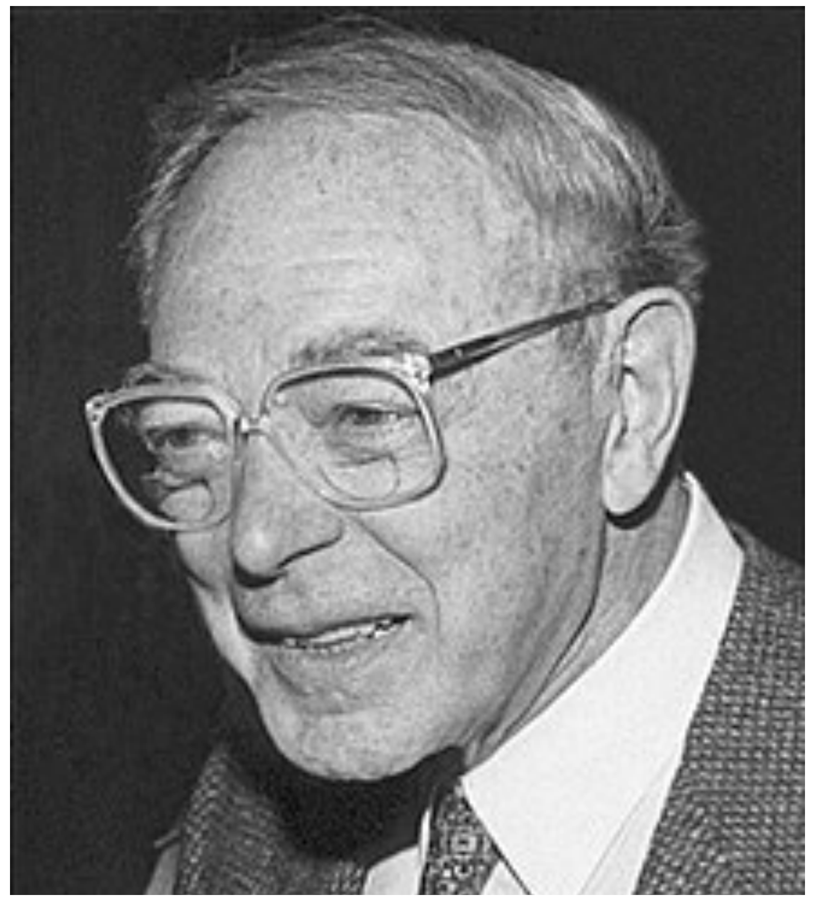
John R Firth (1957)

THE DISTRIBUTIONAL HYPOTHESIS

"YOU SHALL KNOW A WORD BY THE COMPANY IT KEEPS"

John R Firth (1957)

Zellig S Harris (1954)



"WORDS OCCURRING IN (LINGUISTICALLY) SIMILAR CONTEXTS TEND TO BE SEMANTICALLY SIMILAR"

THE DISTRIBUTIONAL HYPOTHESIS

What does

tezgüino mean?

[example from Lin (1998) via Eisenstein (2018)]

THE DISTRIBUTIONAL HYPOTHESIS

What does

tezgüino mean?

"A bottle of **tezgüino** is on the table."

"Everybody likes **tezgüino**."

"Don't have **tezgüino** before you drive."

"We make **tezgüino** out of corn."

[example from Lin (1998) via Eisenstein (2018)]

THE DISTRIBUTIONAL HYPOTHESIS

What does

tezgüino mean?

Tesgüino is an artisanal corn beer produced by several Yuto-Aztec people. The Tarahumara people regard the beer as sacred, and it forms a significant part of their society.

"A bottle of **tezgüino** is on the table."

"Everybody likes **tezgüino**."

"Don't have **tezgüino** before you drive."

"We make **tezgüino** out of corn."

[example from Lin (1998) via Eisenstein (2018)]

THE DISTRIBUTIONAL HYPOTHESIS

Idea: vectors of words appearing in similar contexts should be similar

THE DISTRIBUTIONAL HYPOTHESIS

Idea: vectors of words appearing in similar contexts should be similar

… central bank announced it will maintain interest rates fixed despite **inflation** fears in the economy …

THE DISTRIBUTIONAL HYPOTHESIS

Idea: vectors of words appearing in similar contexts should be similar

… central bank announced it will maintain interest rates fixed despite **inflation** fears in the economy …

… interest rates continued increasing, along with the consumer **price** index, while the US economy…

THE DISTRIBUTIONAL HYPOTHESIS

Idea: vectors of words appearing in similar contexts should be similar

… central bank announced it will maintain interest rates fixed despite **inflation** fears in the economy …

… interest rates continued increasing, along with the consumer **price** index, while the US economy…

THE DISTRIBUTIONAL HYPOTHESIS

Idea: vectors of words appearing in similar contexts should be similar

… central bank announced it will maintain interest rates fixed despite **inflation** fears in the economy …

$$x_{\text{inflation}} \leftrightarrow x_{\text{price}}$$

… interest rates continued increasing, along with the consumer **price** index, while the US economy…

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \text{context}_i(w))$$

APPROACH 2: PREDICT

$$p(w' \mid w)$$

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \mathsf{context}_i(w))$$

Latent Semantic Indexing (LSI)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \mathsf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \textbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \mathbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \textbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

Word2vec (2 flavors)

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \textbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

Word2vec (2 flavors)

GloVe

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \textsf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

Word2vec (2 flavors)

GloVe

FastText

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \textbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

Word2vec (2 flavors)

GloVe

FastText

…

ALGORITHMS FOR ENCODING CO-OCCURRENCE INFORMATION

APPROACH 1: COUNT

$$\sum I(w' \in \mathbf{context}_i(w))$$

Latent Semantic Indexing (LSI)

Hyperspace Analogue to Language (HAL)

Pointwise Mutual Informacion (PMI)

Canonical Correlation Analysis (CCA)

APPROACH 2: PREDICT

$$p(w' \mid w)$$

Word2vec (2 flavors)

GloVe

FastText

…

WORD2VEC [Mikolov et al., 2013; 2014]

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013; 2014]

- Model probability of context given center word

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013; 2014]

- Model probability of context given center word
- Parametrize as neural network

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013; 2014]

- Model probability of context given center word
- Parametrize as neural network
- Train via maximum likelihood objective

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013; 2014]

- Model probability of context given center word
- Parametrize as neural network
- Train via maximum likelihood objective
- Efficient training via SGD + Negative Sampling

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013; 2014]

- Model probability of context given center word
- Parametrize as neural network
- Train via maximum likelihood objective
- Efficient training via SGD + Negative Sampling
- Fascinating linear relationships in vector space

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

>50K combined citations!

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

WORD2VEC [Mikolov et al., 2013]

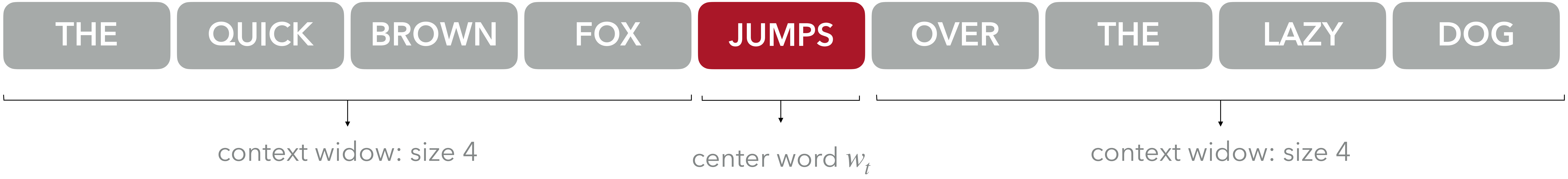
THE QUICK BROWN FOX **JUMPS** OVER THE LAZY DOG

WORD2VEC [Mikolov et al., 2013]

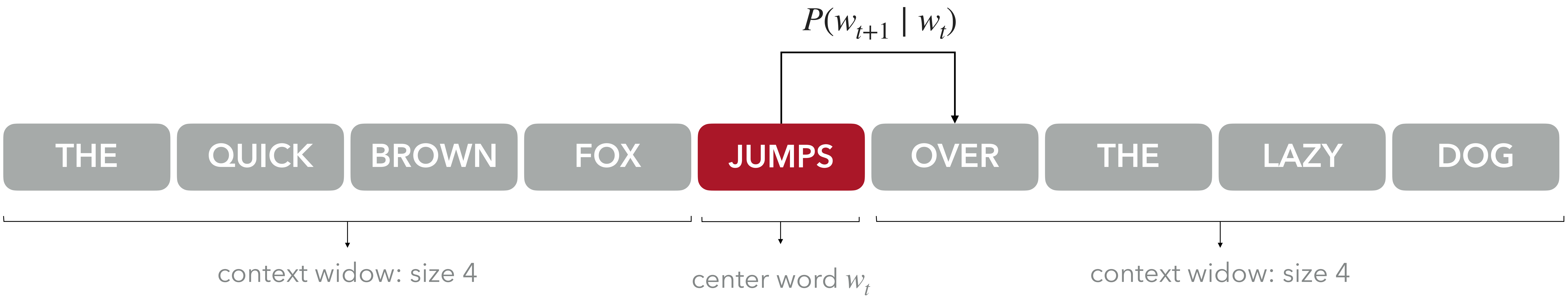
THE QUICK BROWN FOX **JUMPS** OVER THE LAZY DOG

center word w_t

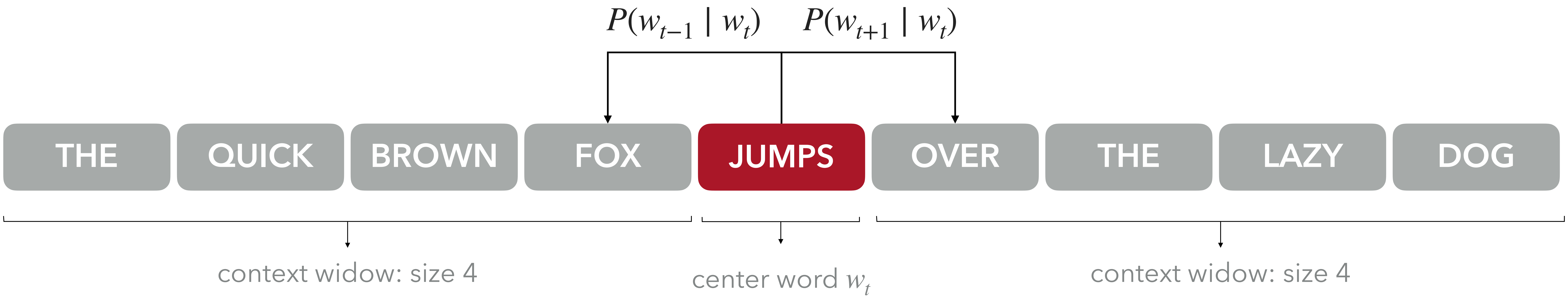
WORD2VEC [Mikolov et al., 2013]



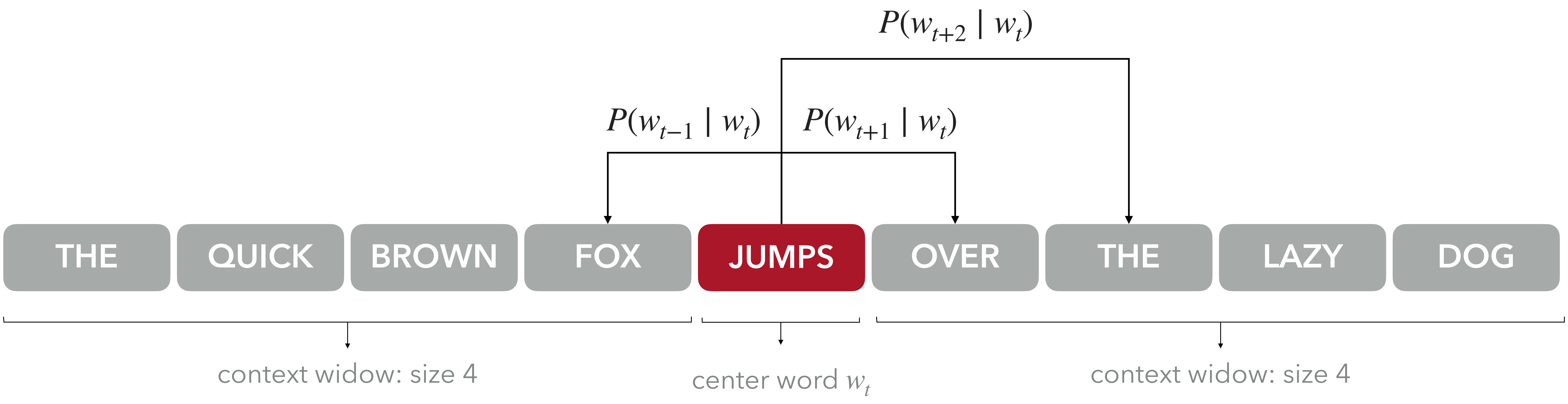
WORD2VEC [Mikolov et al., 2013]



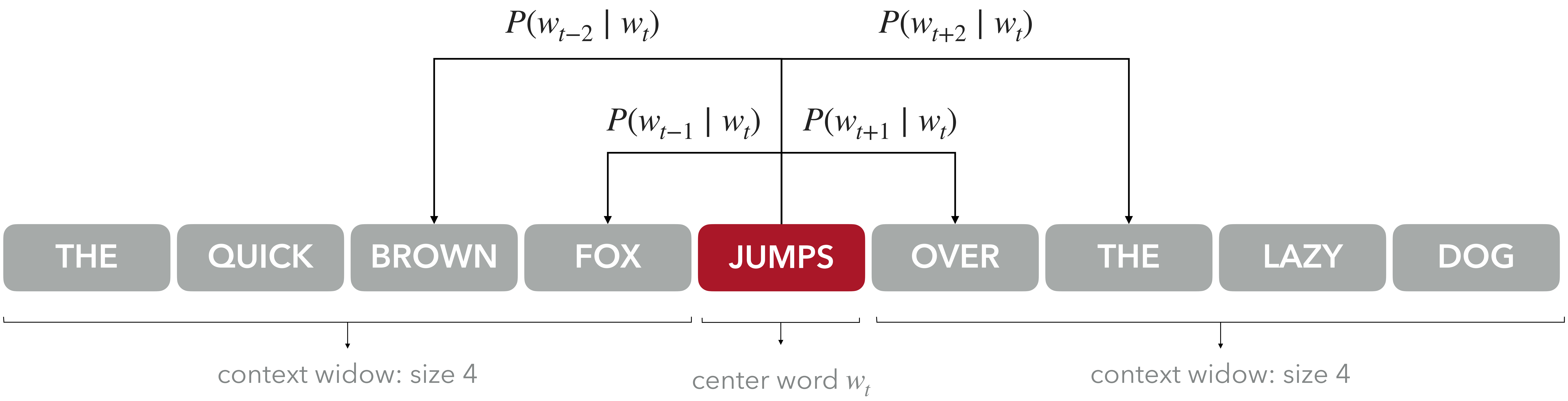
WORD2VEC [Mikolov et al., 2013]



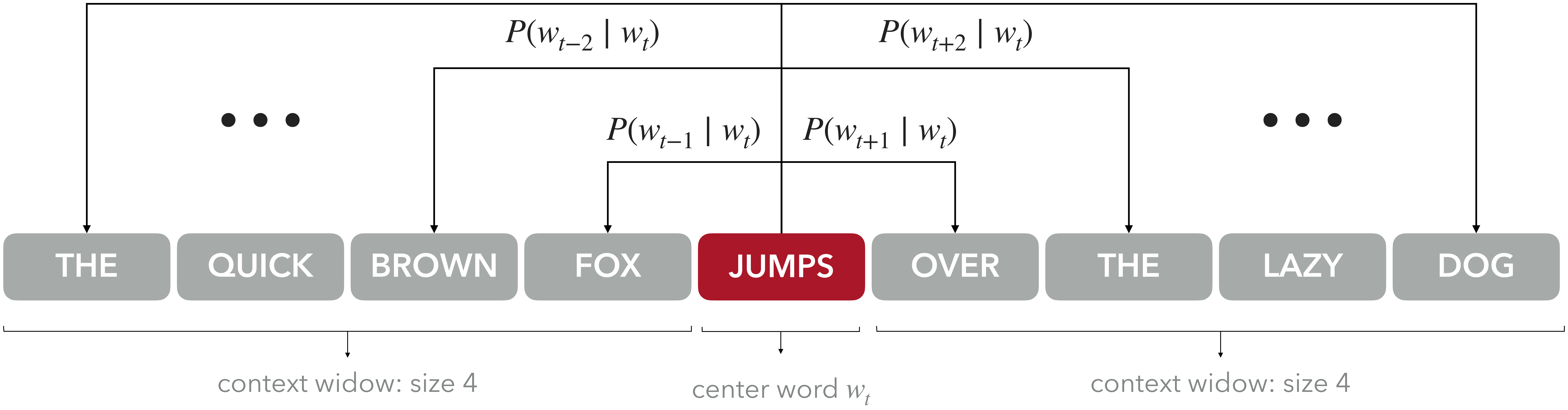
WORD2VEC [Mikolov et al., 2013]



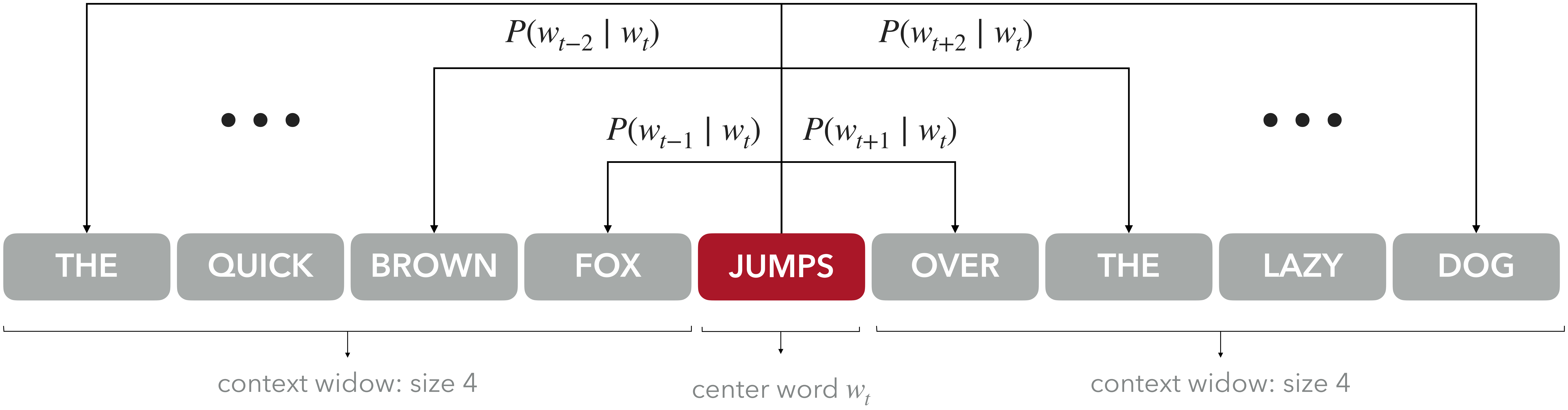
WORD2VEC [Mikolov et al., 2013]



WORD2VEC [Mikolov et al., 2013]



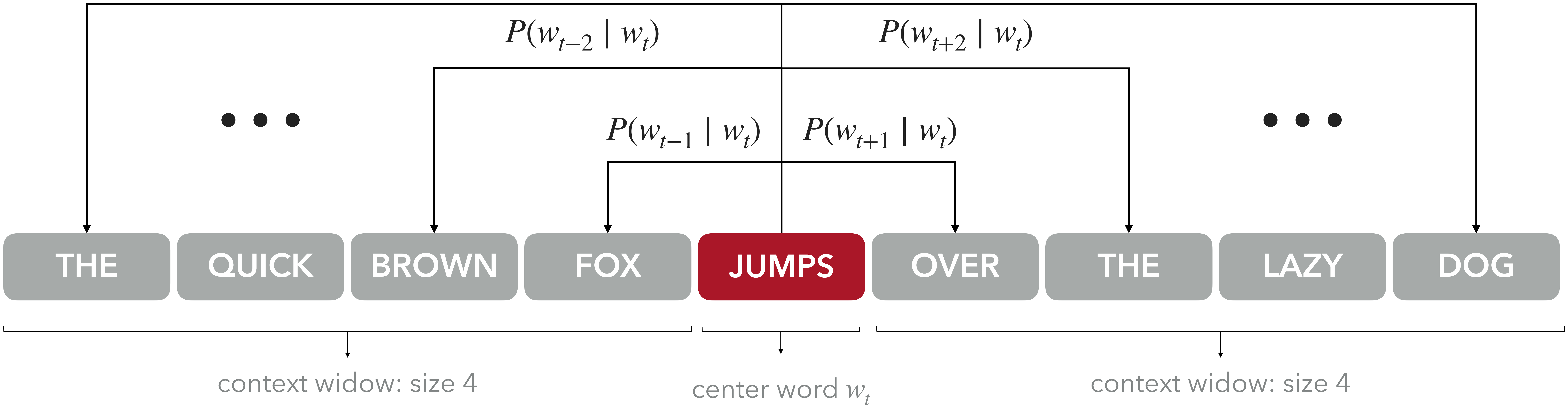
WORD2VEC [Mikolov et al., 2013]



Joint Probability (of context given center word):

$$p(w_{t-m}, \ldots, w_{t+m} \mid w_t) = \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]



Joint Probability (of context given center word):

$$p(w_{t-m}, \ldots, w_{t+m} \mid w_t) = \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

Naive Bayes assumption

WORD2VEC [Mikolov et al., 2013]

QUICK BROWN FOX JUMPS **OVER** THE LAZY DOG THEN

context widow: size 4

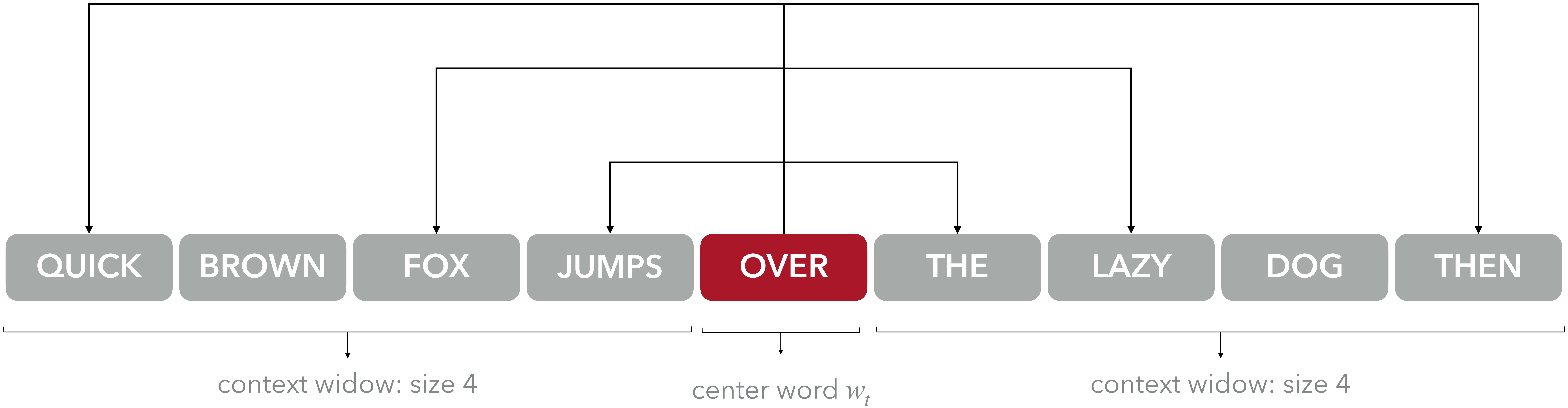
center word w_t

context widow: size 4

Joint Probability (of context given center word):

$$p(w_{t-m}, \ldots, w_{t+m} \mid w_t) = \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]



Joint Probability (of context given center word):

$$p(w_{t-m}, \ldots, w_{t+m} \mid w_t) = \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]

Likelihood
(of entire document)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]

Likelihood
(of entire document)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

Objective Function
(negative log-likelihood)

WORD2VEC [Mikolov et al., 2013]

Likelihood
(of entire document)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

Objective Function
(negative log-likelihood)

$$-\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]

Likelihood
(of entire document)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

Objective Function
(negative log-likelihood)

$$-\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

We want to minimize NLL (i.e., maximize likelihood)

WORD2VEC [Mikolov et al., 2013]

Likelihood
(of entire document)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

Objective Function
(negative log-likelihood)

$$-\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

We want to minimize NLL (i.e., maximize likelihood)

(an instance of Maximum Likelihood Estimation)

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

$$P(o \mid c \; ; \; \theta)$$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

context word

$$P(o \mid c \,;\, \theta)$$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

$$\underset{\substack{\uparrow \\ \text{context word}}}{}\quad\underset{\substack{\uparrow \\ \text{center word}}}{}$$

$$P(\underbrace{o}_{\text{context word}} \mid \underbrace{c}_{\text{center word}} \,;\, \theta)$$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

context word   center word

$$P(o \mid c \; ; \; \theta)$$

θ stands for all the model parameters:

$\theta = \{(u_w, v_w)\}_{w \in \text{Vocab}}$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

context word

center word

$$P(o \mid c\,;\,\theta) = \frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

θ stands for all the model parameters:

$\theta = \{(u_w, v_w)\}_{w \in \text{Vocab}}$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

Each word gets two vectors:

v_w when w is a center word,

u_w when it is a context word

context word

center word

$$P(o \mid c\,;\,\theta) = \frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

θ stands for all the model parameters:

$\theta = \{(u_w, v_w)\}_{w \in \text{Vocab}}$

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

Each word gets two vectors:
v_w when w is a center word,
u_w when it is a context word

context word

center word

$$P(o \mid c\ ;\ \theta) = \frac{\exp(u_o^\top v_c)}{\underbrace{\sum_{w \in V} \exp(u_w^\top v_c)}}$$

θ stands for all the model parameters:
$\theta = \{(u_w, v_w)\}_{w \in \text{Vocab}}$

normalizing term, makes P add up to 1

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

Each word gets two vectors:
v_w when w is a center word,
u_w when it is a context word

$u_o^\top v_c$: high if vectors point in the same direction
(i.e., a notion of similarity)

context word

center word

$$P(o \mid c\,;\,\theta) = \frac{\exp(u_o^\top v_c)}{\underbrace{\sum_{w \in V} \exp(u_w^\top v_c)}}$$

θ stands for all the model parameters:
$\theta = \{(u_w, v_w)\}_{w \in \text{Vocab}}$

normalizing term,
makes P add up to 1

WORD2VEC [Mikolov et al., 2013]

Modeling word-to-word Probability

Each word gets two vectors:
v_w when w is a center word,
u_w when it is a context word

$u_o^\top v_c$: high if vectors point in the same direction
(i.e., a notion of similarity)

context word (o), center word (c)

$$P(o \mid c\,;\,\theta) = \underbrace{\frac{\exp(u_o^\top v_c)}{\sum_{w\in V}\exp(u_w^\top v_c)}}_{} = \text{softmax}(\underset{|V|\times d}{\mathbf{U}^\top}(\underset{d\times|V|}{\mathbf{V}}\underset{|V|\times 1}{\mathbf{1}_c}))\cdot \mathbf{1}_o$$

θ stands for all the model parameters:
$\theta = \{(u_w, v_w)\}_{w\in\text{Vocab}}$

normalizing term,
makes P add up to 1

WORD2VEC [Mikolov et al., 2013]

Modeling Document Likelihood

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

WORD2VEC [Mikolov et al., 2013]

Modeling Document Likelihood

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

$$= -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log \frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

WORD2VEC [Mikolov et al., 2013]

Modeling Document Likelihood

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

$$= -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log \frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

$$= -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log u_o^\top v_c + \log \sum_{w \in V} \exp(u_w^\top v_c)$$

WORD2VEC [Mikolov et al., 2013]

Modeling Document Likelihood

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

$$= -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log \frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

$$= -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log u_o^\top v_c + \log \sum_{w \in V} \exp(u_w^\top v_c)$$

Optimization: Stochastic Gradient Descent one update for every t

WORD2VEC [Mikolov et al., 2013]

Algorithmic Considerations

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log u_o^\top v_c + \log \sum_{w \in V} \exp(u_w^\top v_c)$$

WORD2VEC [Mikolov et al., 2013]

Algorithmic Considerations

$$\text{Loss}(\mathbf{U},\mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log u_o^\top v_c + \log \sum_{w\in V}\exp(u_w^\top v_c)$$

What's wrong with this objective?

WORD2VEC [Mikolov et al., 2013]

Algorithmic Considerations

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log u_o^\top v_c + \underbrace{\log \sum_{w \in V} \exp(u_w^\top v_c)}$$

What's wrong with this objective? This an $O(|V|)$ sum! Potentially huge

WORD2VEC [Mikolov et al., 2013]

Algorithmic Considerations

$$\text{Loss}(\mathbf{U}, \mathbf{V}) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log u_o^\top v_c + \underbrace{\log \sum_{w \in V} \exp(u_w^\top v_c)}$$

What's wrong with this objective? This an $O(|V|)$ sum! Potentially huge

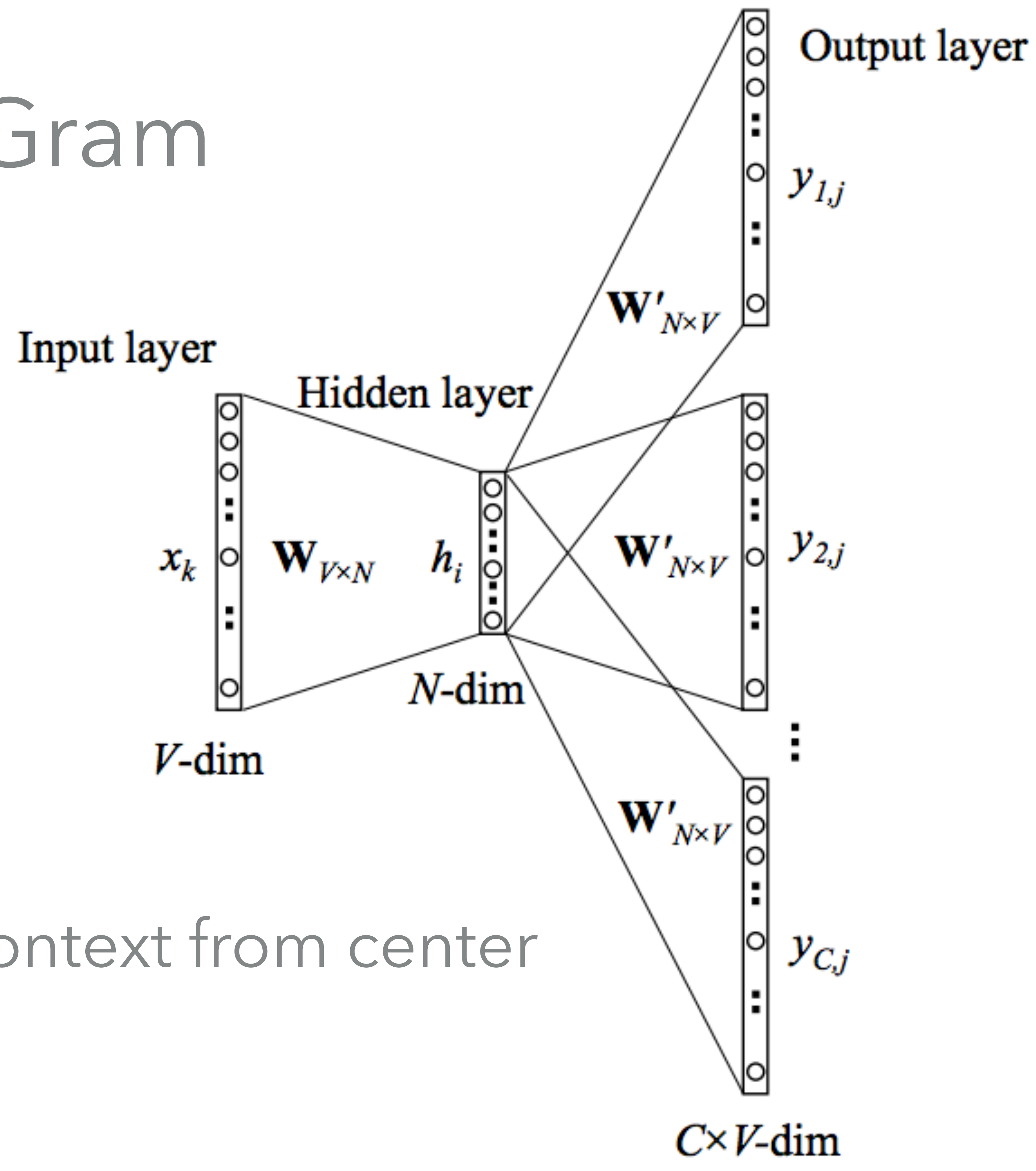
Two Solutions:

1. Negative sampling (solves a slightly different objective)
2. Hierarchical softmax (computes softmax via binary tree)

WORD2VEC [Mikolov et al., 2013]

Two Flavors of Prediction

Skip-Gram

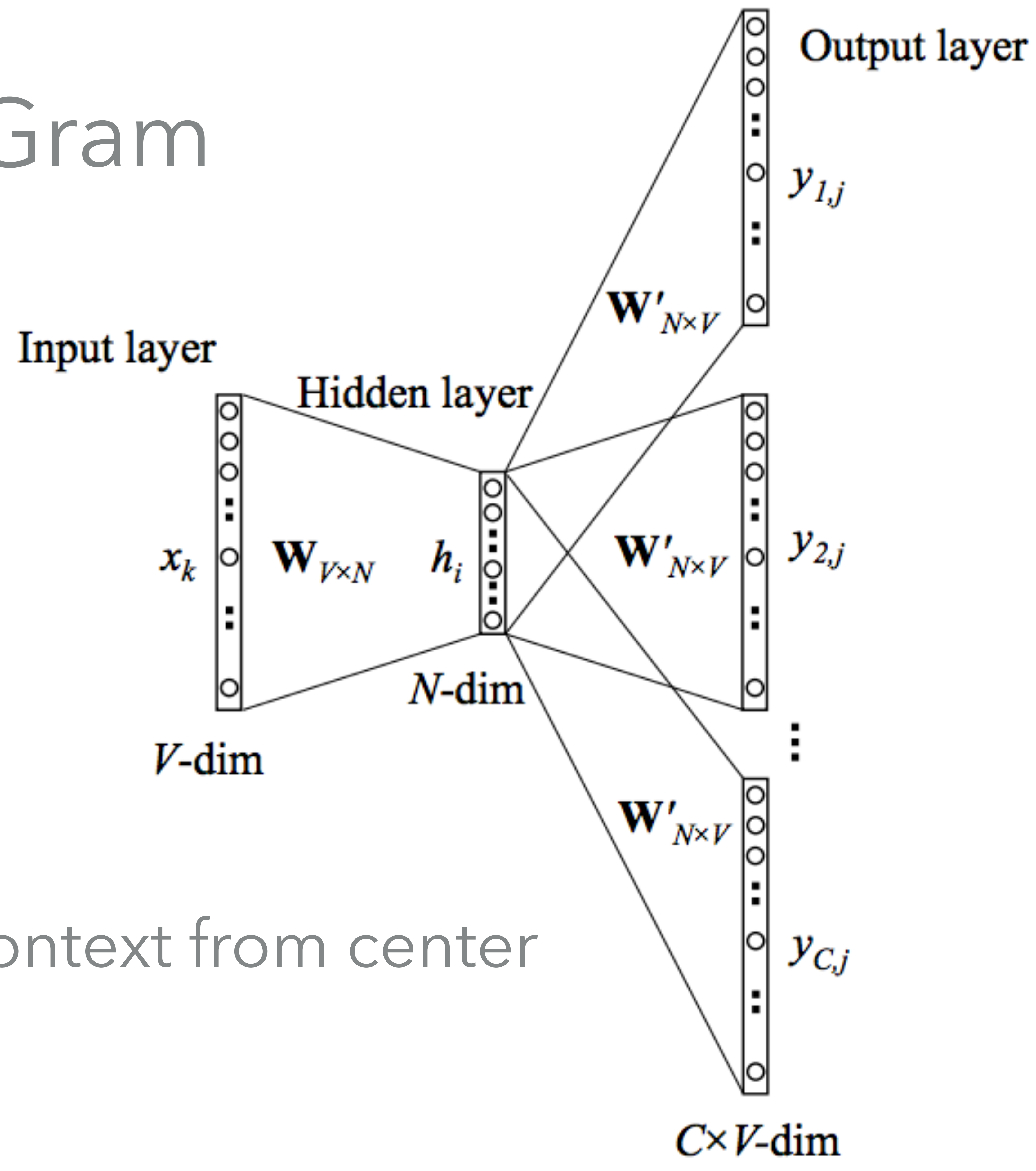


predict context from center

WORD2VEC [Mikolov et al., 2013]

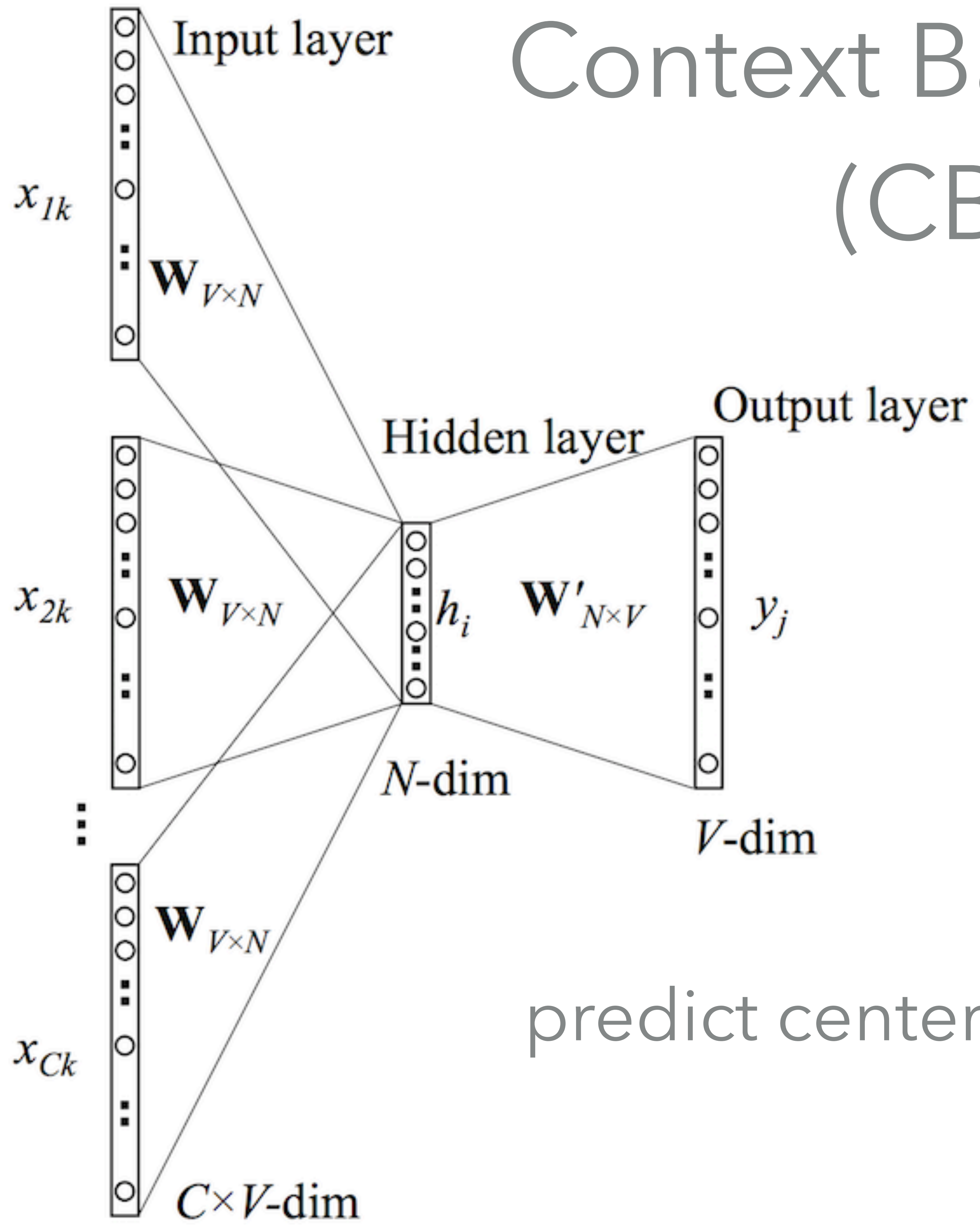
Two Flavors of Prediction

Skip-Gram



predict context from center

Context Bag of Words (CBOW)



predict center from context

WORD2VEC [Mikolov et al., 2013]

Visualizing Word2Vec Embeddings

(DEMO): https://projector.tensorflow.org/

WORD2VEC [Mikolov et al., 2013]

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \quad \approx$$

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \quad \approx \quad x_{\text{queen}}$$

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \quad \approx \quad x_{\text{queen}}$$

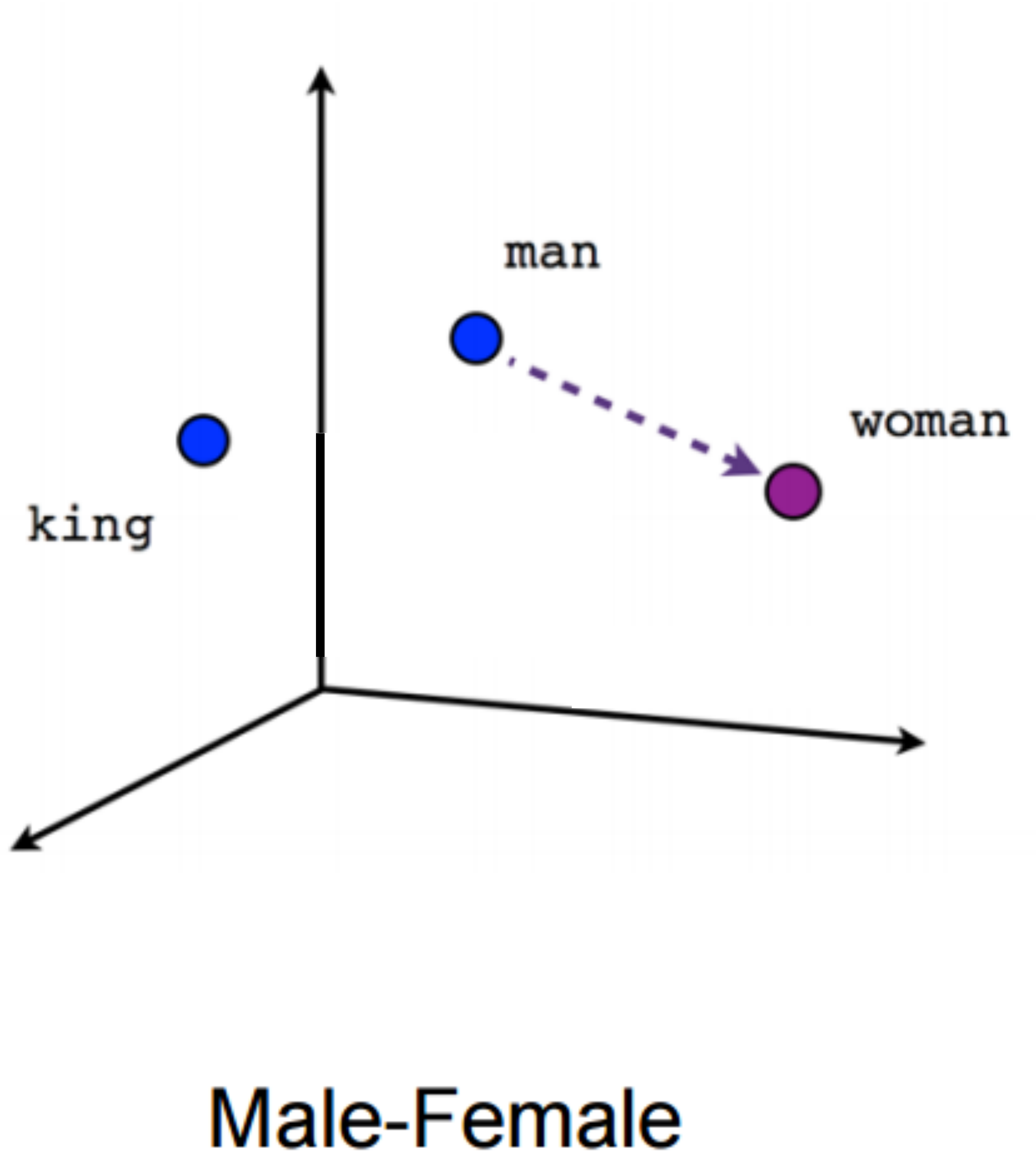
!!!

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \quad \approx \quad x_{\text{queen}}$$

!!!



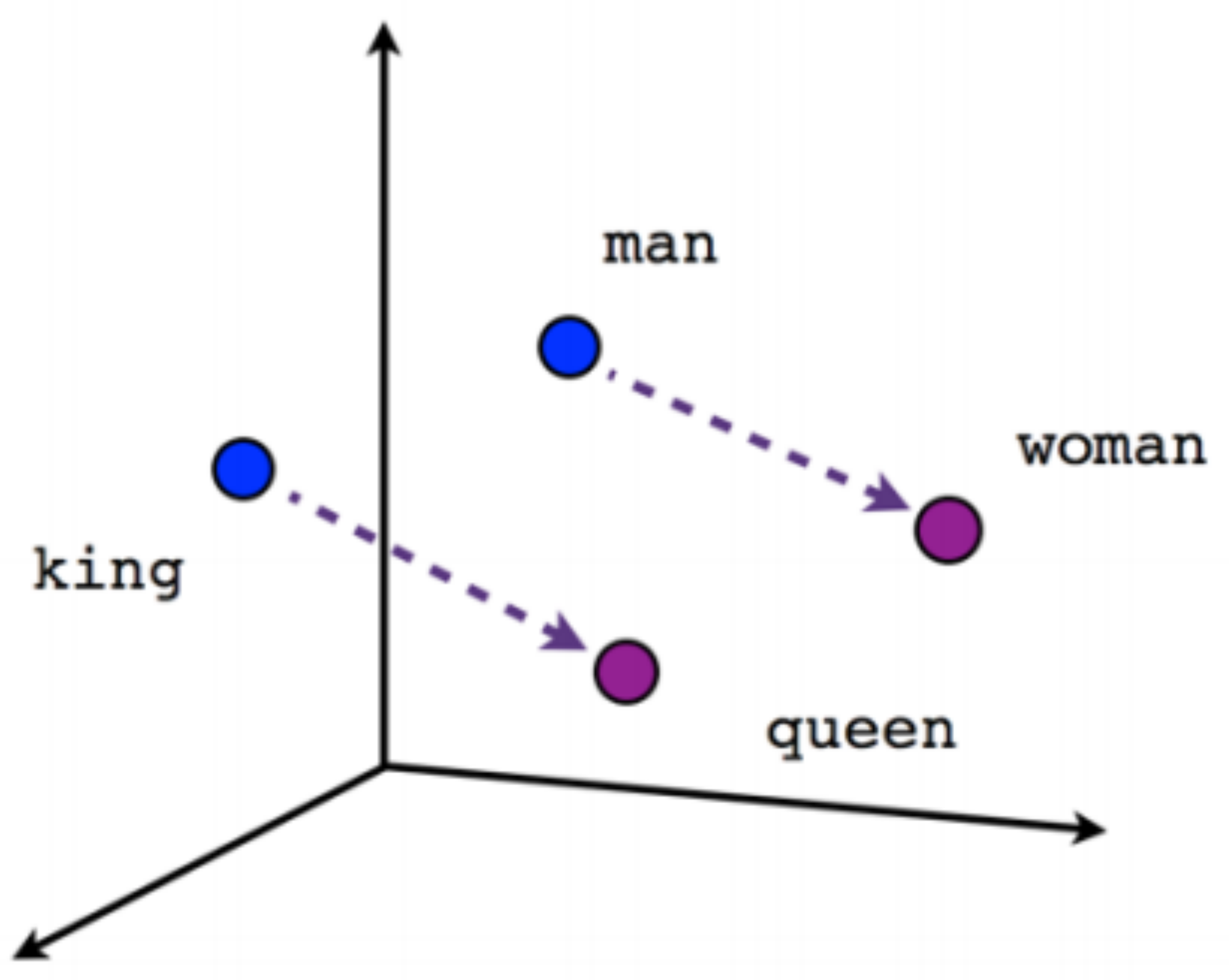
Source: tensorflow.org/tutorials

LINEAR ALGEBRA WITH WORDS

- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \quad \approx \quad x_{\text{queen}}$$

!!!



Male-Female

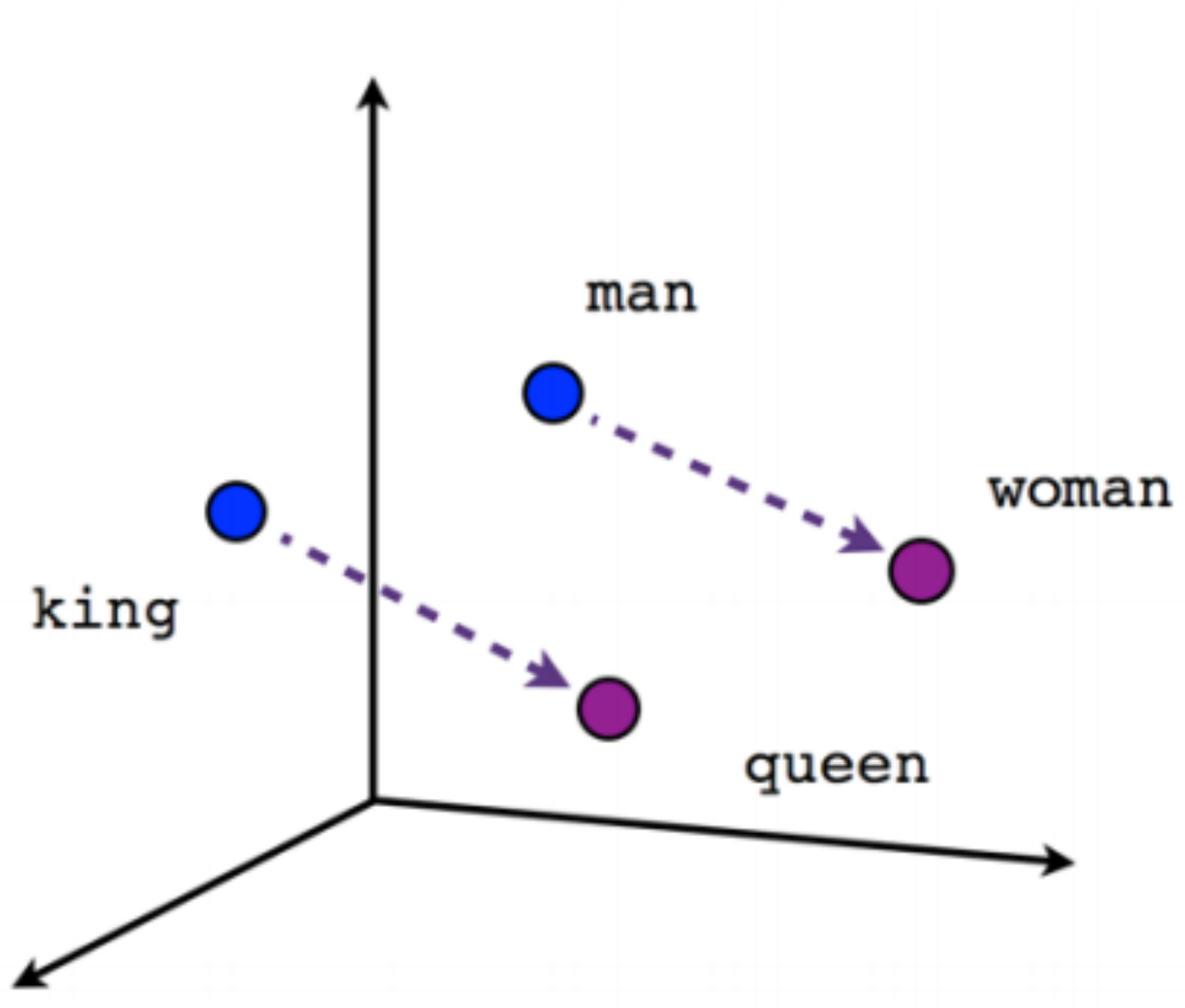
Source: tensorflow.org/tutorials

LINEAR ALGEBRA WITH WORDS

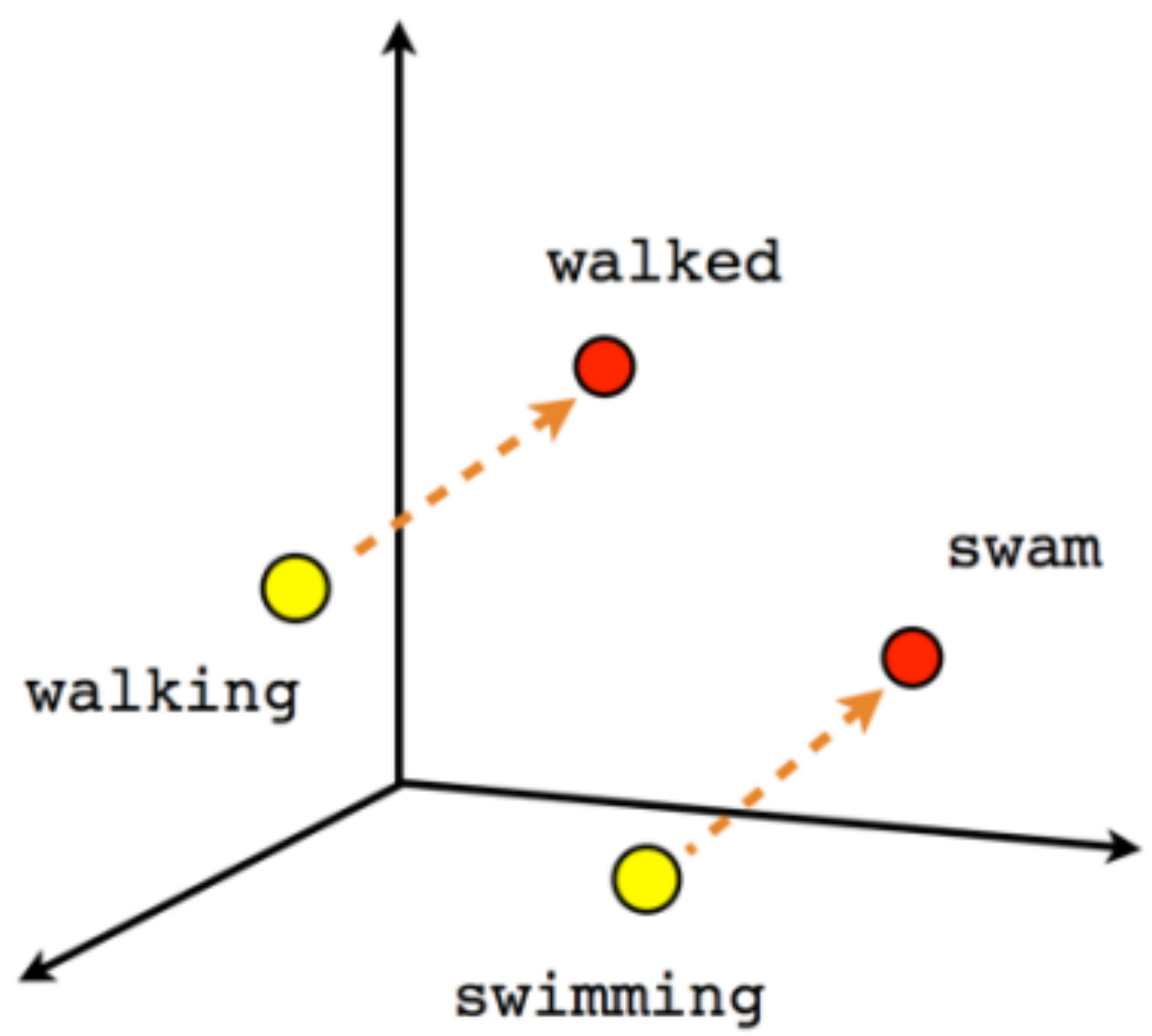
- Mikolov et al. (2013): Geometry of word2vec space is linear

$$x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} \approx x_{\text{queen}}$$

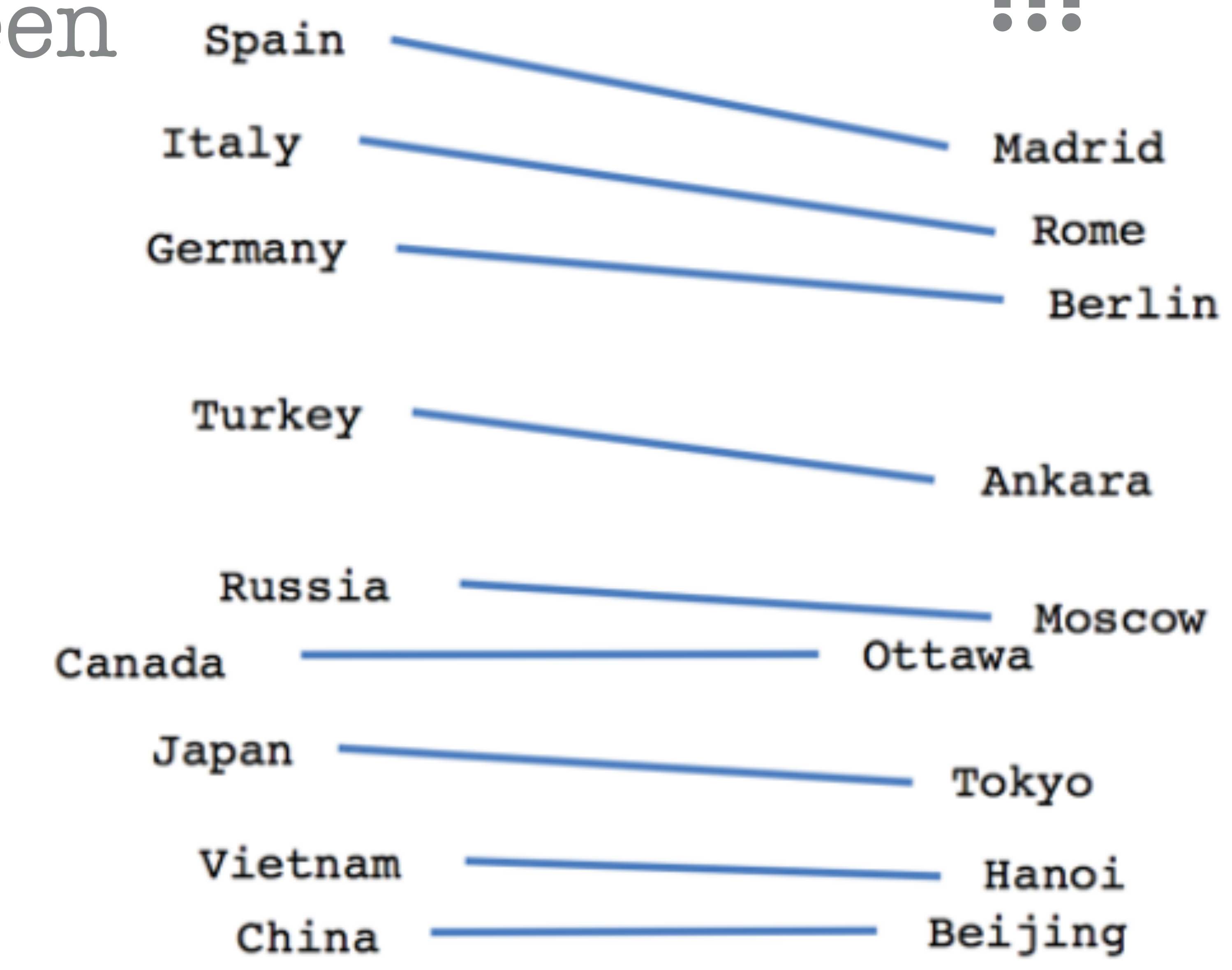
!!!



Male-Female



Verb tense



Country-Capital

Source: tensorflow.org/tutorials

IN SEARCH OF AN EXPLANATION

IN SEARCH OF AN EXPLANATION

- **Levy & Goldberg (2014)**: "Count-based vectors have this property too!"

IN SEARCH OF AN EXPLANATION

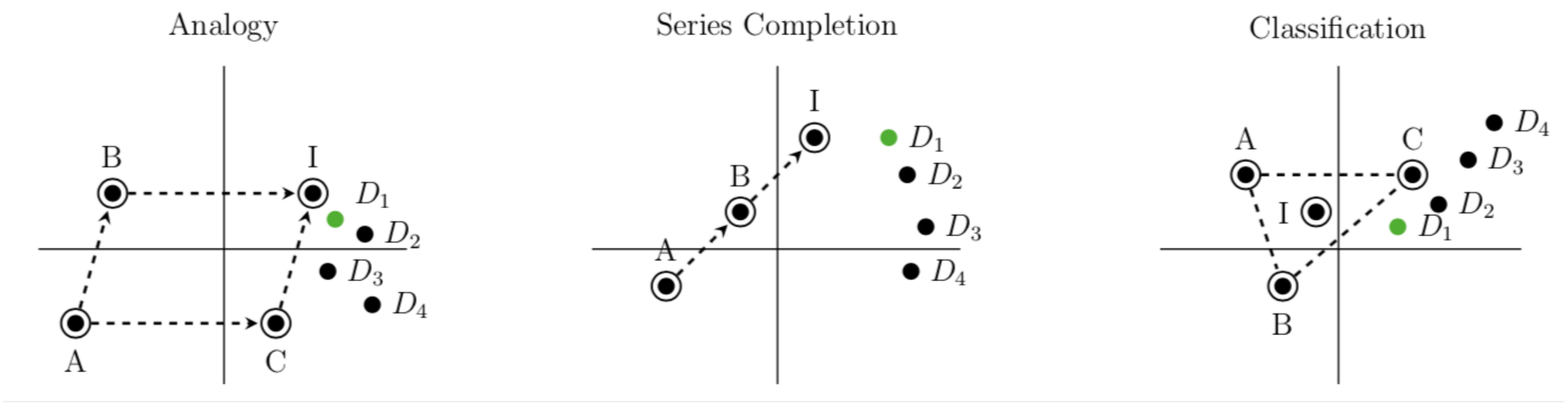
- **Levy & Goldberg (2014)**: "Count-based vectors have this property too!"
- **Arora et al. (2015)**: "It is a direct consequence of using co-occurrence statistics."

IN SEARCH OF AN EXPLANATION

- **Levy & Goldberg (2014)**: "Count-based vectors have this property too!"
- **Arora et al. (2015)**: "It is a direct consequence of using co-occurrence statistics."
- **Hashimoto et al. (2016)**: "This has been observed in cog. sci. much earlier! Conjecture: word2vec recovers metric of an implicit cognitive-semantic vector space"

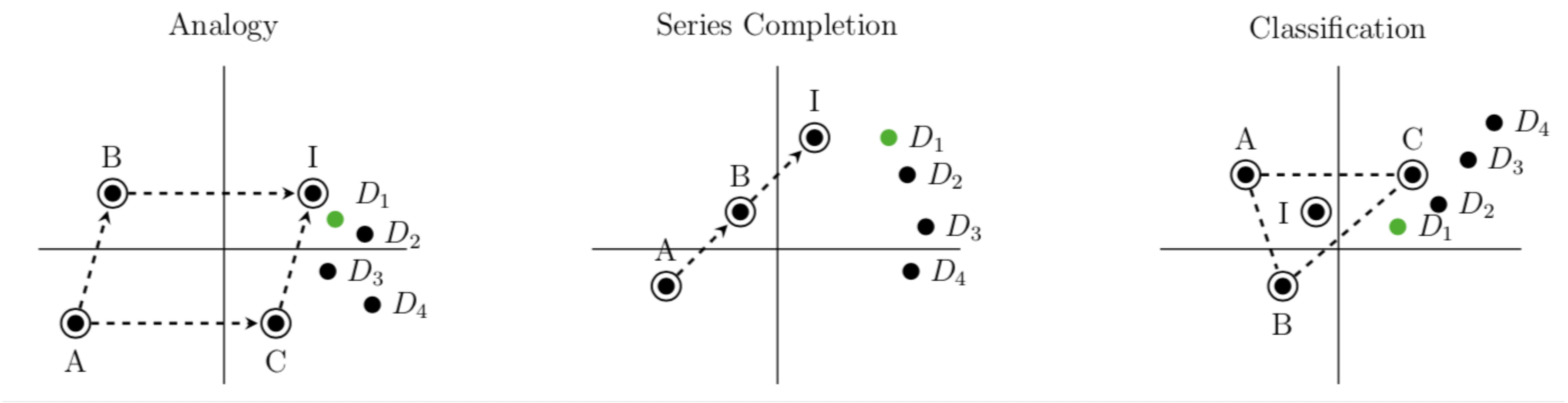
IN SEARCH OF AN EXPLANATION

- **Levy & Goldberg (2014)**: "Count-based vectors have this property too!"
- **Arora et al. (2015)**: "It is a direct consequence of using co-occurrence statistics."
- **Hashimoto et al. (2016)**: "This has been observed in cog. sci. much earlier! Conjecture: word2vec recovers metric of an implicit cognitive-semantic vector space"



IN SEARCH OF AN EXPLANATION

- **Levy & Goldberg (2014)**: "Count-based vectors have this property too!"
- **Arora et al. (2015)**: "It is a direct consequence of using co-occurrence statistics."
- **Hashimoto et al. (2016)**: "This has been observed in cog. sci. much earlier! Conjecture: word2vec recovers metric of an implicit cognitive-semantic vector space"



[STERNBERG & GARDNER, 1983]

BASED ON WORD ASSOCIATION TESTS

ALL VECTORS LEAD TO ROME

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Abstract

tor space

the finer structure of the word vector space by examining not the scalar distance between word vec-

Neural Word Embedding as Implicit Matrix Factorization

Omer Levy
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

Yoav Goldberg
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

We analyze skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al., and show that it is implicitly factorizing

ALL VECTORS LEAD TO ROME

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

GloVe: Global Vectors for Word Representation

hard Socher, Christopher D. Manning
nt, Stanford University, Stanford, CA 94305
ard@socher.org, manning@stanford.edu

ce

the finer structure of the word vector space by examining not the scalar distance between word vec-

THEY ARE ALL (ESSENTIALLY) EQUIVALENT

[HASHIMOTO, AM & JAAKKOLA, 2015]

Neural Word Embedding as Implicit Matrix Factorization

Omer Levy
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

Yoav Goldberg
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

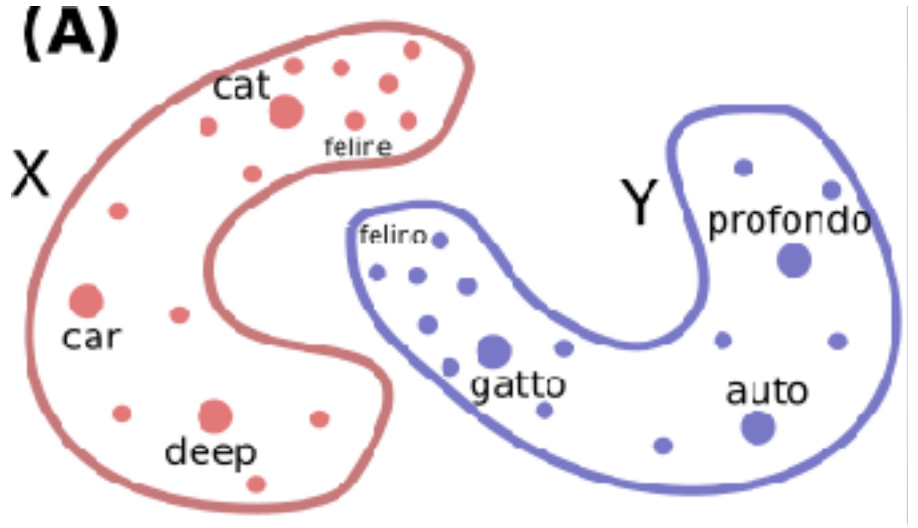
We analyze skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al., and show that it is implicitly factorizing

BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:

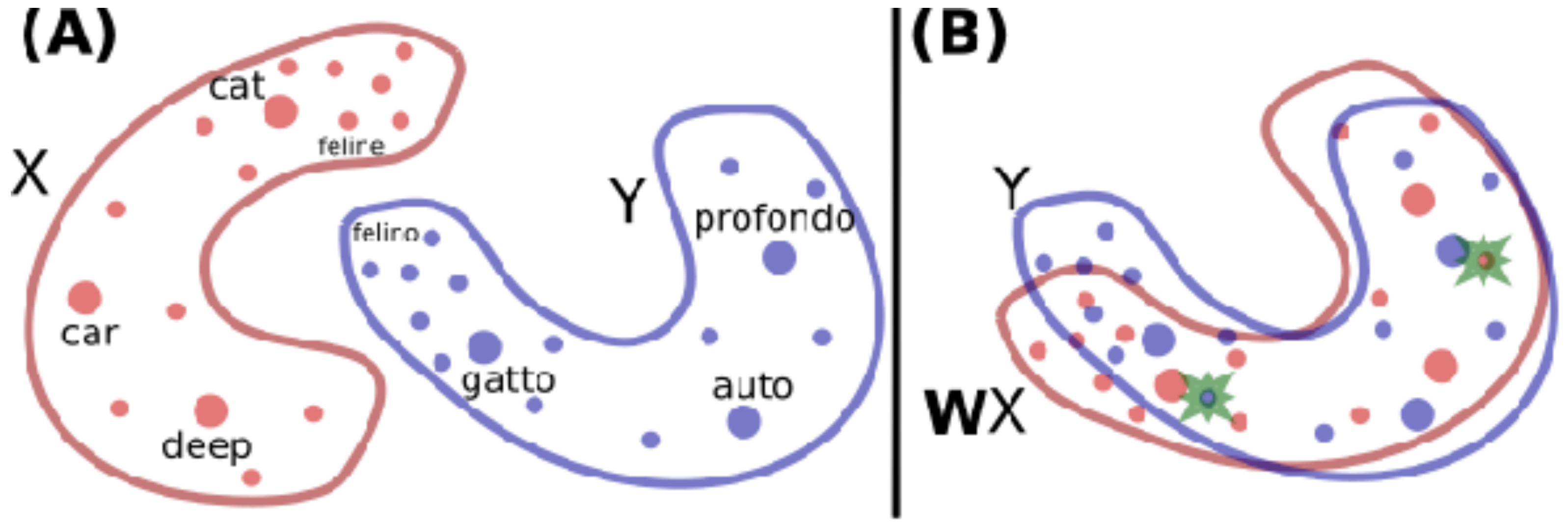
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



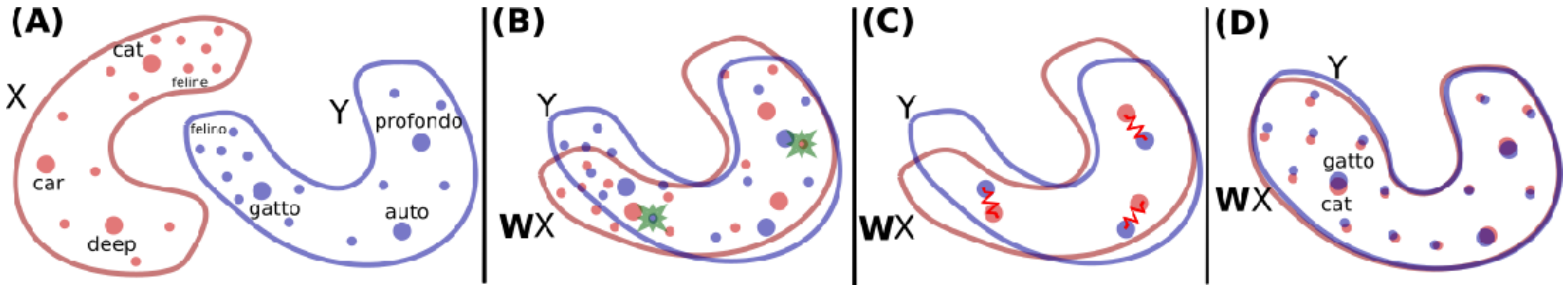
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



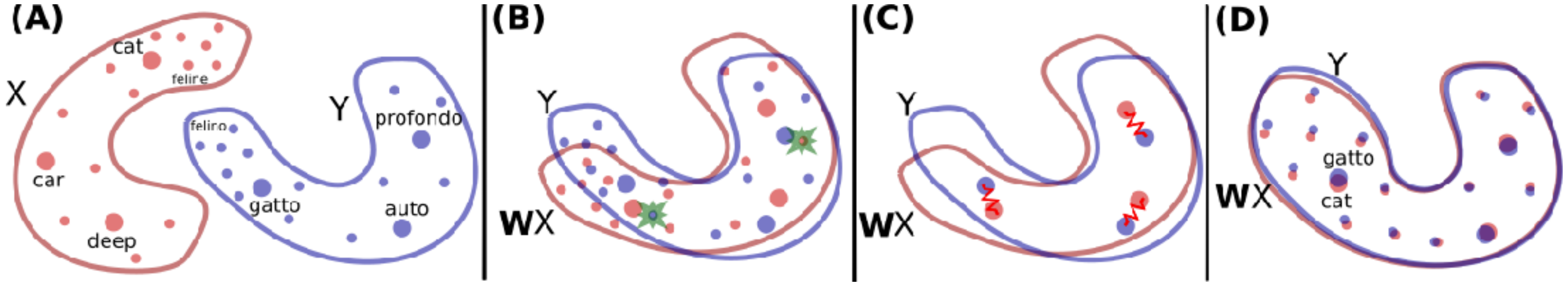
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

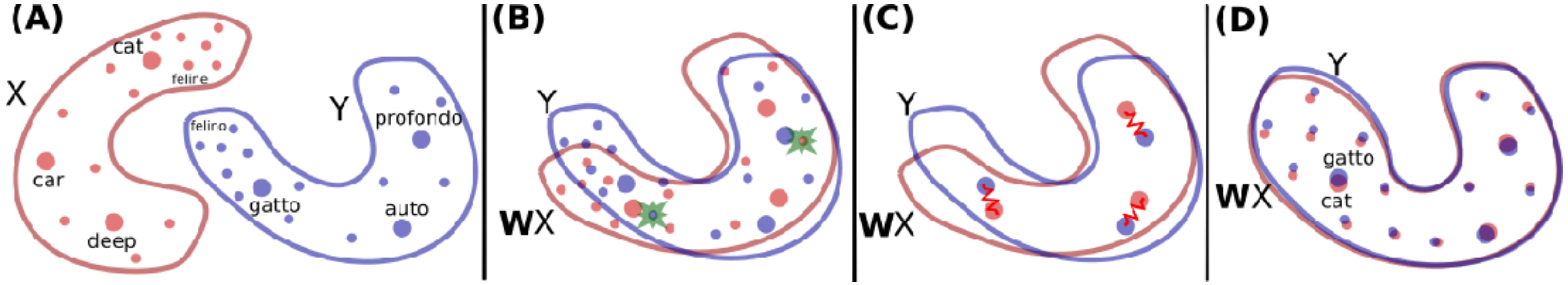
[Conneau et al 2018]:



SOLUTION FOUND THROUGH ADVERSARIAL TRAINING

BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:

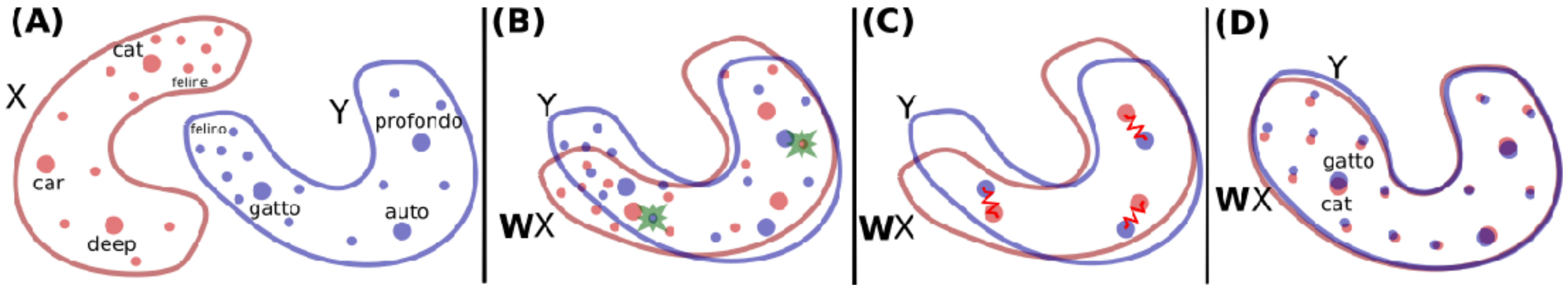


SOLUTION FOUND THROUGH ADVERSARIAL TRAINING

[AM & Jaakkola 2018]:

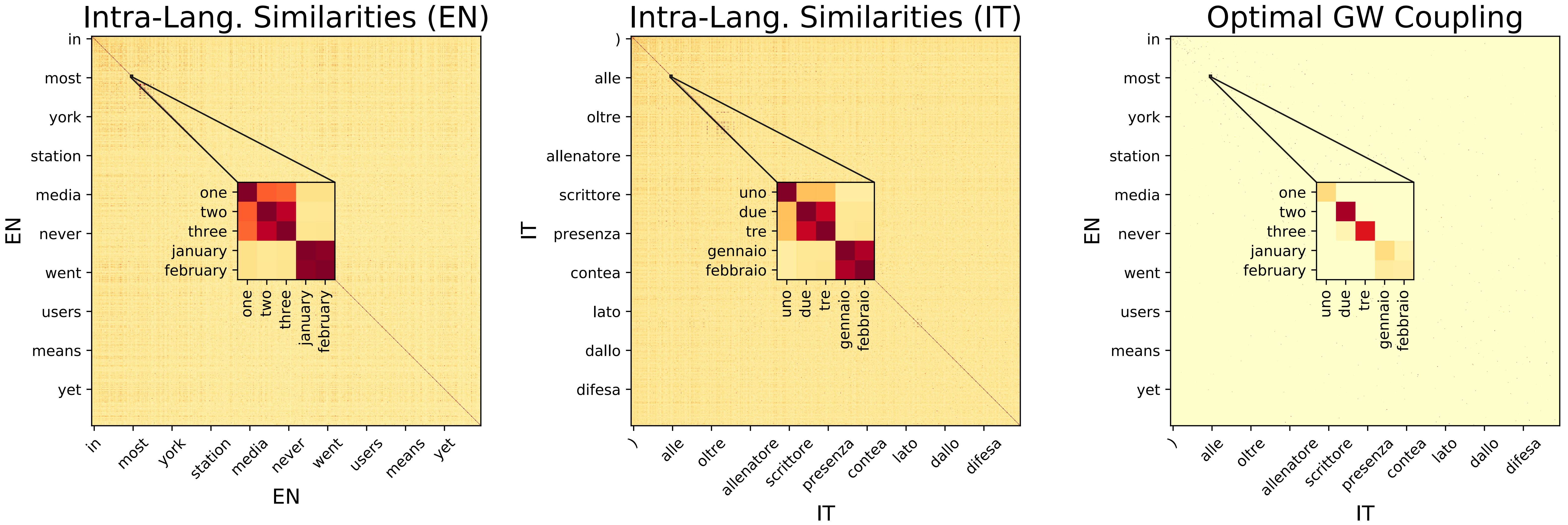
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



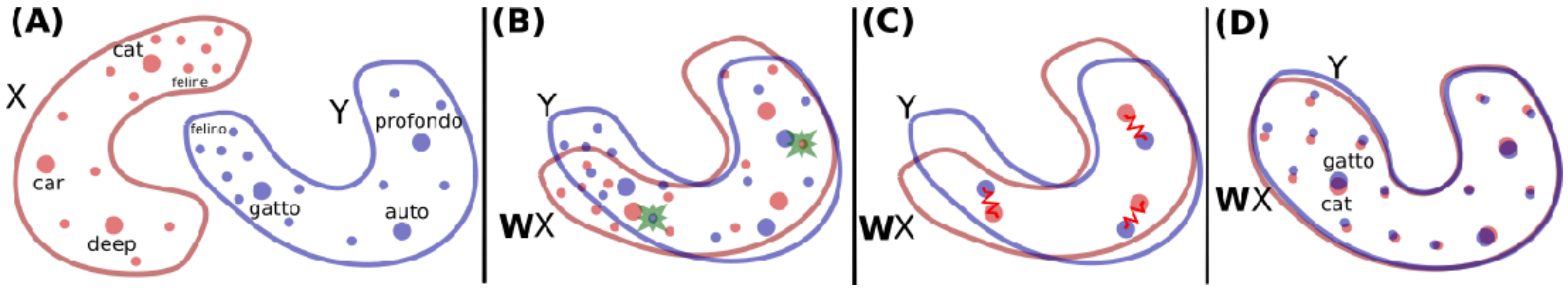
SOLUTION FOUND THROUGH ADVERSARIAL TRAINING

[AM & Jaakkola 2018]:



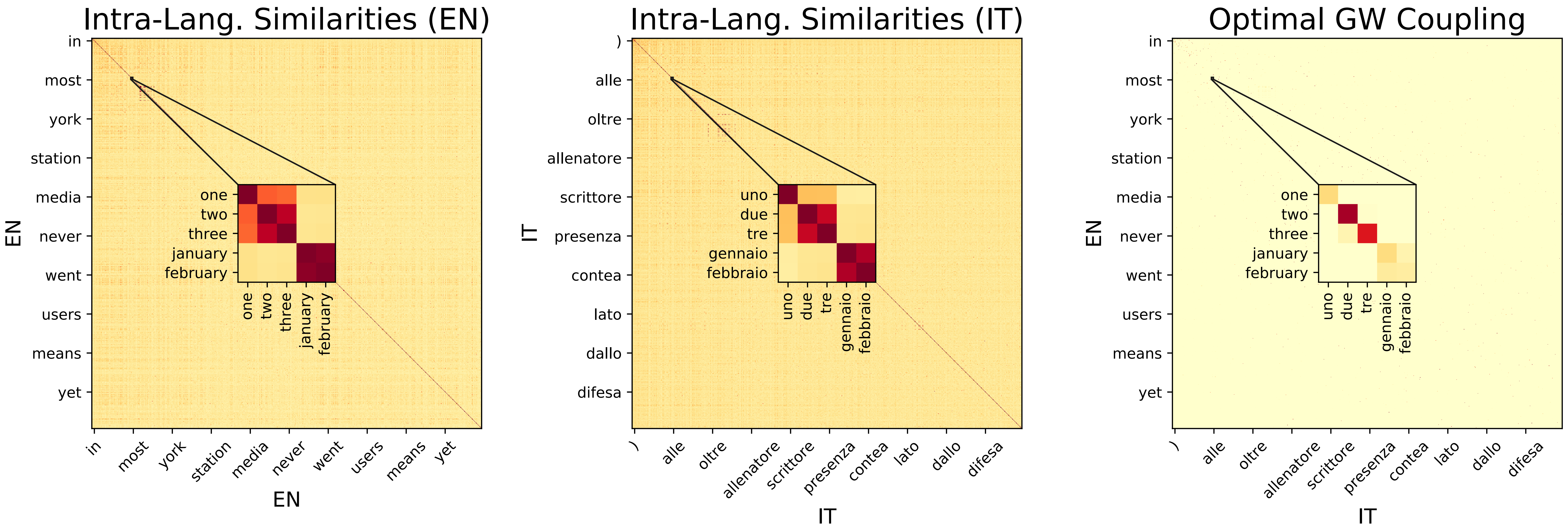
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



SOLUTION FOUND THROUGH ADVERSARIAL TRAINING

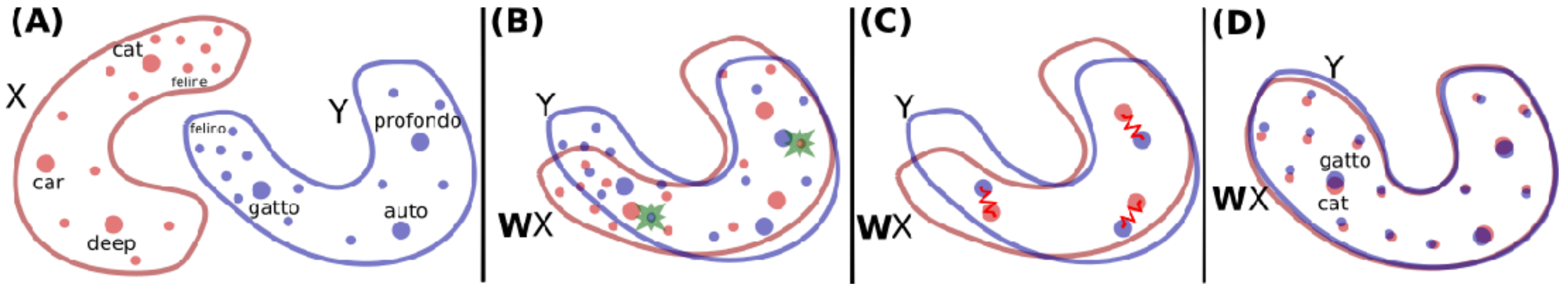
[AM & Jaakkola 2018]:



ALLOWS FOR EMBEDDING SPACES OF DIFFERENT DIMENSION

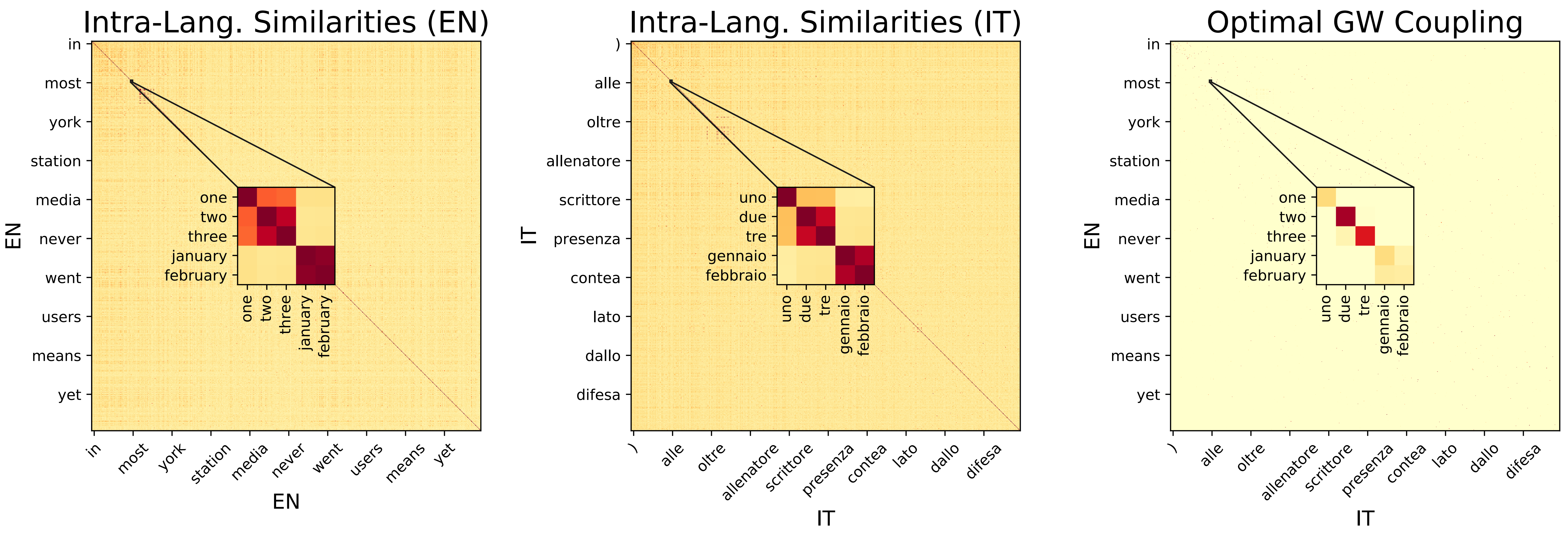
BONUS: AUTOMATIC TRANSLATION USING EMBEDDINGS

[Conneau et al 2018]:



SOLUTION FOUND THROUGH ADVERSARIAL TRAINING

[AM & Jaakkola 2018]:



ALLOWS FOR EMBEDDING SPACES OF DIFFERENT DIMENSION

PROBLEM SOLVED THROUGH EXPLICIT OPTIMIZATION (GROMOV-WASSERSTEIN)

PART 2:

PROCESSING SENTENCES

WITH RECURRENT NEURAL NETWORKS

FROM WORDS TO SENTENCES: REPRESENTATION

"The house is green …"

FROM WORDS TO SENTENCES: REPRESENTATION

"The house is green …"

How do we represent
an entire sentence?

FROM WORDS TO SENTENCES: REPRESENTATION

$$\underset{v_{\text{the}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} \quad \underset{v_{\text{house}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} \quad \underset{v_{\text{is}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} \quad \underset{v_{\text{green}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}$$

"The house is green …"

How do we represent
an entire sentence?

FROM WORDS TO SENTENCES: REPRESENTATION

$$\underset{v_{\text{the house ...}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} = \underset{v_{\text{the}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} + \underset{v_{\text{house}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} + \underset{v_{\text{is}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}} + \underset{v_{\text{green}}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}$$

"The house is green ..."

How do we represent
an entire sentence?

FROM WORDS TO SENTENCES: REPRESENTATION

A car leaves its shed.

A tree shed its leaves.

$$\underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{the house ...}}} = \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{the}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{house}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{is}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{green}}}$$

"The house is green …"

How do we represent
an entire sentence?

FROM WORDS TO SENTENCES: REPRESENTATION

$$\underset{v_{\text{the house ...}}}{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}} = \underset{v_{\text{the}}}{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}} + \underset{v_{\text{house}}}{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}} + \underset{v_{\text{is}}}{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}} + \underset{v_{\text{green}}}{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}$$

"The house is green ..."

How do we represent
an entire sentence?

A car leaves its shed.

A tree shed its leaves.

Same vector!

FROM WORDS TO SENTENCES: REPRESENTATION

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{v_{\text{the house} \dots}} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{v_{\text{the}}} + \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{v_{\text{house}}} + \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{v_{\text{is}}} + \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{v_{\text{green}}}$$

“The house is green …”

How do we represent
an entire sentence?

A car leaves its shed.

A tree shed its leaves.

Same vector!

Only he told his wife that he loved her.

He **only** told his wife that he loved her.

He told **only** his wife that he loved her.

He told his **only** wife that he loved her.

He told his wife **only** that he loved her.

He told his wife that **only** he loved her.

He told his wife that he **only** loved her.

He told his wife that he loved **only** her.

He told his wife that he loved her **only**.

FROM WORDS TO SENTENCES: REPRESENTATION

$$\underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{the house ...}}} = \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{the}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{house}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{is}}} + \underbrace{\begin{bmatrix}x_1\\x_2\\\vdots\\x_m\end{bmatrix}}_{v_{\text{green}}}$$

"The house is green ..."

How do we represent
an entire sentence?

A car leaves its shed.
A tree shed its leaves.

→ Same vector!

Only he told his wife that he loved her.
He **only** told his wife that he loved her.
He told **only** his wife that he loved her.
He told his **only** wife that he loved her.
He told his wife **only** that he loved her.
He told his wife that **only** he loved her.
He told his wife that he **only** loved her.
He told his wife that he loved **only** her.
He told his wife that he loved her **only**.

→ Same vector!

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house is green, and it ...

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The
- car $(p = 0.2)$
- house $(p = 0.7)$ is green, and it …
- boat $(p = 0.1)$

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house is green, and it …

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house
- was $(p = 0.4)$
- is $(p = 0.4)$ green, and it …
- can $(p = 0.2)$

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house is green, and it …

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house is
- blue $(p = 0.3)$
- green $(p = 0.4)$, and it …
- big $(p = 0.3)$

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

The house is green, and it …

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language Model: a system that assigns probability to a piece of text

$$p(w_1, \dots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \dots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \dots, w_1)$$

The house is green, and it …

FROM WORDS TO SENTENCES: LANGUAGE MODELING

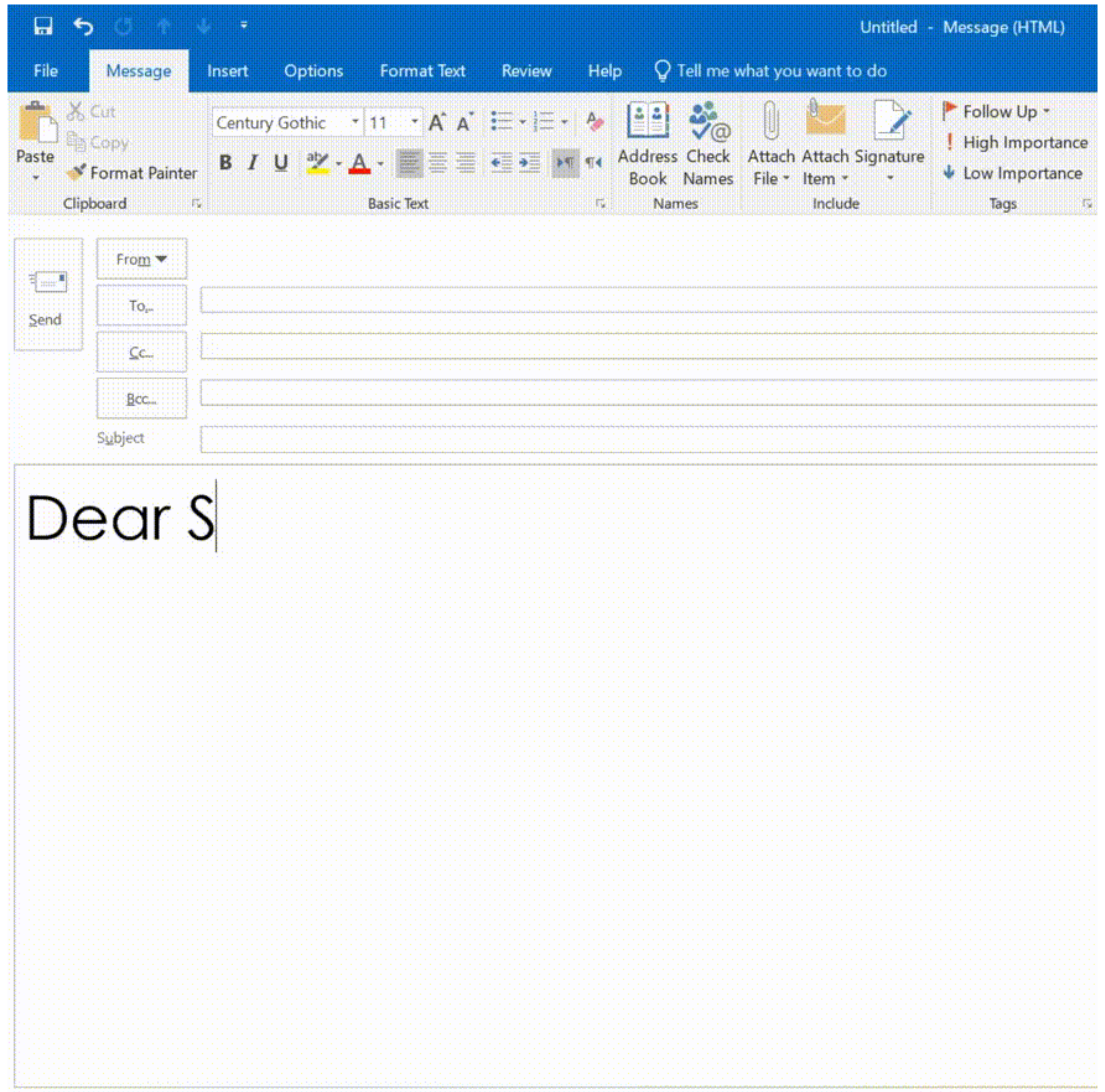
Language Model: a system that assigns probability to a piece of text

$$p(w_1, \ldots, w_T) = p(w_1) \times p(w_2 \mid w_1) \times \cdots \times p(w_T \mid w_{T-1}, \ldots, w_1) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

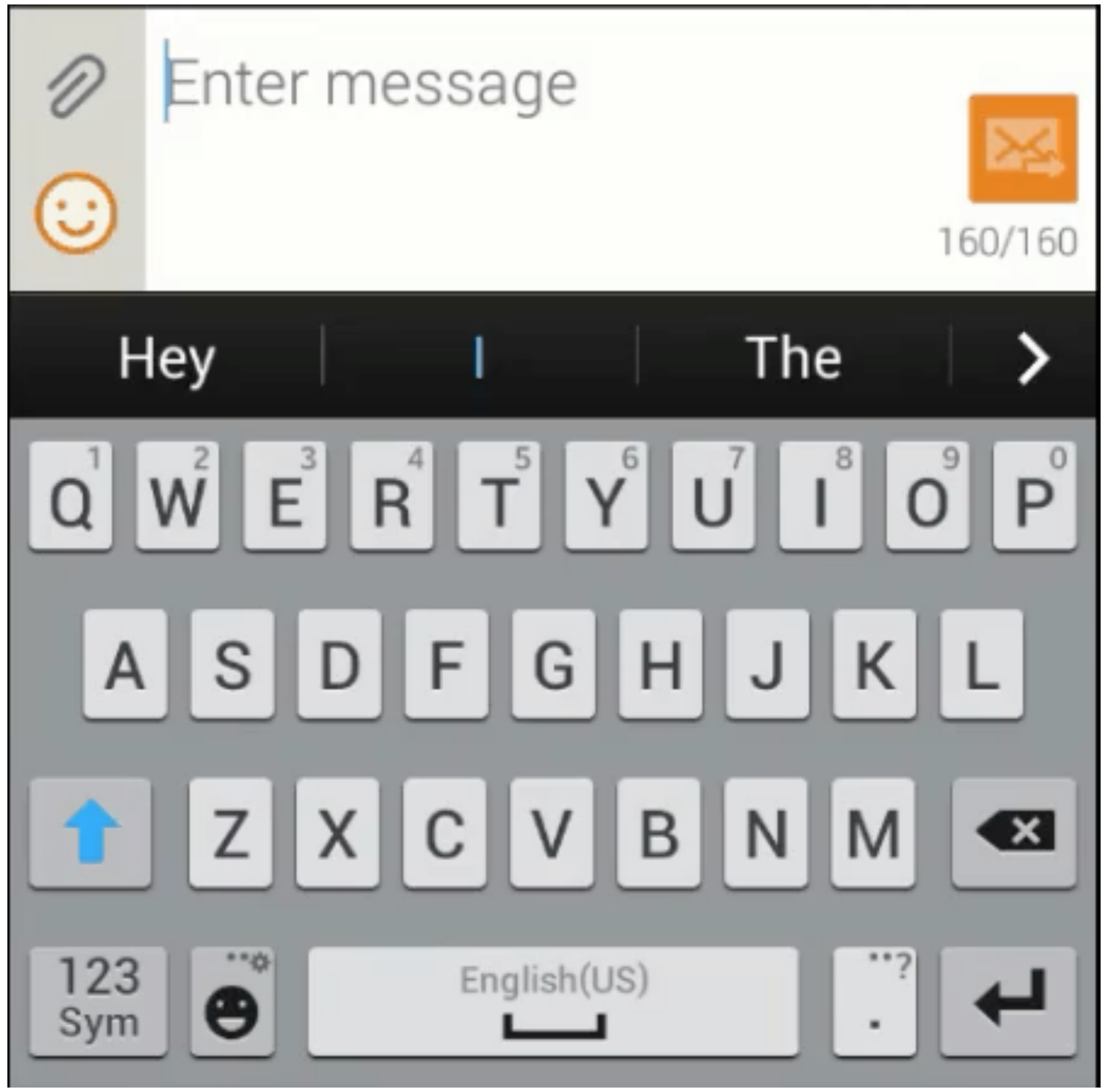
The house is green, and it … $(p = 0.1)$

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language modeling in the wild....



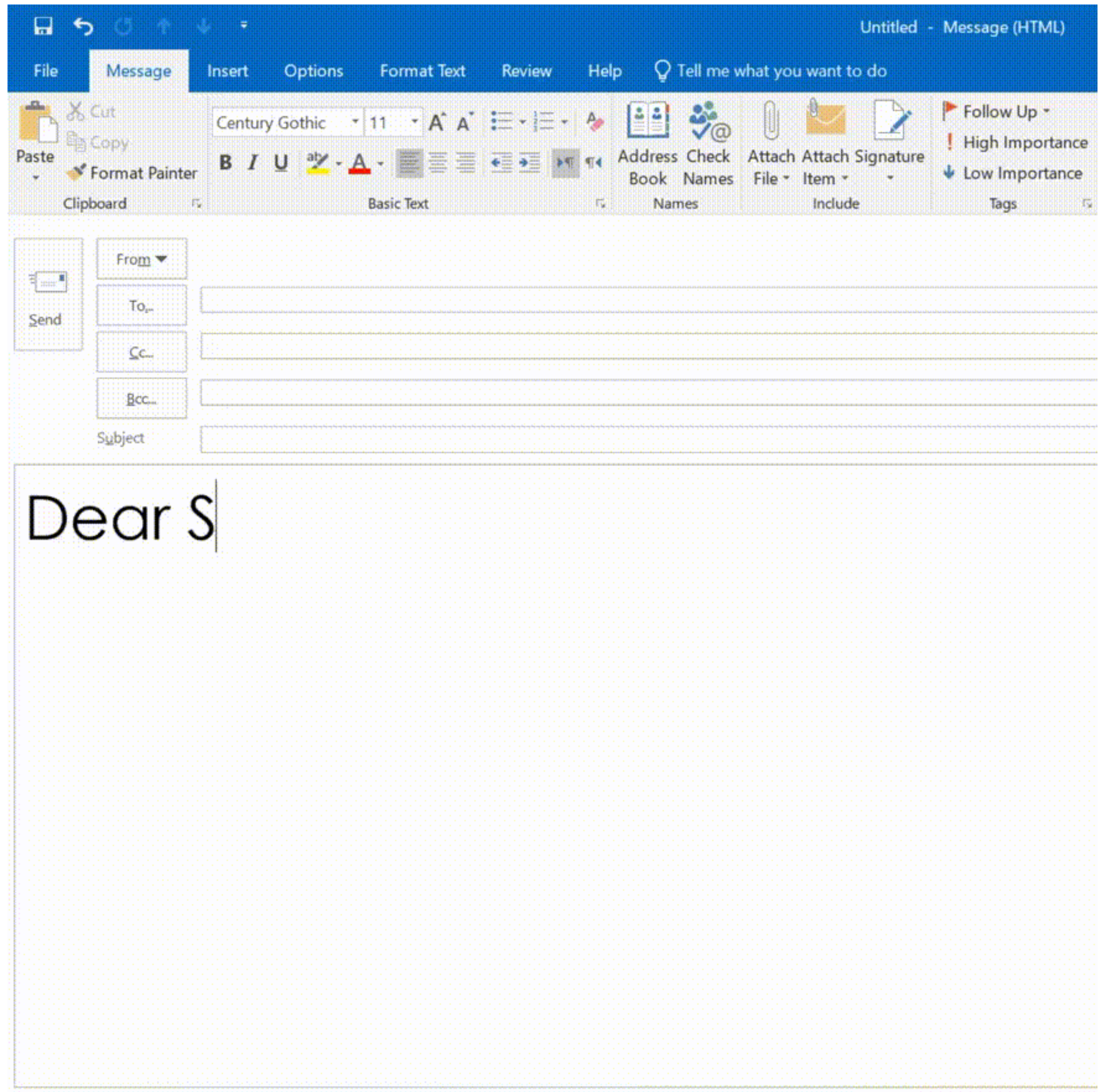
Source: www.lightkey.io



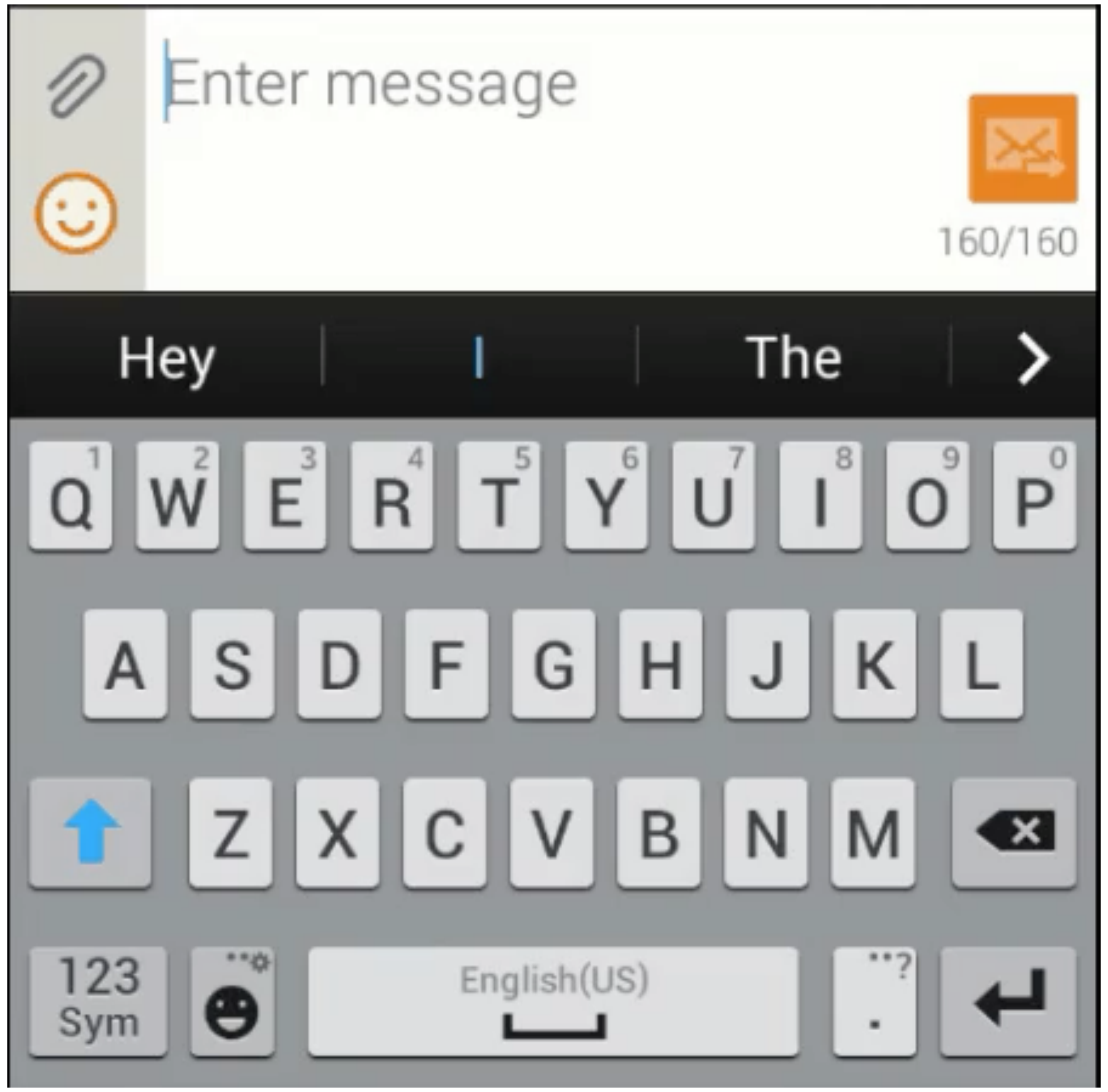
Source: reddit.com/user/wardetbestanee/

FROM WORDS TO SENTENCES: LANGUAGE MODELING

Language modeling in the wild....



Source: www.lightkey.io



Source: reddit.com/user/wardetbestanee/

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

- Unigrams: ["The", "house", "is", "green"]
- Bigrams: ["The house", "house is", "is green"]
- Trigrams: ["The house is", "house is green"]
- 4-grams: ["The house is green"]

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

- Unigrams: ["The", "house", "is", "green"]
- Bigrams: ["The house", "house is", "is green"]
- Trigrams: ["The house is", "house is green"]
- 4-grams: ["The house is green"]

Main idea: estimate next word probability using n-gram counts

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

- Unigrams: ["The", "house", "is", "green"]
- Bigrams: ["The house", "house is", "is green"]
- Trigrams: ["The house is", "house is green"]
- 4-grams: ["The house is green"]

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2})$$

(Markov assumption)

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

- Unigrams: ["The", "house", "is", "green"]
- Bigrams: ["The house", "house is", "is green"]
- Trigrams: ["The house is", "house is green"]
- 4-grams: ["The house is green"]

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \qquad \text{(Markov assumption)}$$

$$= \frac{p(w_{t+1}, w_t, \ldots, w_{t-n+2})}{p(w_t, \ldots, w_{t-n+2})} \qquad \text{(def. of conditional prob.)}$$

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

a sequence of n consecutive words

"The house is green"

- Unigrams: ["The", "house", "is", "green"]
- Bigrams: ["The house", "house is", "is green"]
- Trigrams: ["The house is", "house is green"]
- 4-grams: ["The house is green"]

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2})$$

(Markov assumption)

$$= \frac{p(w_{t+1}, w_t, \ldots, w_{t-n+2})}{p(w_t, \ldots, w_{t-n+2})}$$

(def. of conditional prob.)

$$\approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

(approximate probs via counts, estimated from large corpus)

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

Example: estimate next-word probability for "The house is _____" using trigrams

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

Example: estimate next-word probability for "The house is _____" using trigrams

$$p(\text{green} \mid \text{is}, \text{house}, \text{the}) = p(\text{green} \mid \text{is}, \text{house})$$

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

Example: estimate next-word probability for "The house is _____" using trigrams

$$p(\text{green} \mid \text{is, house, the}) = p(\text{green} \mid \text{is, house})$$

$$= \frac{\text{(no. of times "house is green" occurs in corpus)}}{\text{(no. of times "house is" occurs in corpus)}}$$

LANGUAGE MODELING: N-GRAM MODELS

The classic (pre-neural) approach: learning **n-gram** probabilities

Main idea: estimate next word probability using n-gram counts

$$p(w_{t+1} \mid w_t, \ldots, w_1) = p(w_{t+1} \mid w_t, \ldots, w_{t-n+2}) \approx \frac{\text{counts}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{counts}(w_t, \ldots, w_{t-n+2})}$$

Example: estimate next-word probability for "The house is _____" using trigrams

$$p(\text{green} \mid \text{is, house, the}) = p(\text{green} \mid \text{is, house})$$

$$= \frac{\text{(no. of times "house is green" occurs in corpus)}}{\text{(no. of times "house is" occurs in corpus)}}$$

N-gram models with small n are "miopic", what about large n?

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

$$e_t = Ex_t$$

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding
(e.g. Word2Vec)

$$e_t = Ex_t$$

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding
(e.g. Word2Vec)

$$e_t = Ex_t$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding
(e.g. Word2Vec)

A non-linear
activation function

$$e_t = Ex_t$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding
(e.g. Word2Vec)

A non-linear
activation function

$$e_t = Ex_t$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding
(e.g. Word2Vec)

A non-linear
activation function

$$e_t = Ex_t$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

Predicted class
(in this case, next word)

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

For every t:

Word Embedding (e.g. Word2Vec)

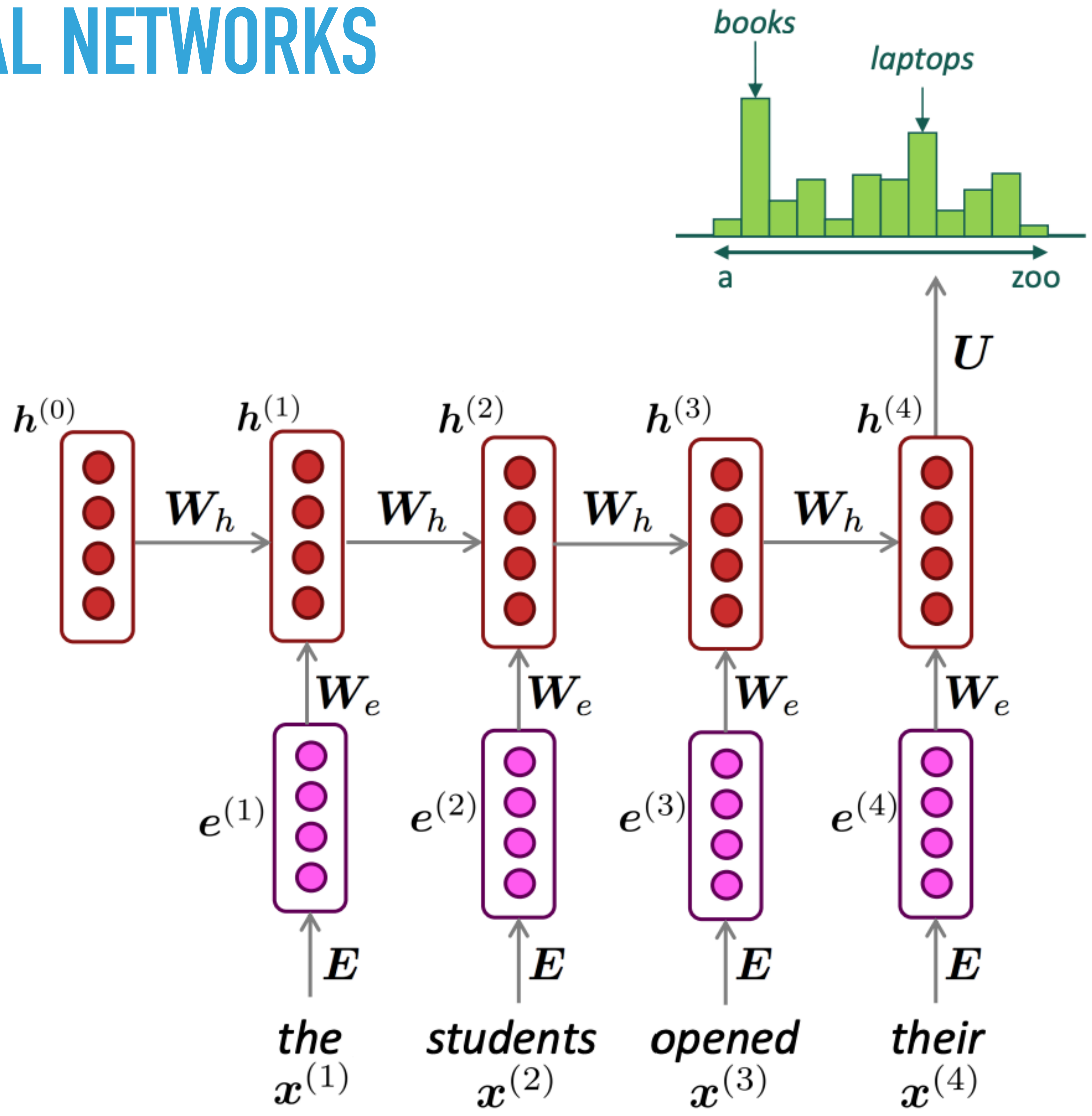
$$e_t = Ex_t$$

A non-linear activation function

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

Predicted class (in this case, next word)

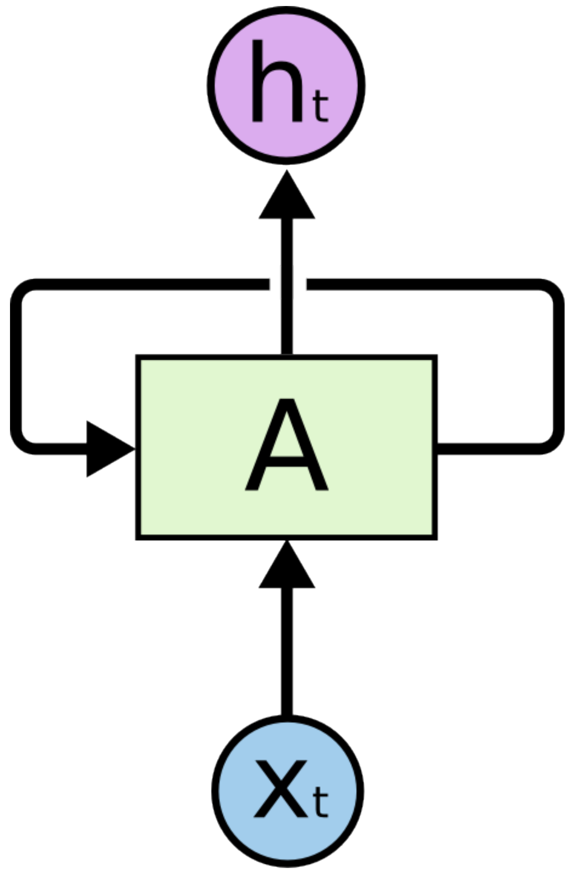


Credit: Stanford CS224n

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

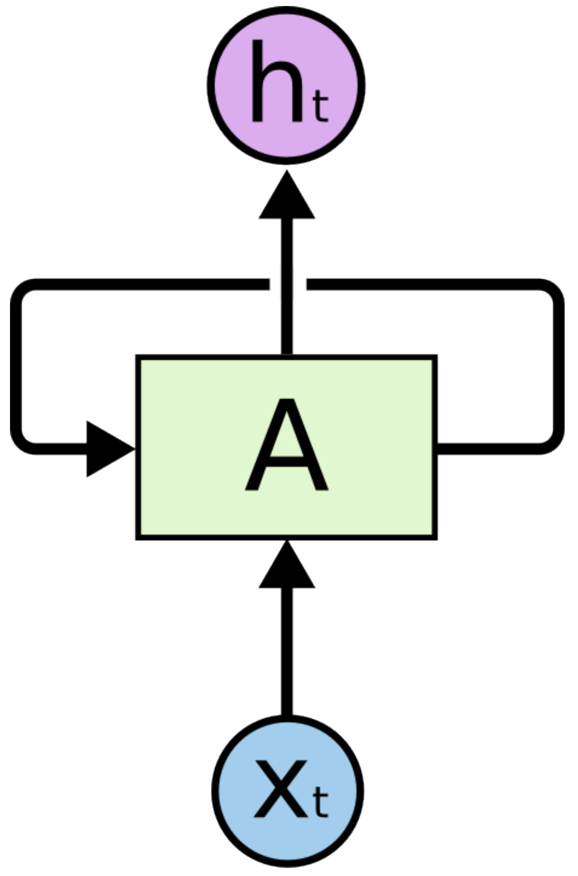
$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$



LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

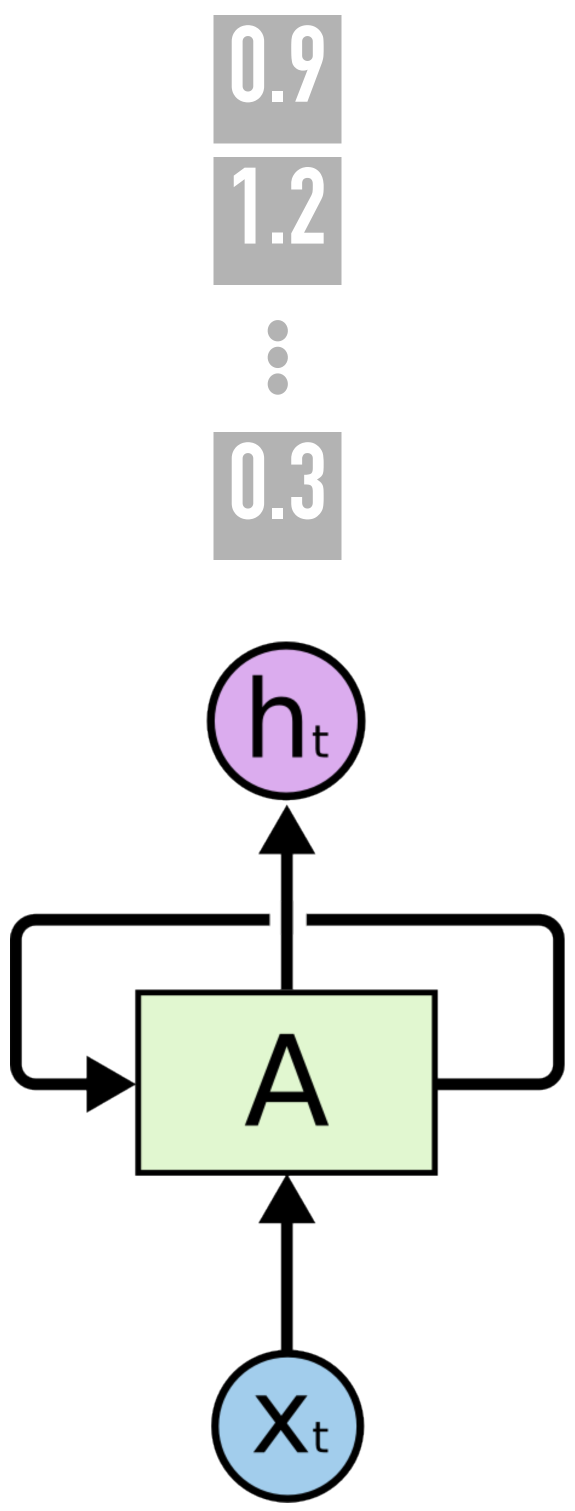


The house is green

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

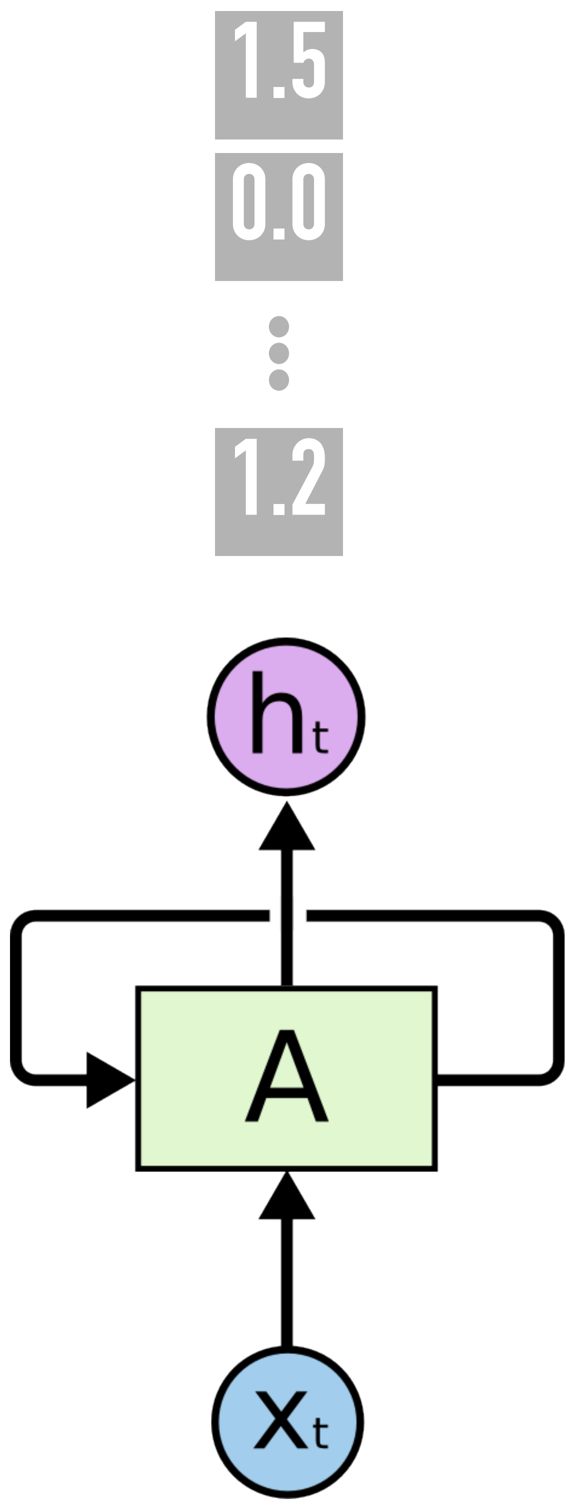


The house is green

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

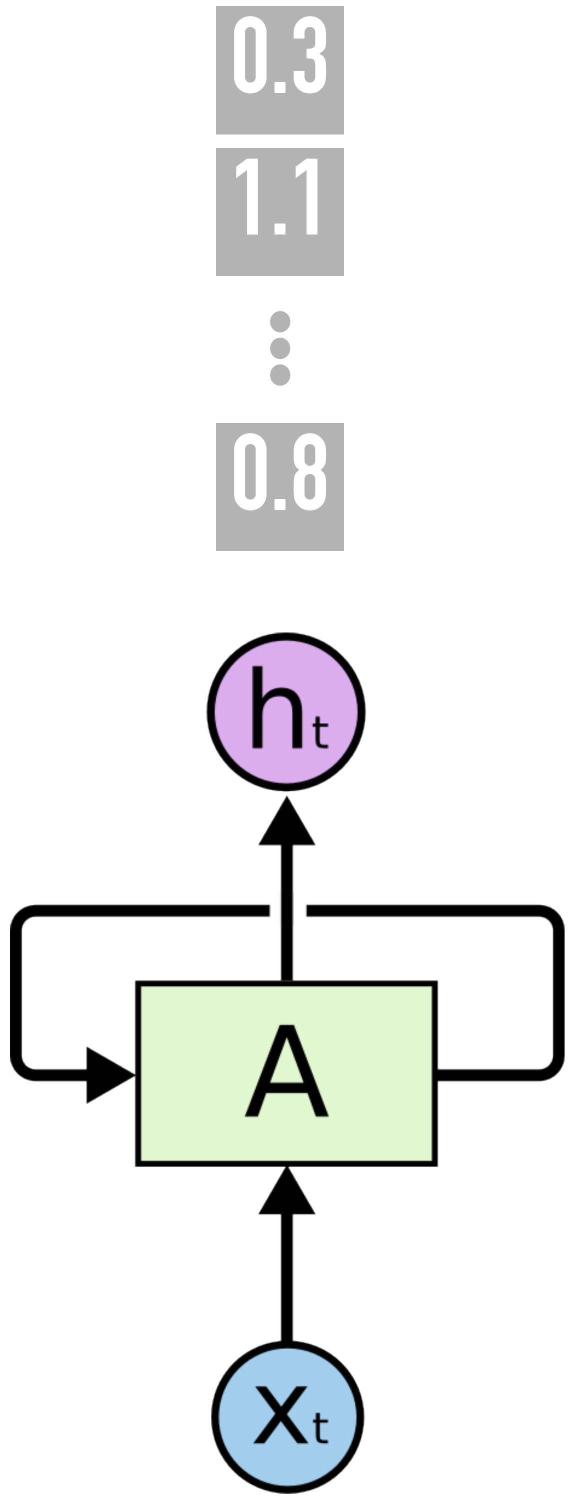


The house is green

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

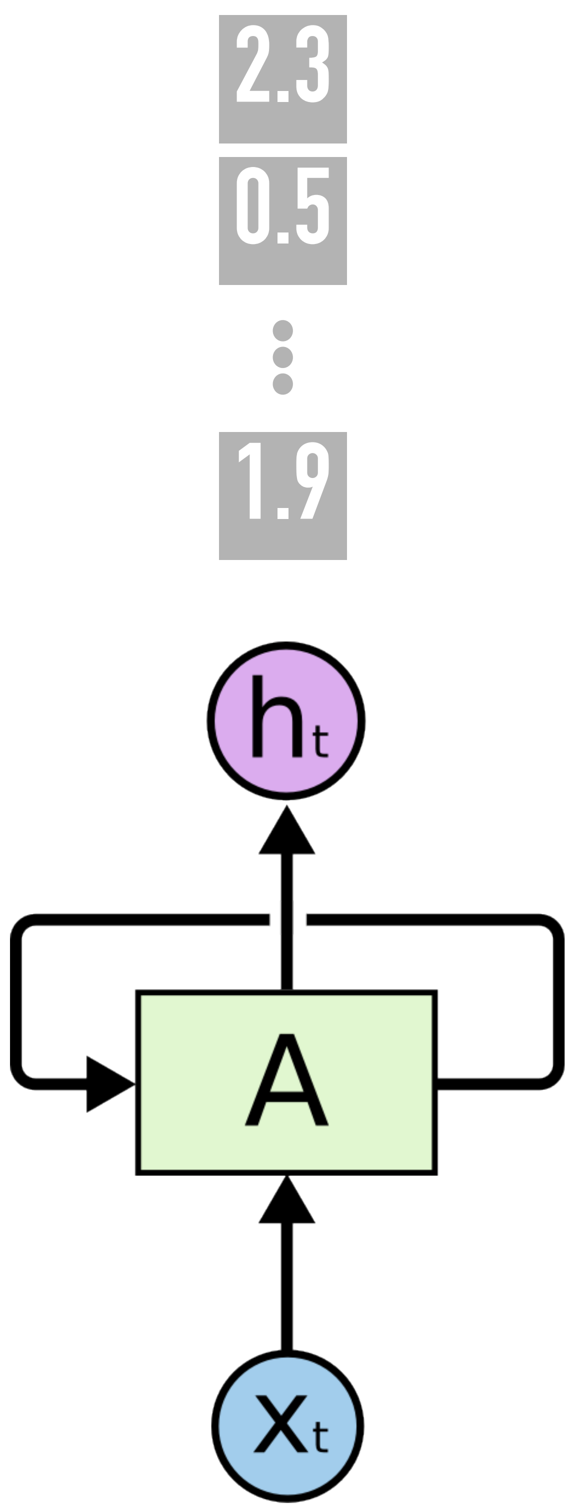


The house is green

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

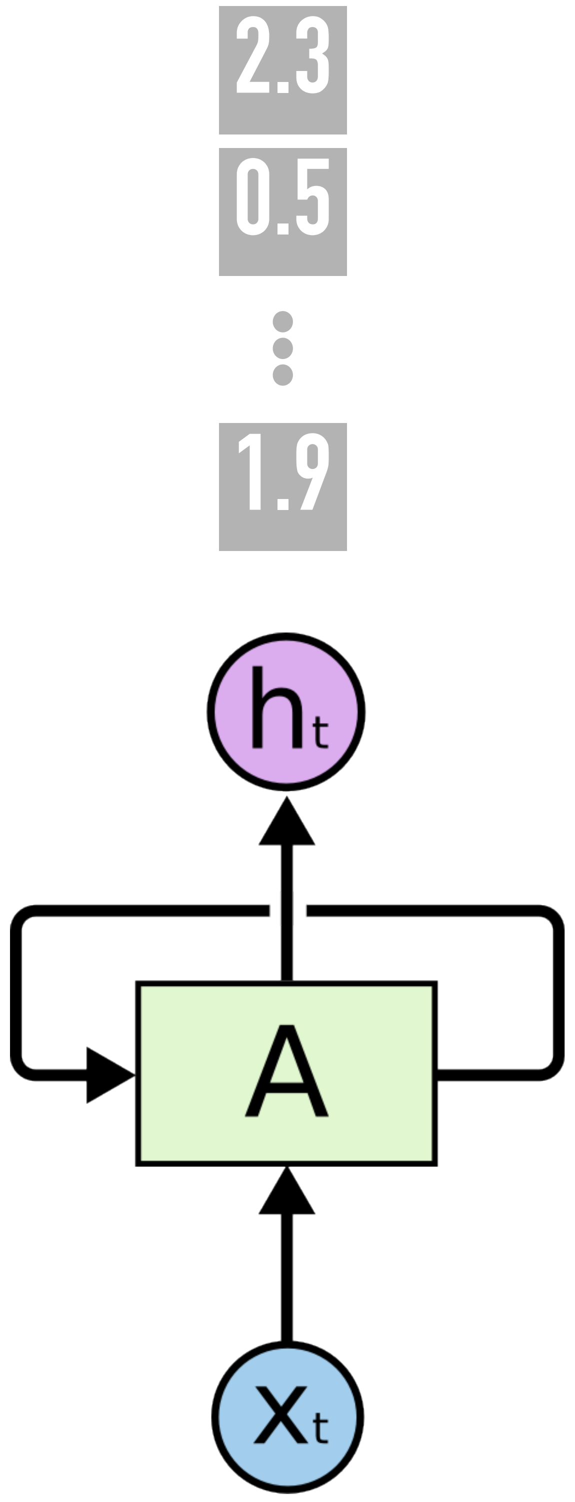


The house is green

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]

$$A: \quad h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

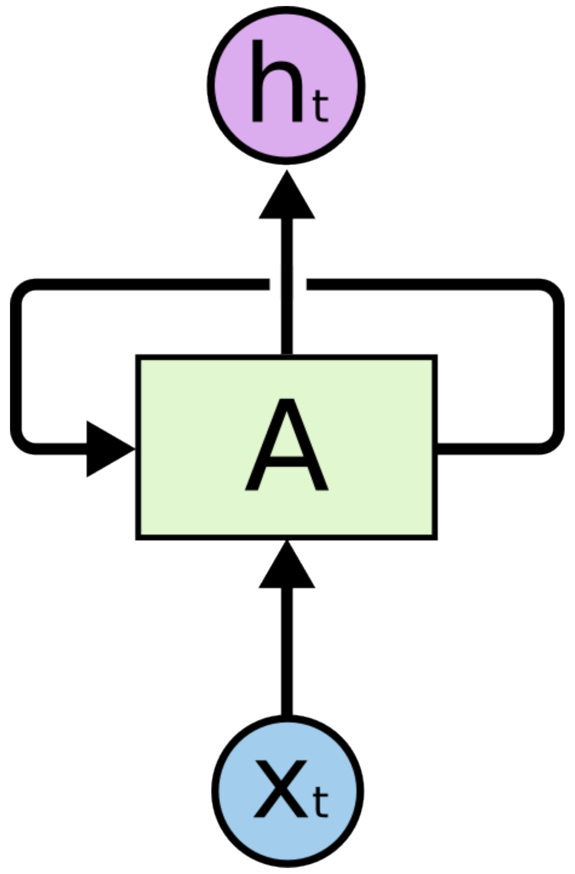


Note: same weights applied every time

The house is green

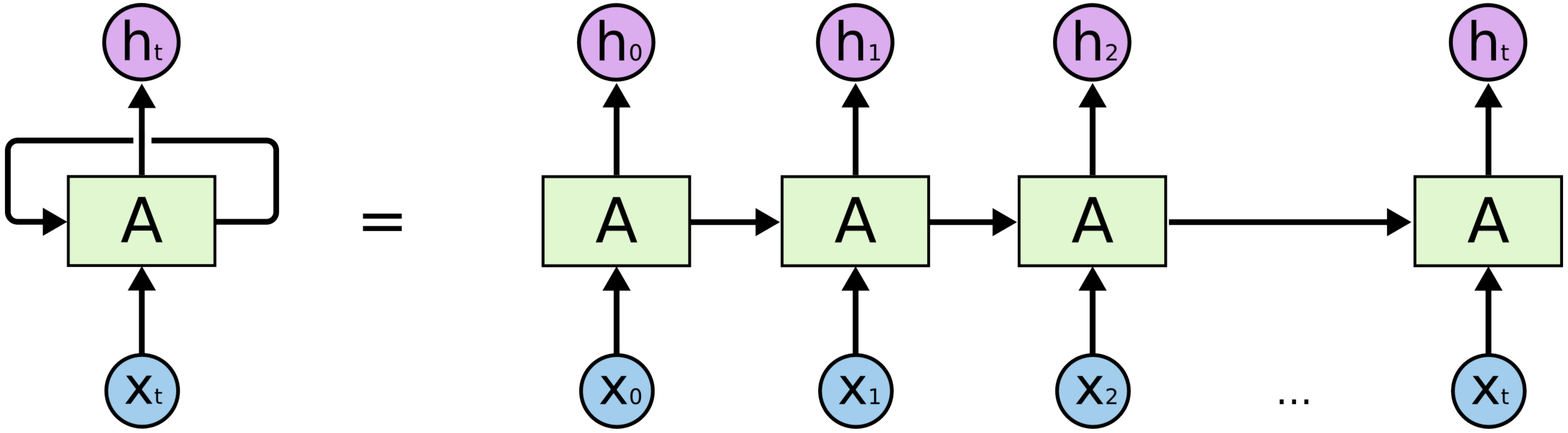
LANGUAGE MODELING: RECURRENT NEURAL NETWORKS

[Rumelhart, 1986; Hopfield, 1982]



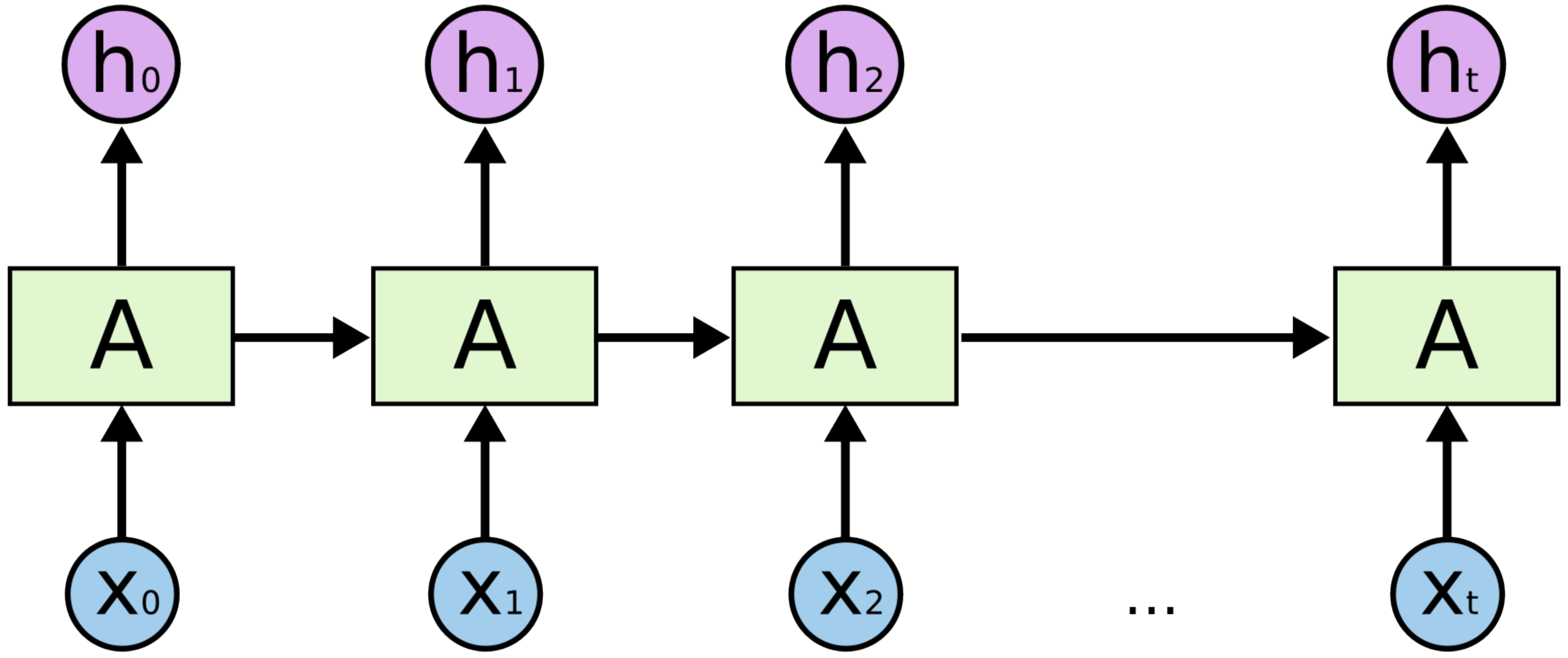
How do we train it?

LANGUAGE MODELING: RECURRENT NEURAL NETWORKS
[Rumelhart, 1986; Hopfield, 1982]

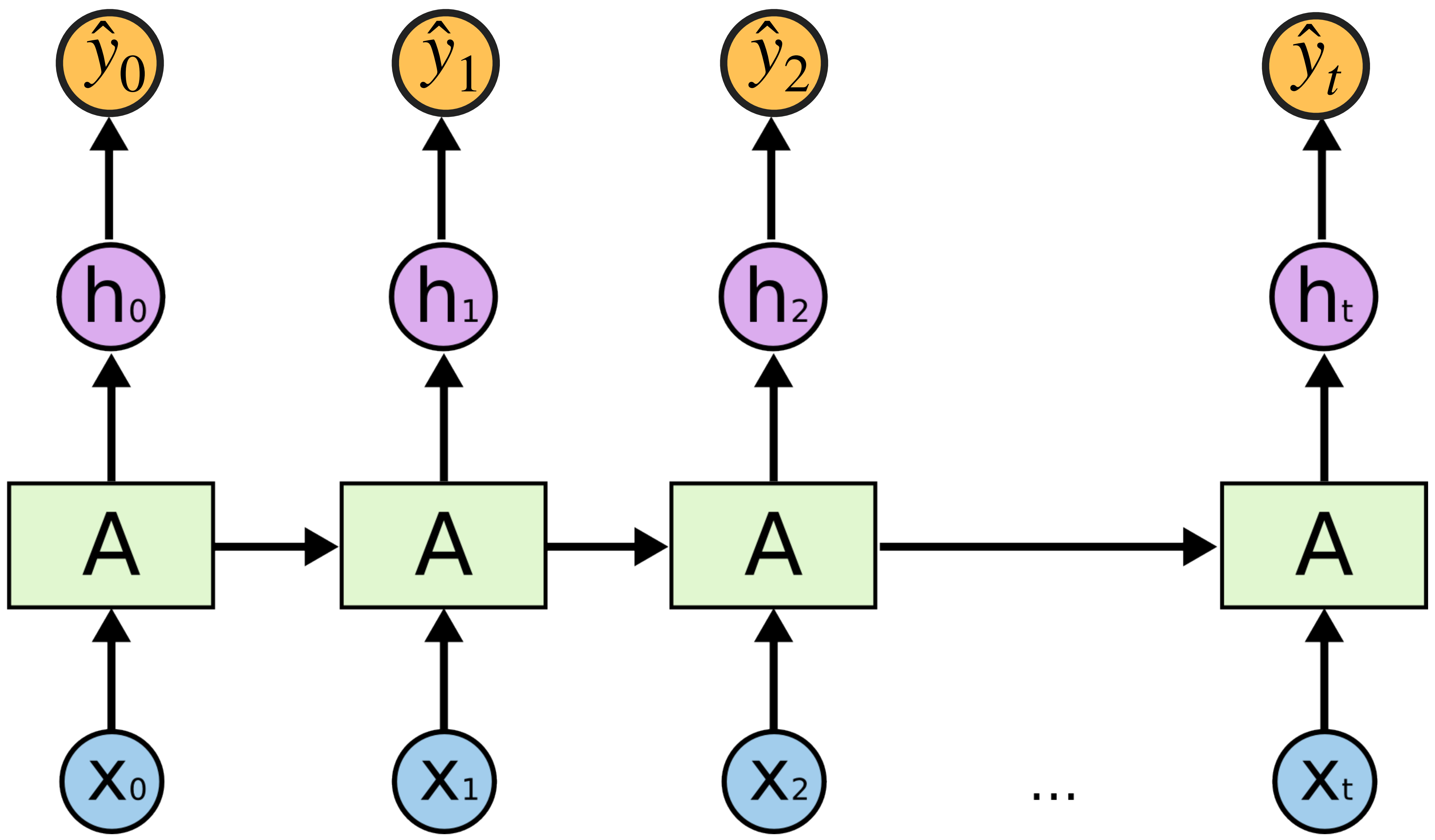


How do we train it?

RECURRENT NEURAL NETS: TRAINING



RECURRENT NEURAL NETS: TRAINING

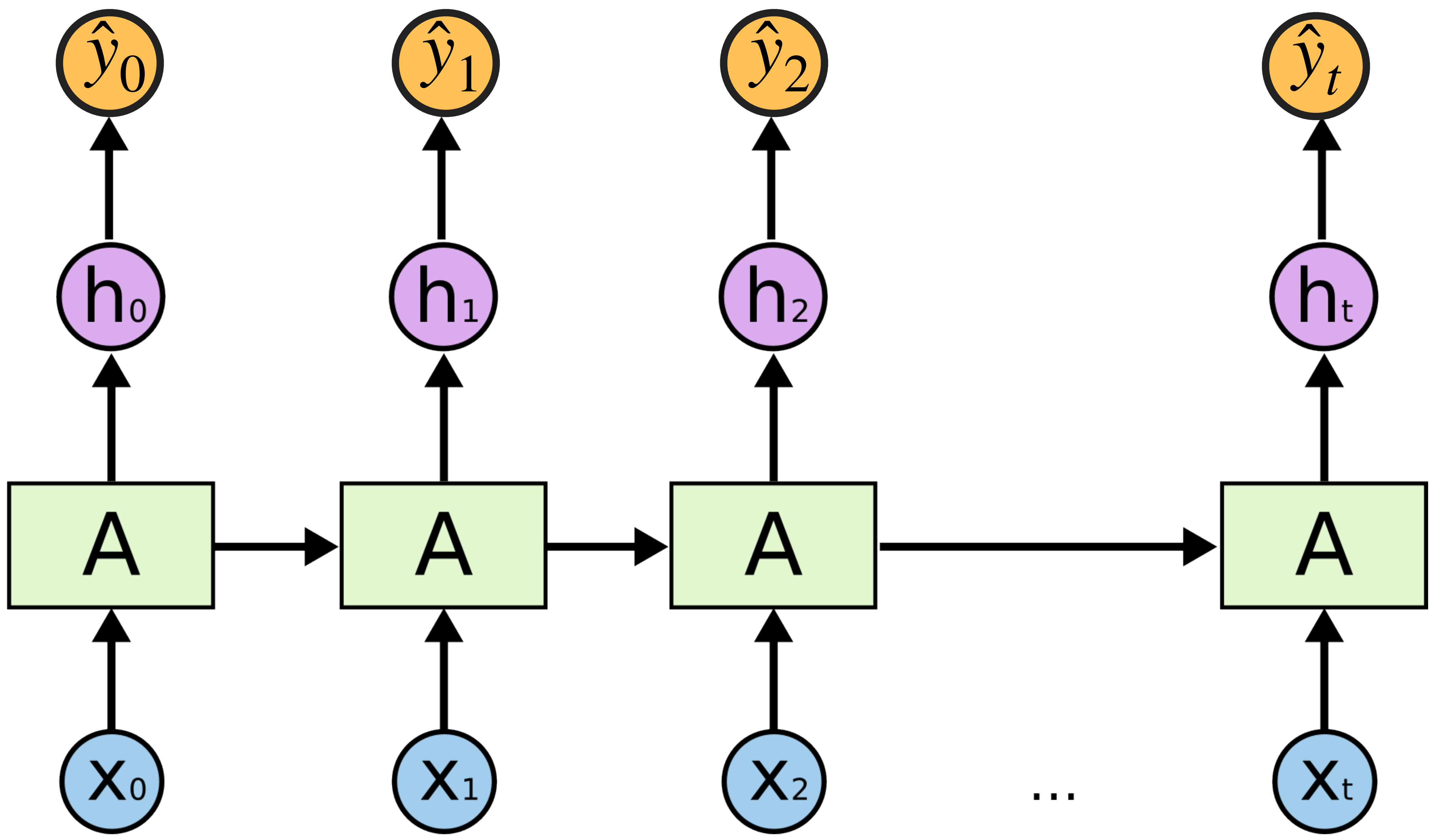


RECURRENT NEURAL NETS: TRAINING

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$e_t = Ex_t$$

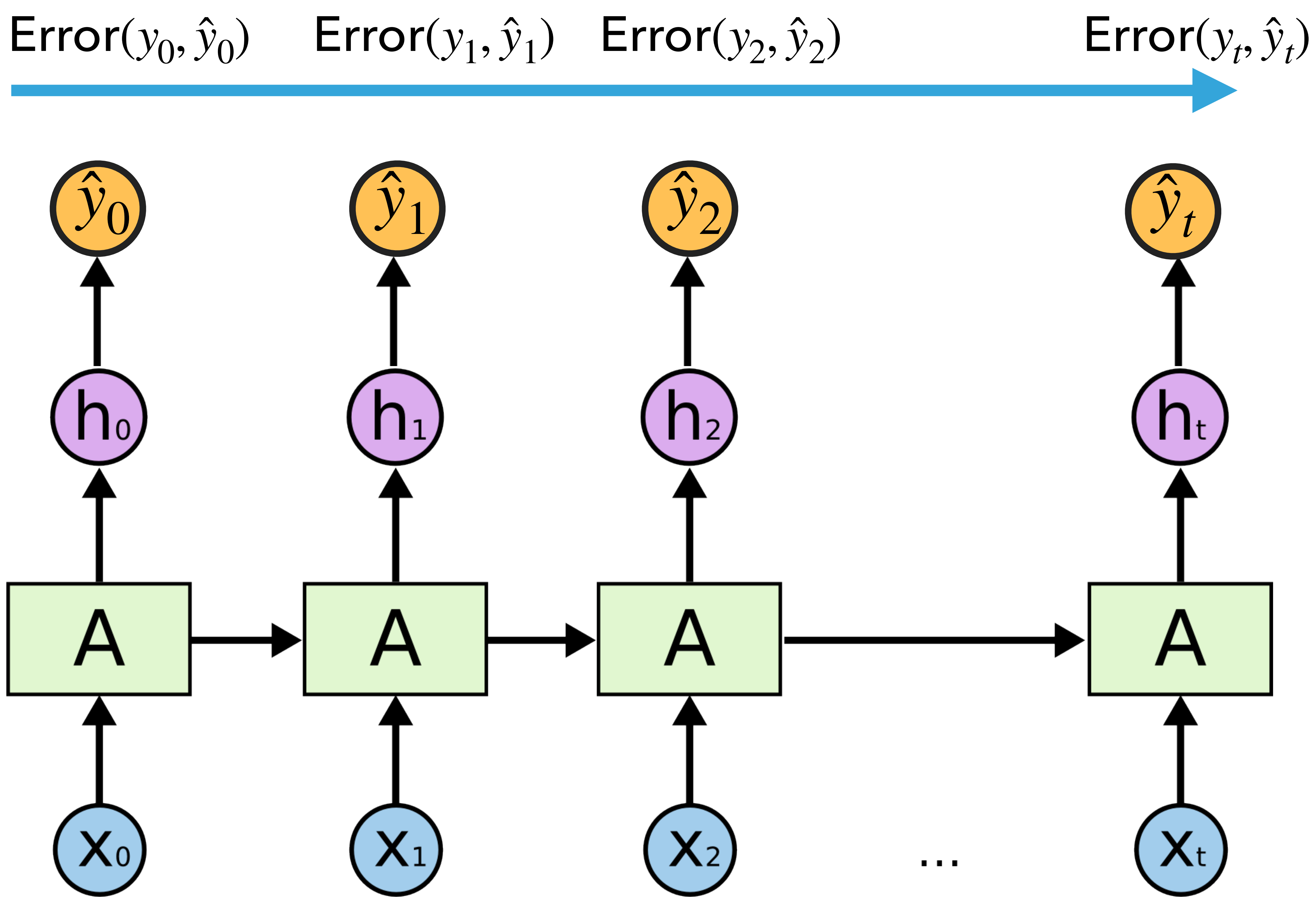


RECURRENT NEURAL NETS: TRAINING

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$e_t = Ex_t$$

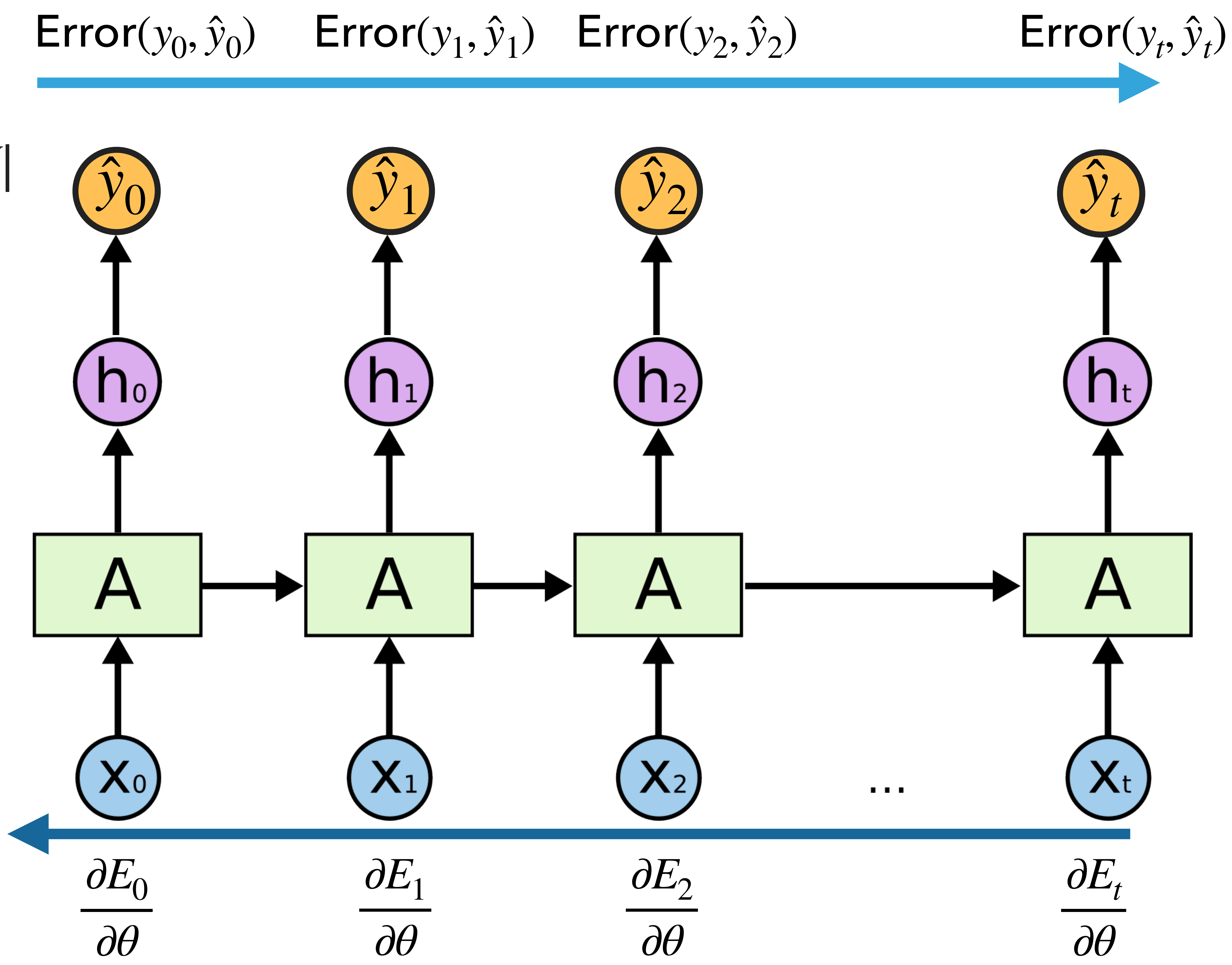


RECURRENT NEURAL NETS: TRAINING

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$e_t = Ex_t$$

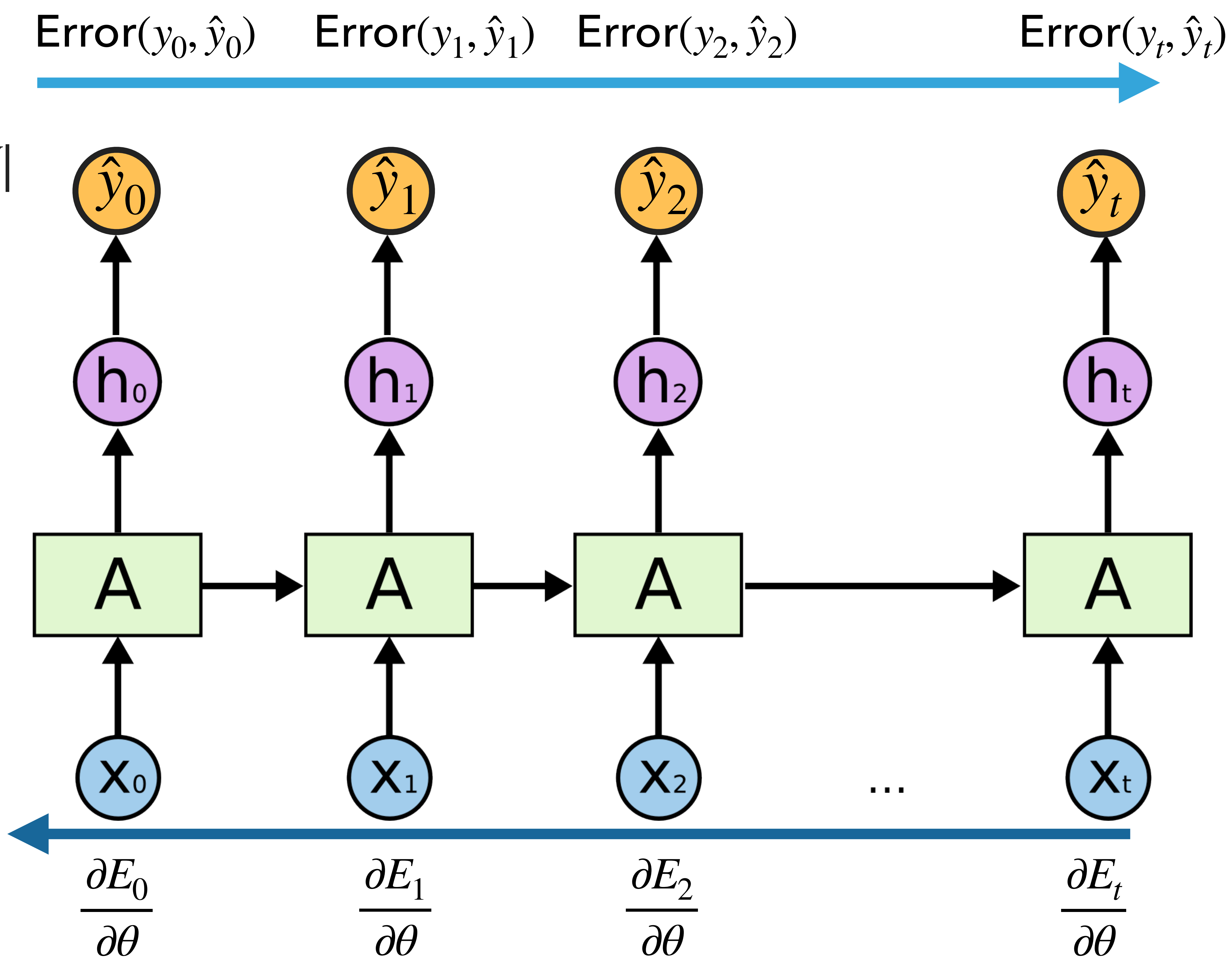


RECURRENT NEURAL NETS: TRAINING

$$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in R^{|V|}$$

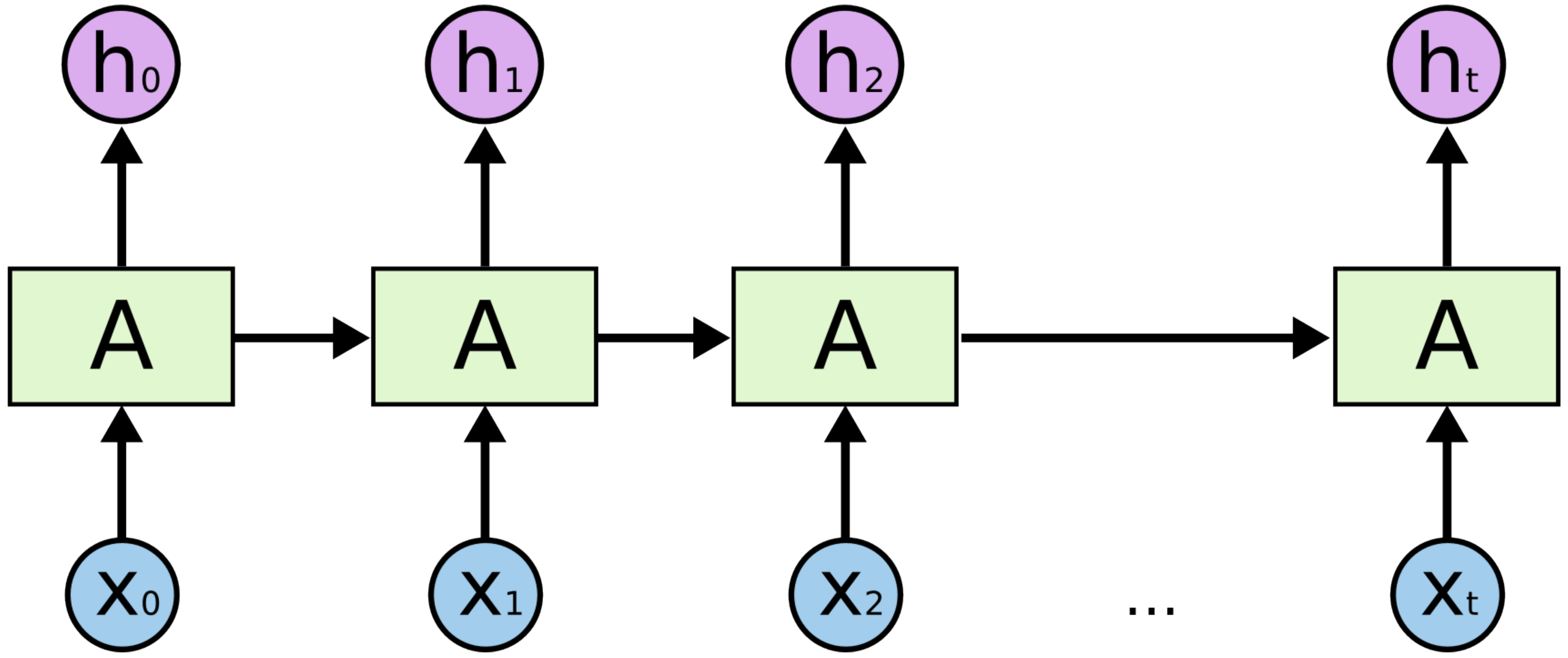
$$h_t = \sigma(W_e e_t + W_h h_{t-1} + b_1)$$

$$e_t = Ex_t$$



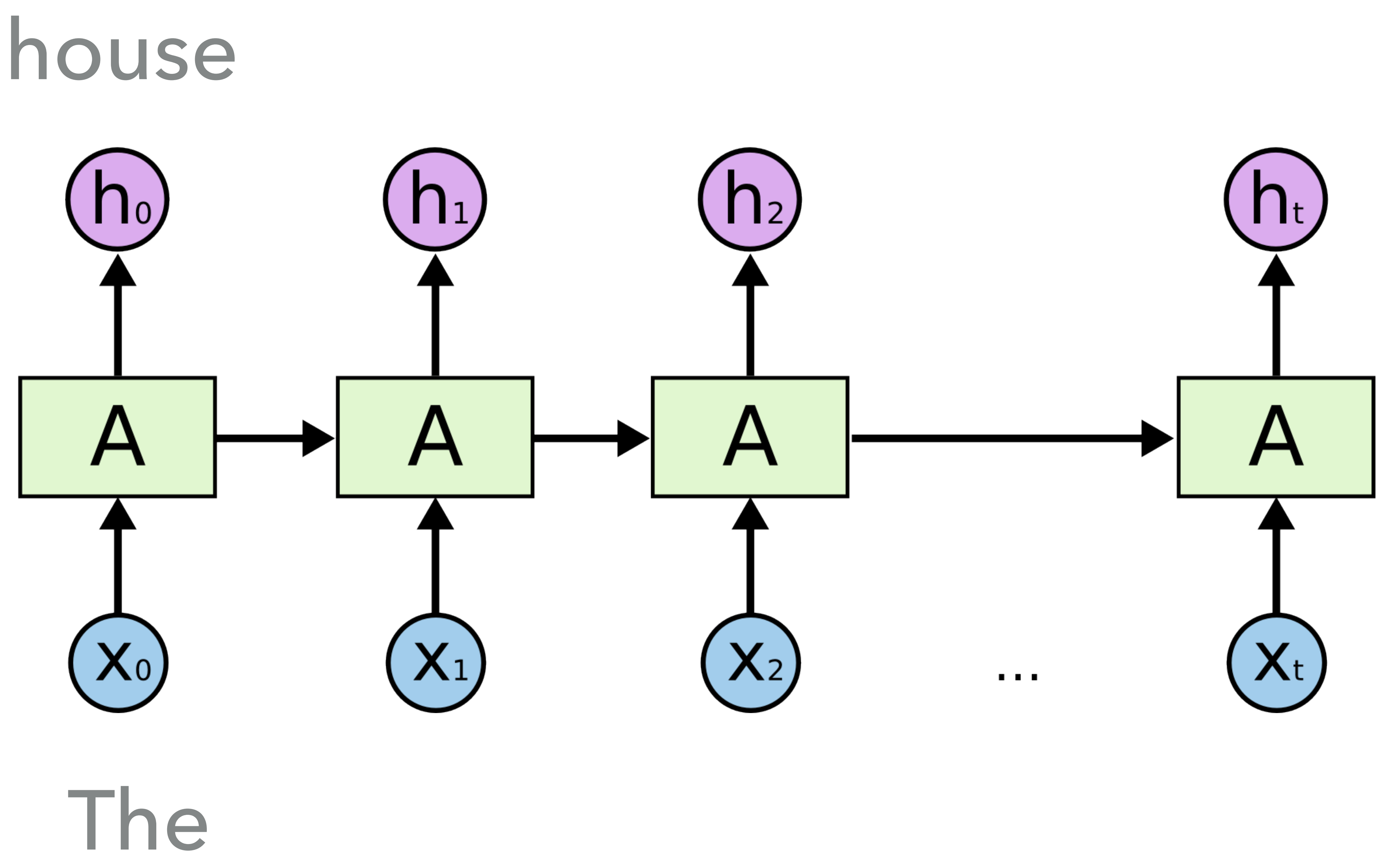
Back-Propagation Through Time (BPTT)

RECURRENT NEURAL NETS: APPLICATIONS



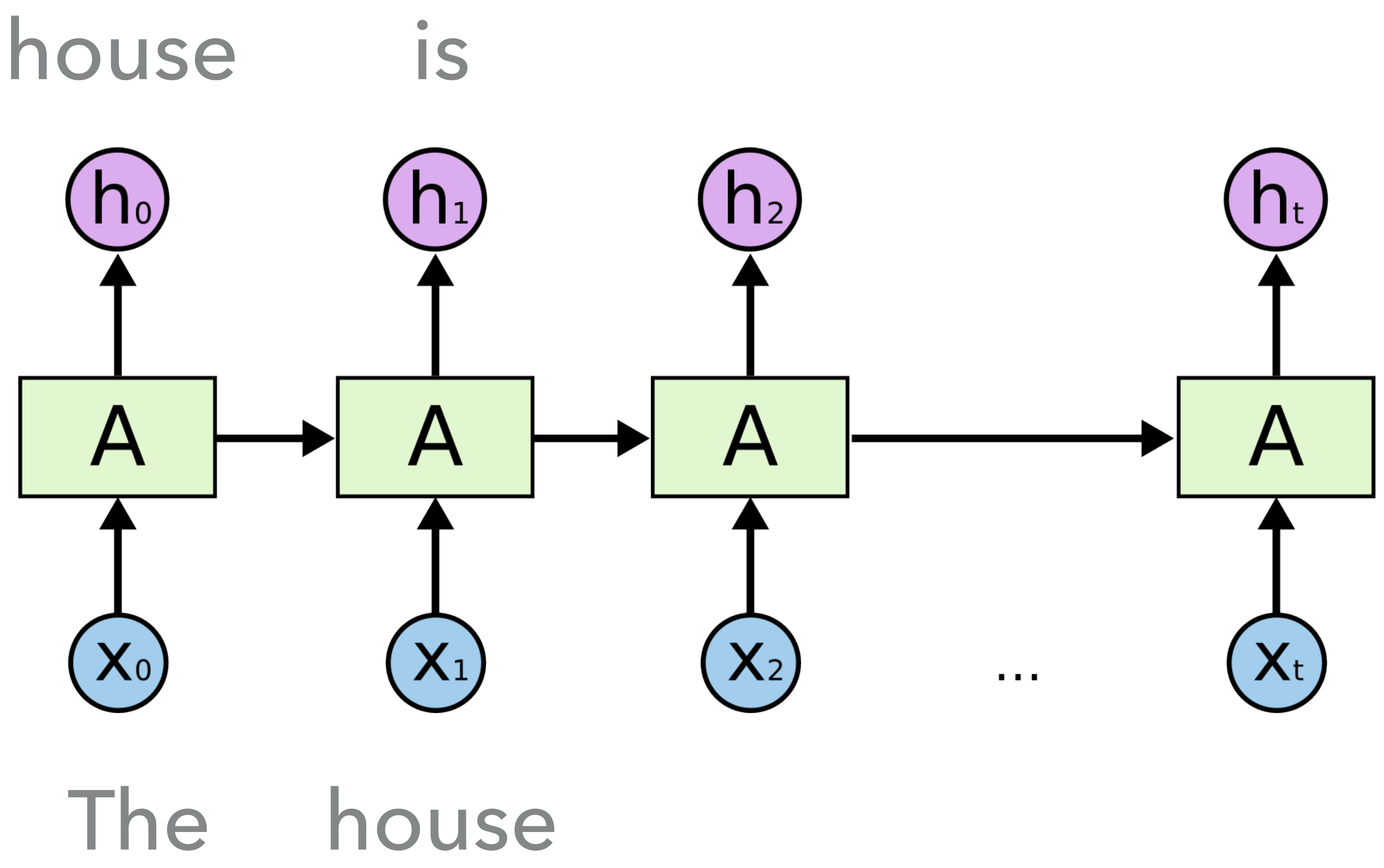
Language Modeling (LM)

RECURRENT NEURAL NETS: APPLICATIONS



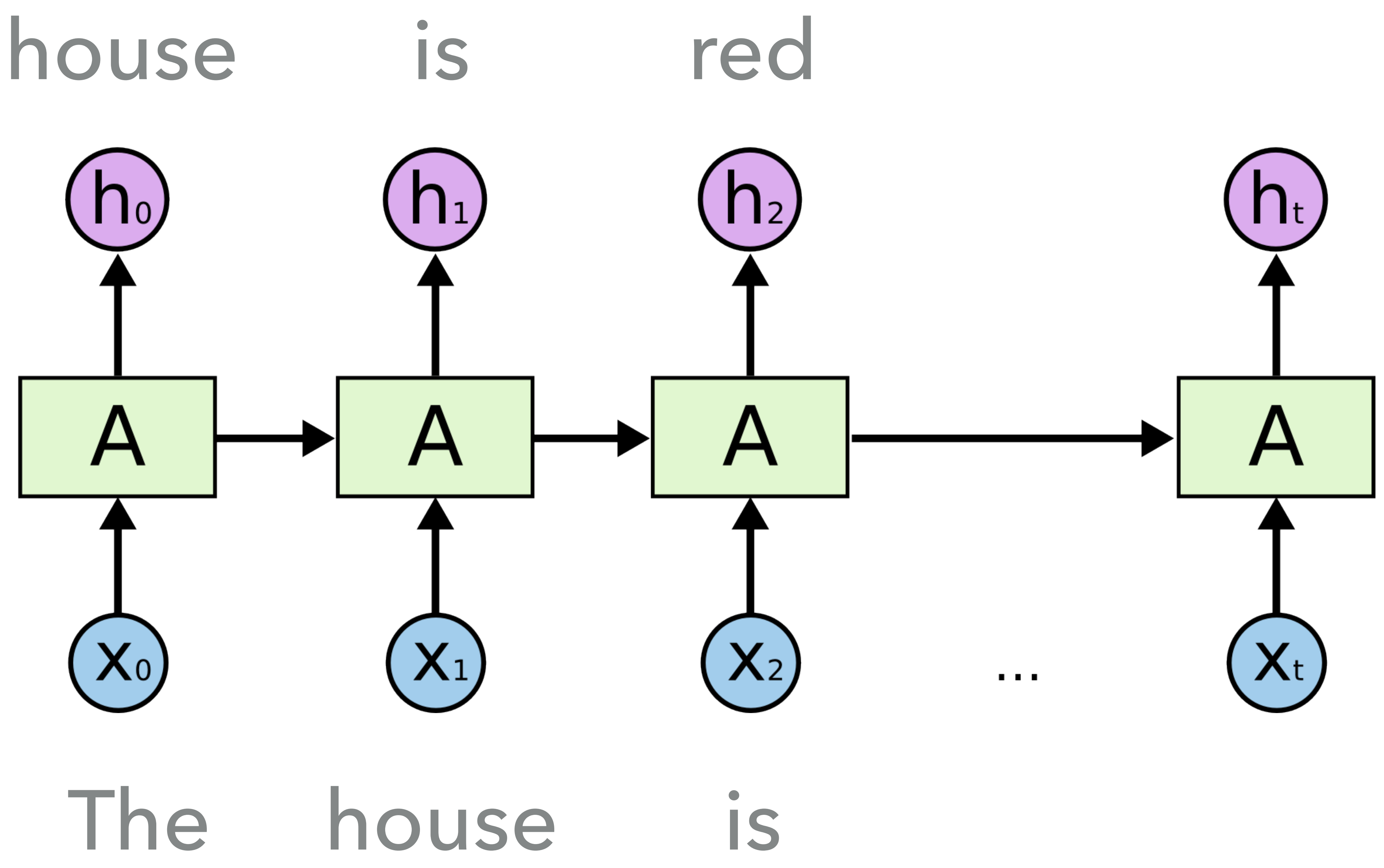
Language Modeling (LM)

RECURRENT NEURAL NETS: APPLICATIONS



Language Modeling (LM)

RECURRENT NEURAL NETS: APPLICATIONS



Language Modeling (LM)

RECURRENT NEURAL NETS: EXAMPLE TEXT GENERATION

[Sutskever et al, 2011]

Training data: Wikipedia

Input: "The meaning of life is "

Generated Text:

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. In the show's agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning, written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population. Mar??a Nationale, Kelli, Zedlat-Dukastoe, Florendon, Ptu's thought is. To adapt in most parts of North America, the dynamic fairy Dan please believes, the free speech are much related to the

Training data: NYT

Input: "[space]"

Generated Text:

while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a half-suit defending the Bharatiya Fernall 's office . Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where many civil rights defined , officials said early that forms , " said Bernard J. Marco Jr. of Pennsylvania , was monitoring New York

RECURRENT NEURAL NETS: EXAMPLE TEXT GENERATION

Example Source: medium.com/@samim/

Training data: Obama's Speeches

Input: "YES WE CAN"

Generated Text:

Good morning. And as we mark the fact that they can stand with their companies that are consistent to the state of Pakistan and the United States of America.

With the financial system we can do that. And the people of the United States will not be able to continue to support the people of the greatest problem of the American people to stay in the White House. And that's why […]

Thank you very much. God bless you. God bless you. God bless you. God bless you.

Example Source: http://karpathy.github.io/

Training data: Shakespeare

Input: " "

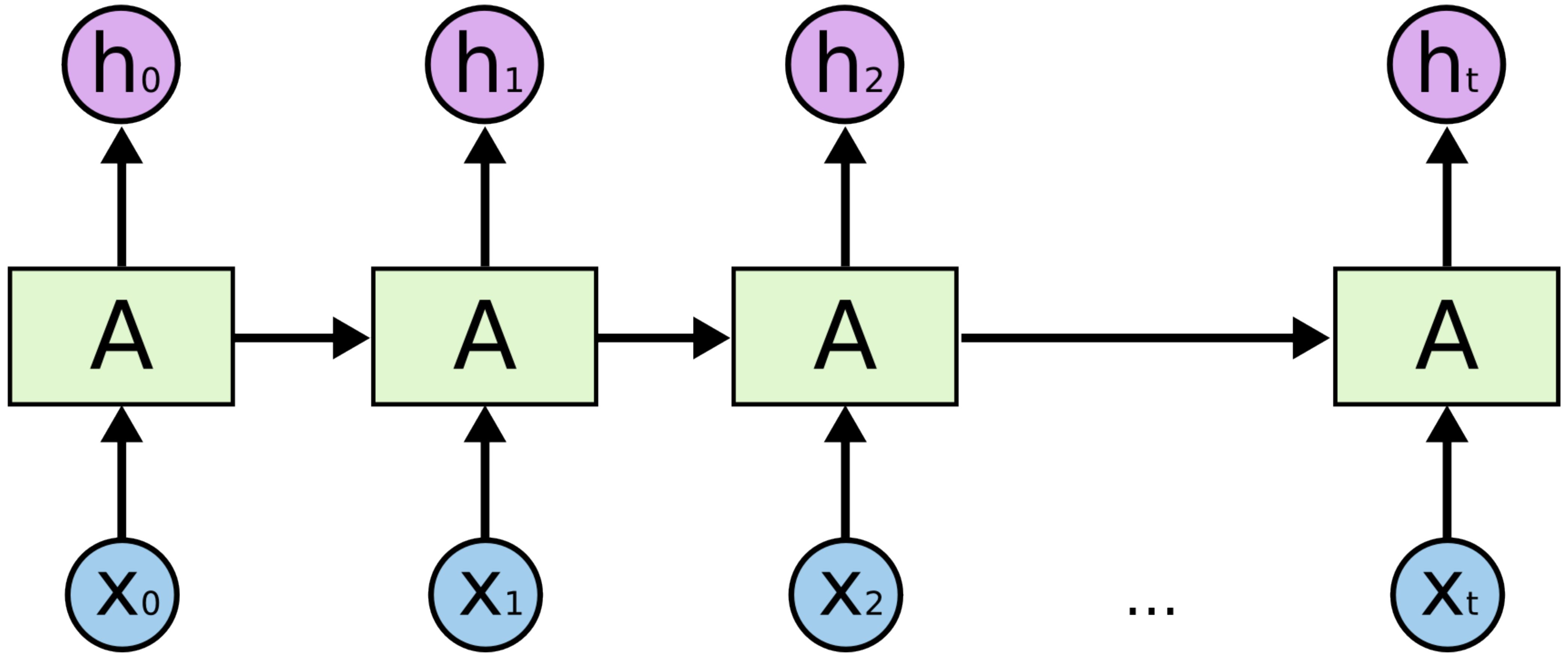
Generated Text:

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

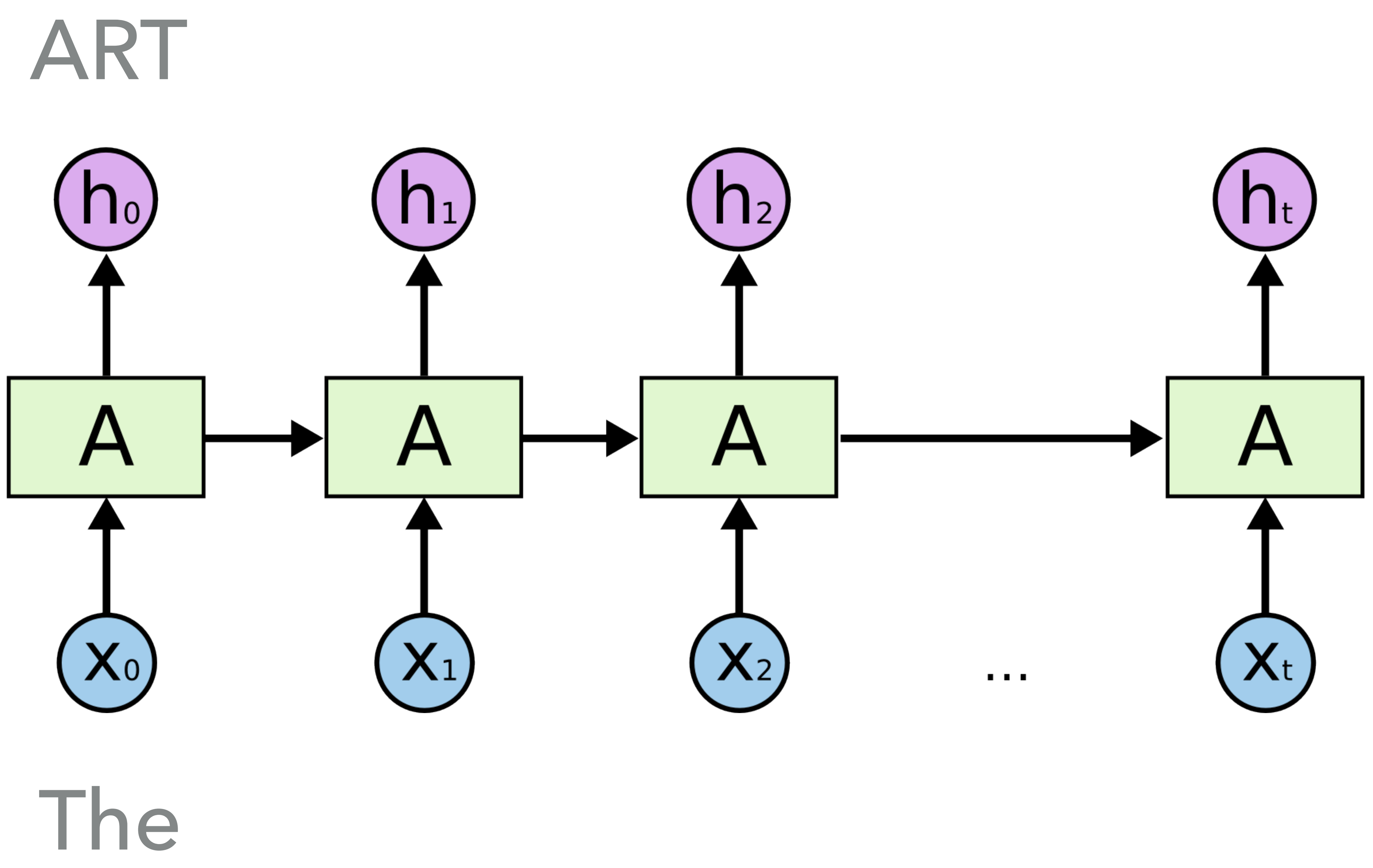
DUKE VINCENTIO:
Well, your wit is in the care of side and that. […]

RECURRENT NEURAL NETS: APPLICATIONS



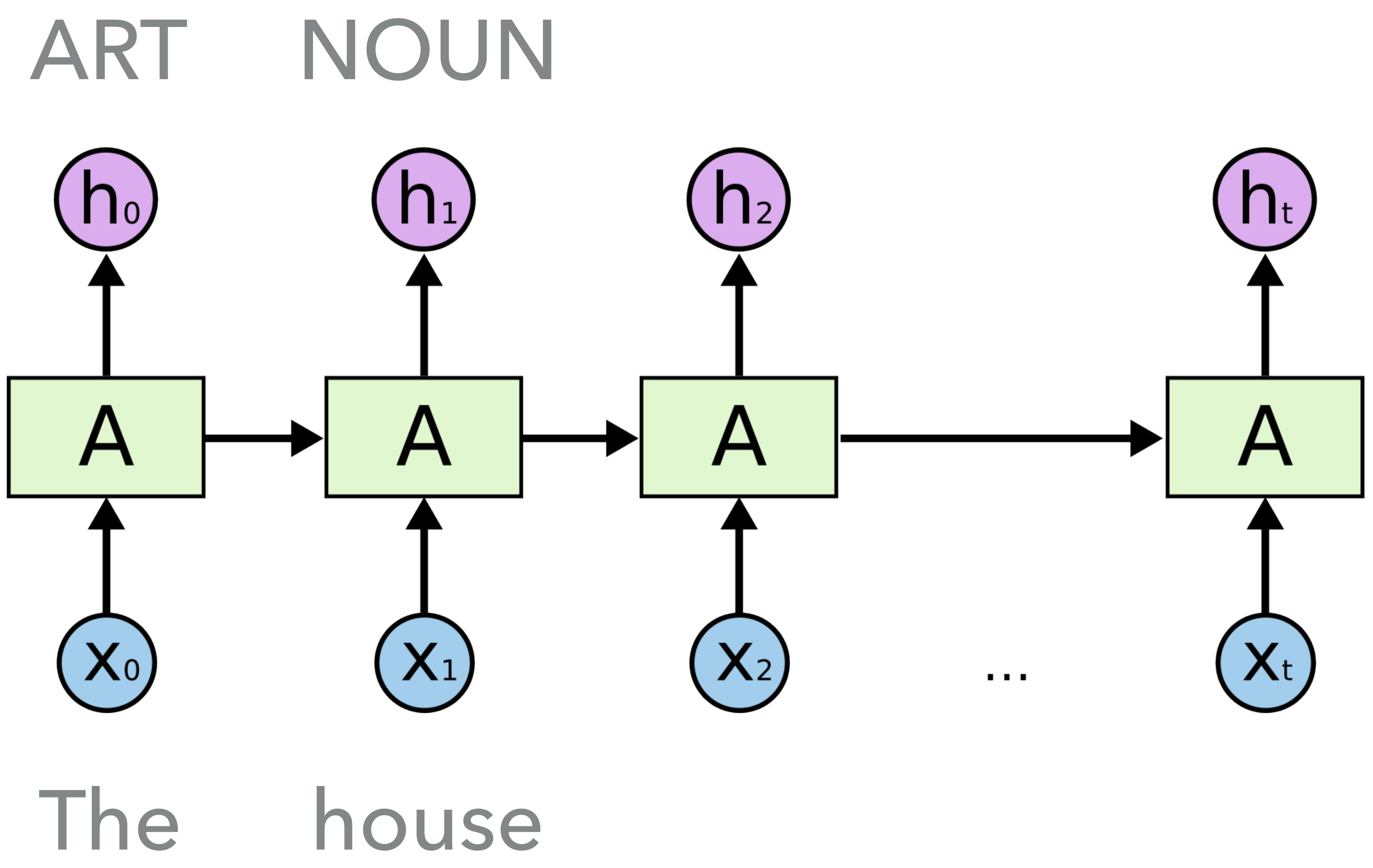
Part-of-Speech Tagging (POS)

RECURRENT NEURAL NETS: APPLICATIONS



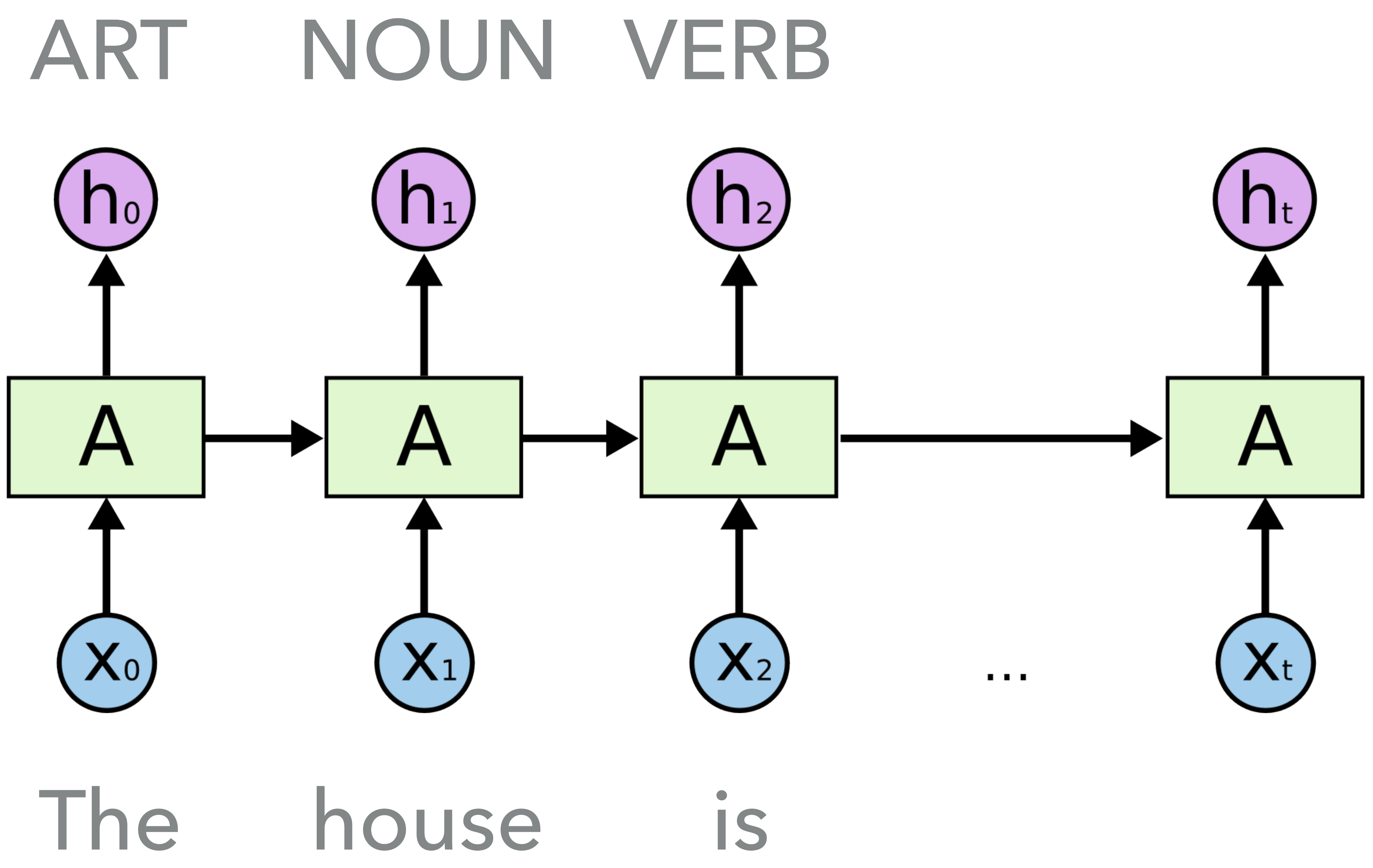
Part-of-Speech Tagging (POS)

RECURRENT NEURAL NETS: APPLICATIONS



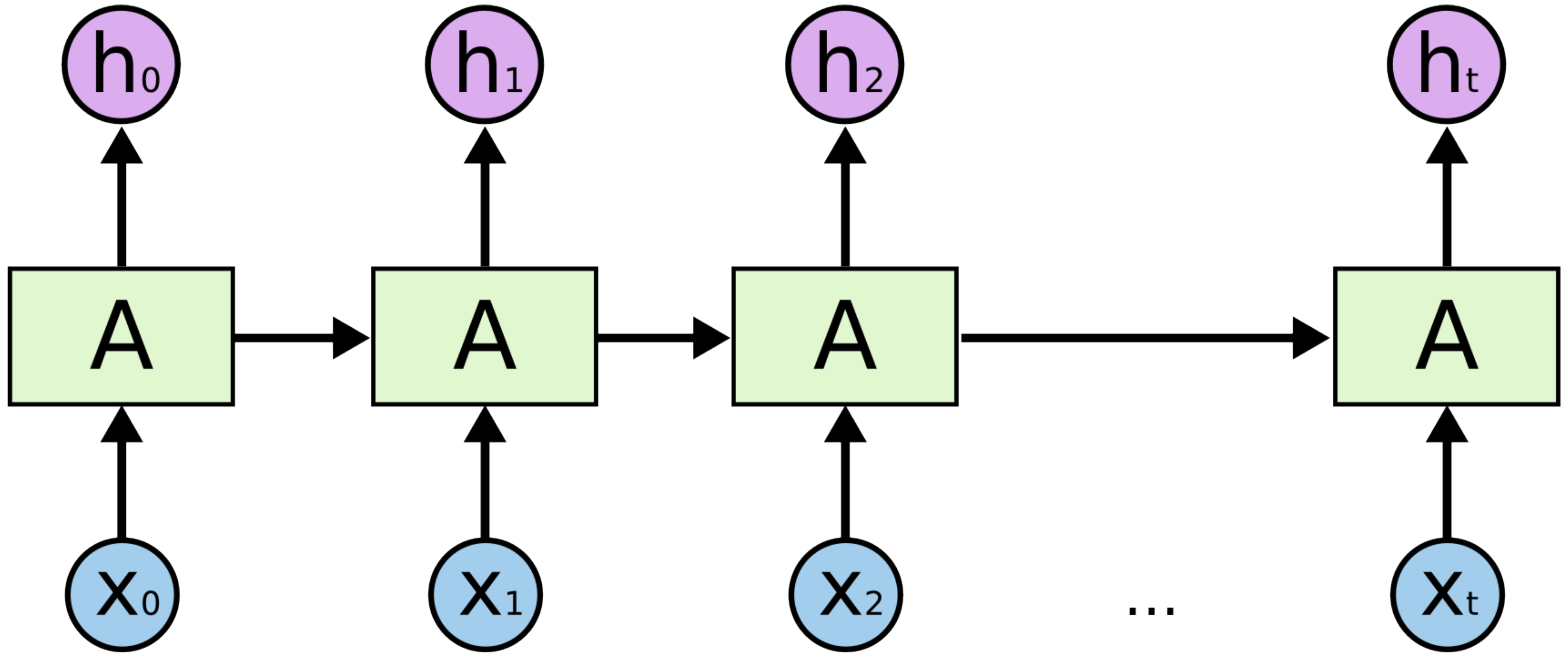
Part-of-Speech Tagging (POS)

RECURRENT NEURAL NETS: APPLICATIONS



Part-of-Speech Tagging (POS)

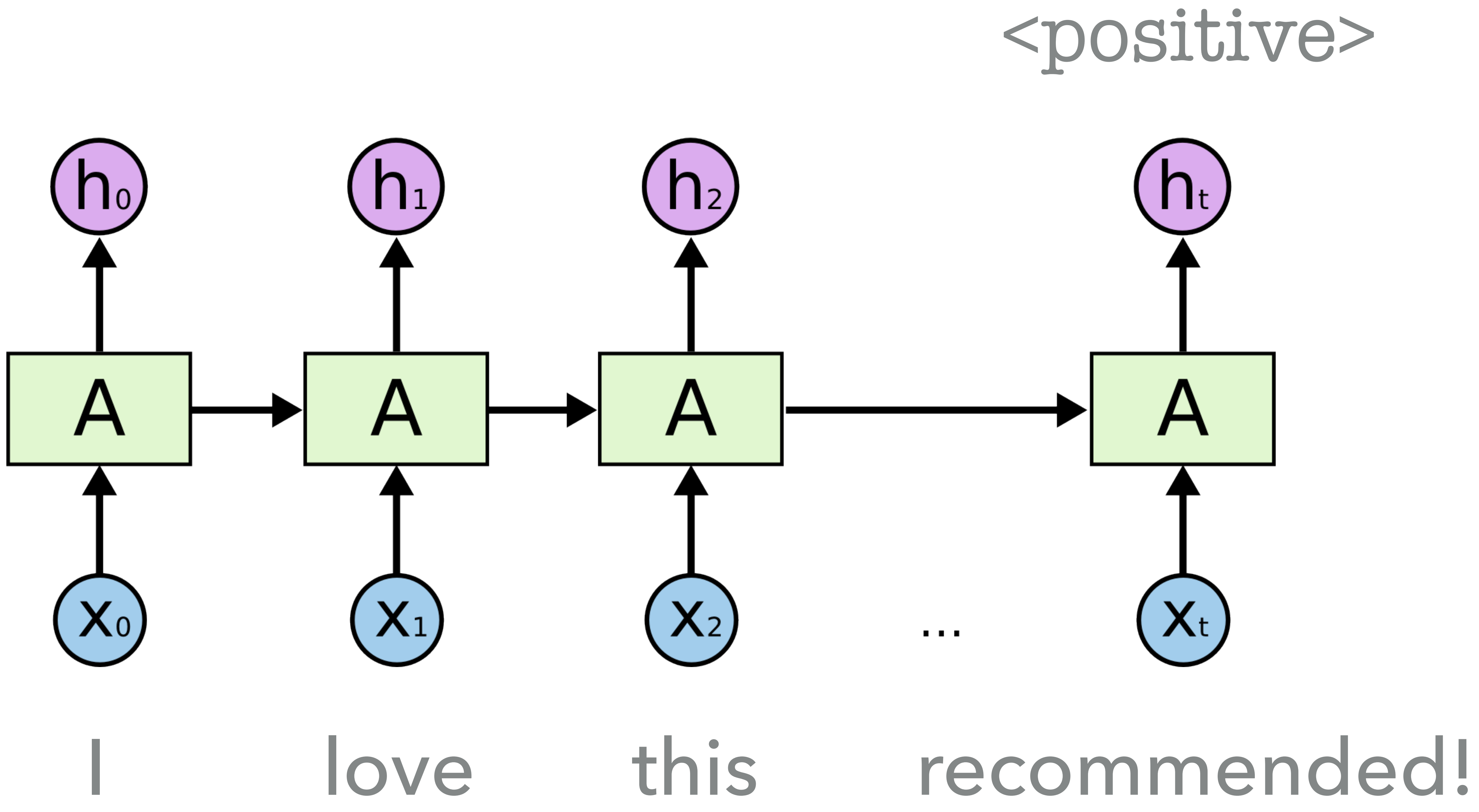
RECURRENT NEURAL NETS: APPLICATIONS



Sentence Classification

(e.g. sentiment analysis)

RECURRENT NEURAL NETS: APPLICATIONS



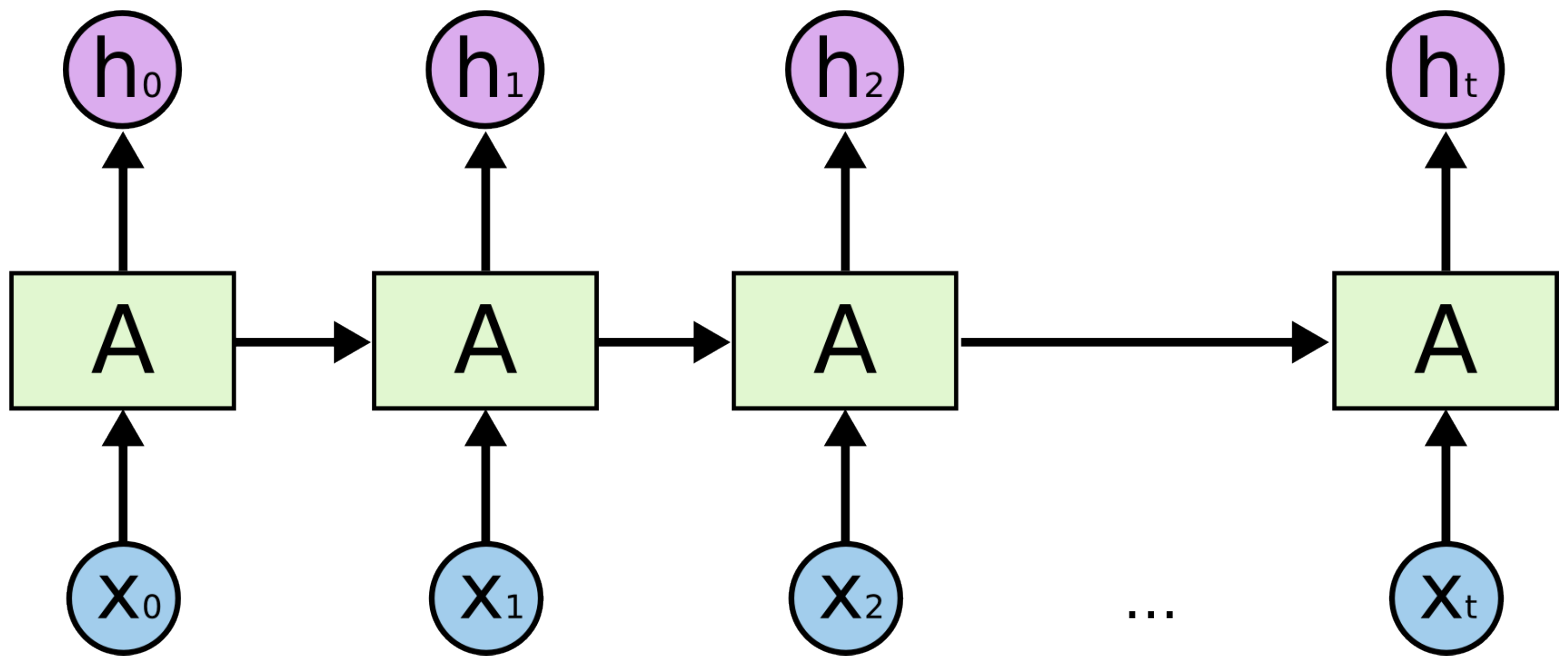
Sentence Classification

(e.g. sentiment analysis)

RECURRENT NEURAL NETS: PROS AND CONS

PROS

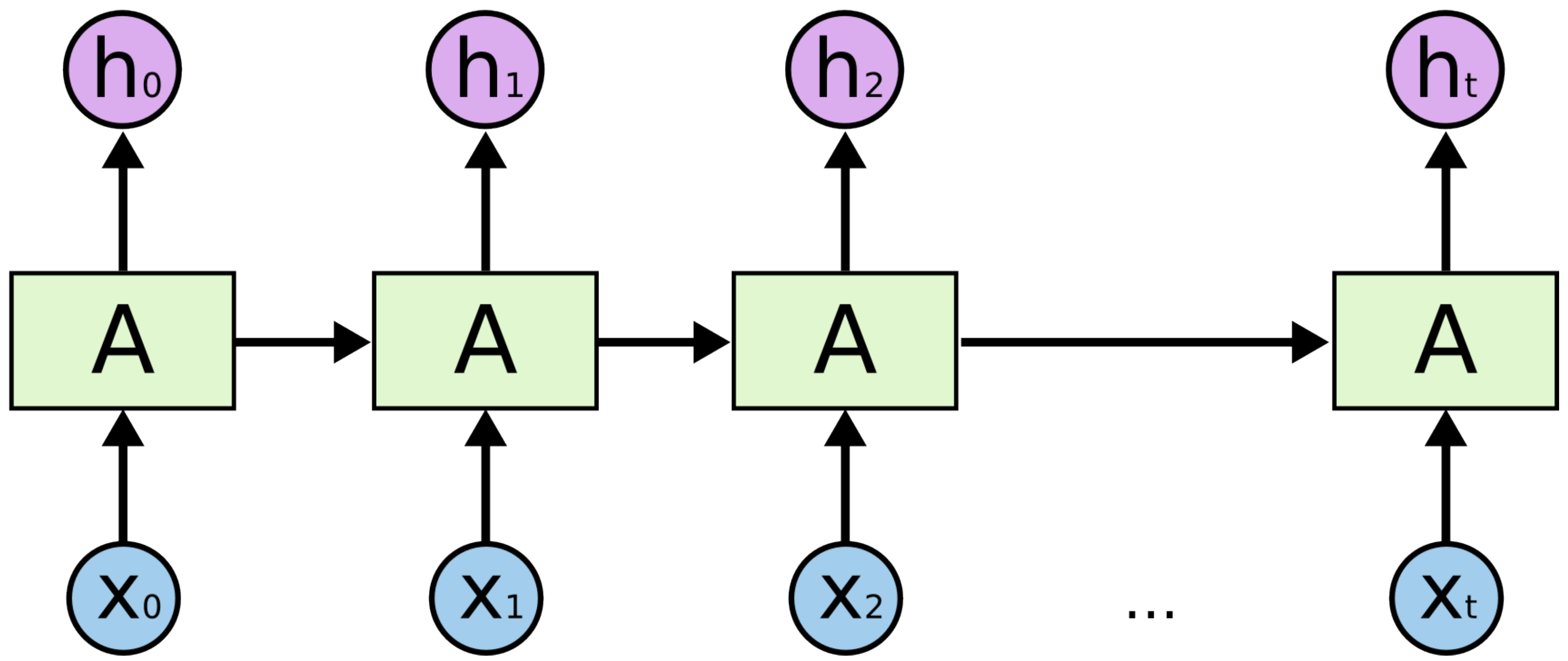
CONS



RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length

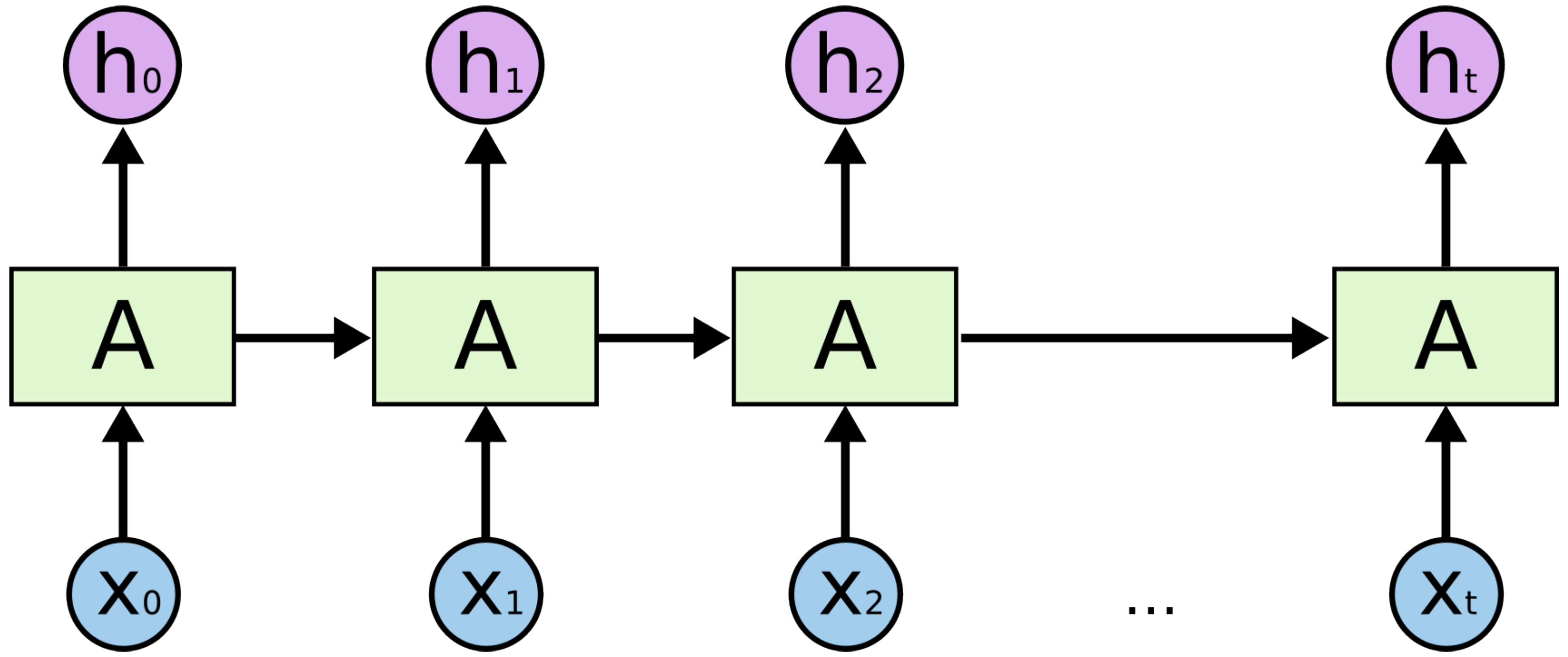


CONS

RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length
- Can model long-range dependencies

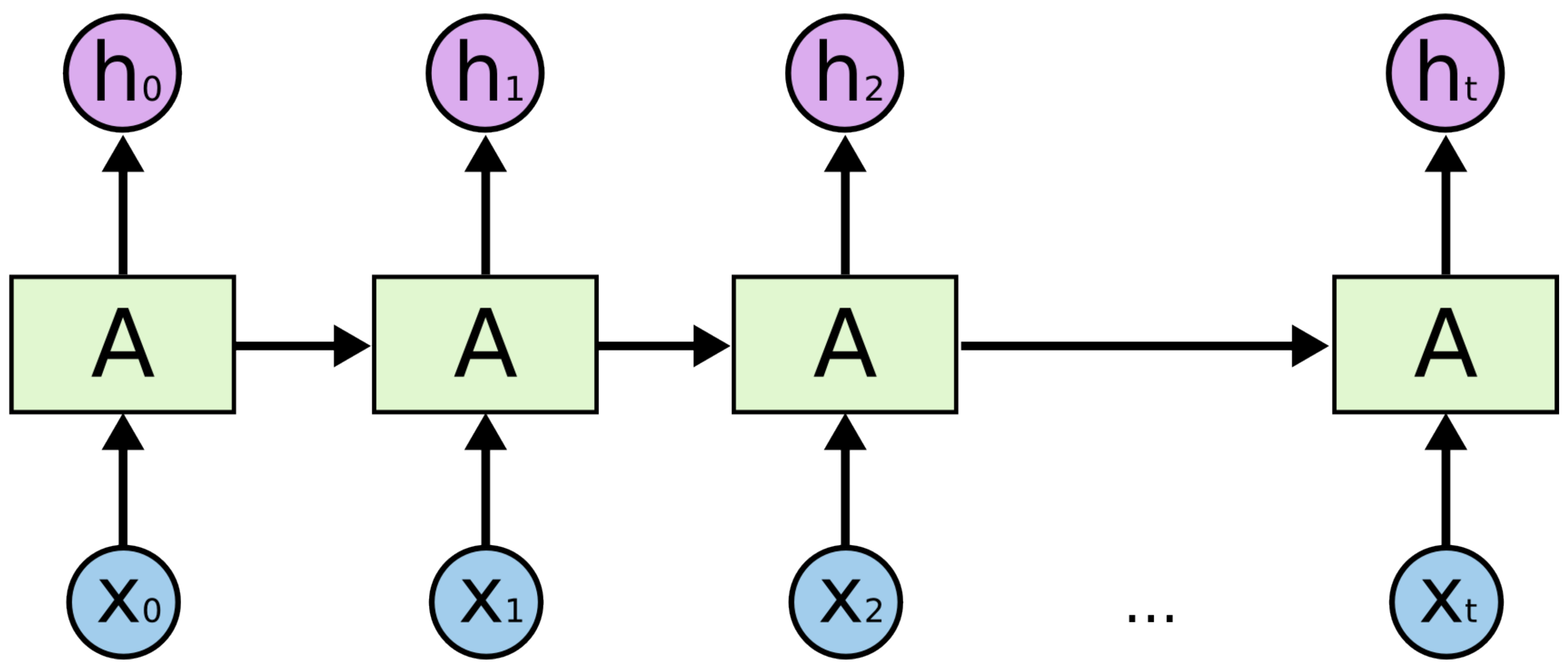


CONS

RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length
- Can model long-range dependencies
- Fixed model size regardless of input size

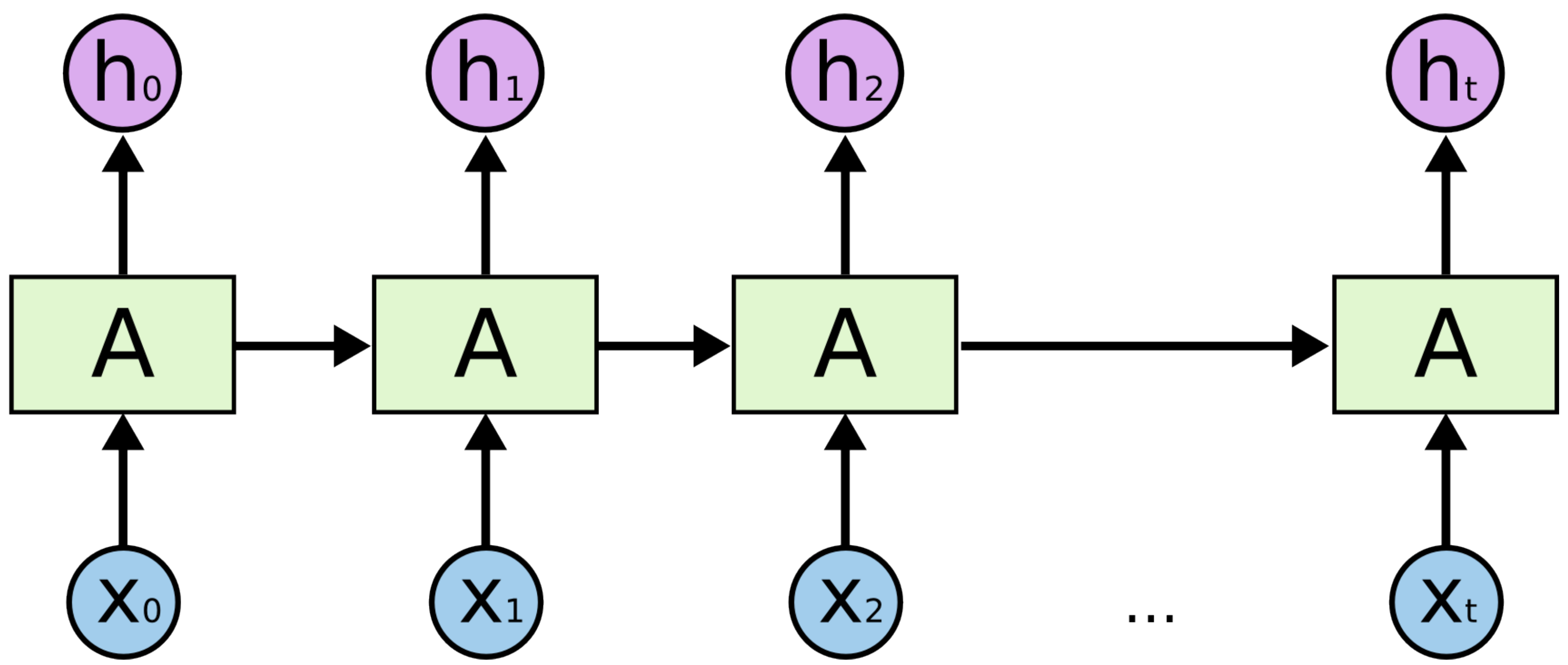


CONS

RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length
- Can model long-range dependencies
- Fixed model size regardless of input size



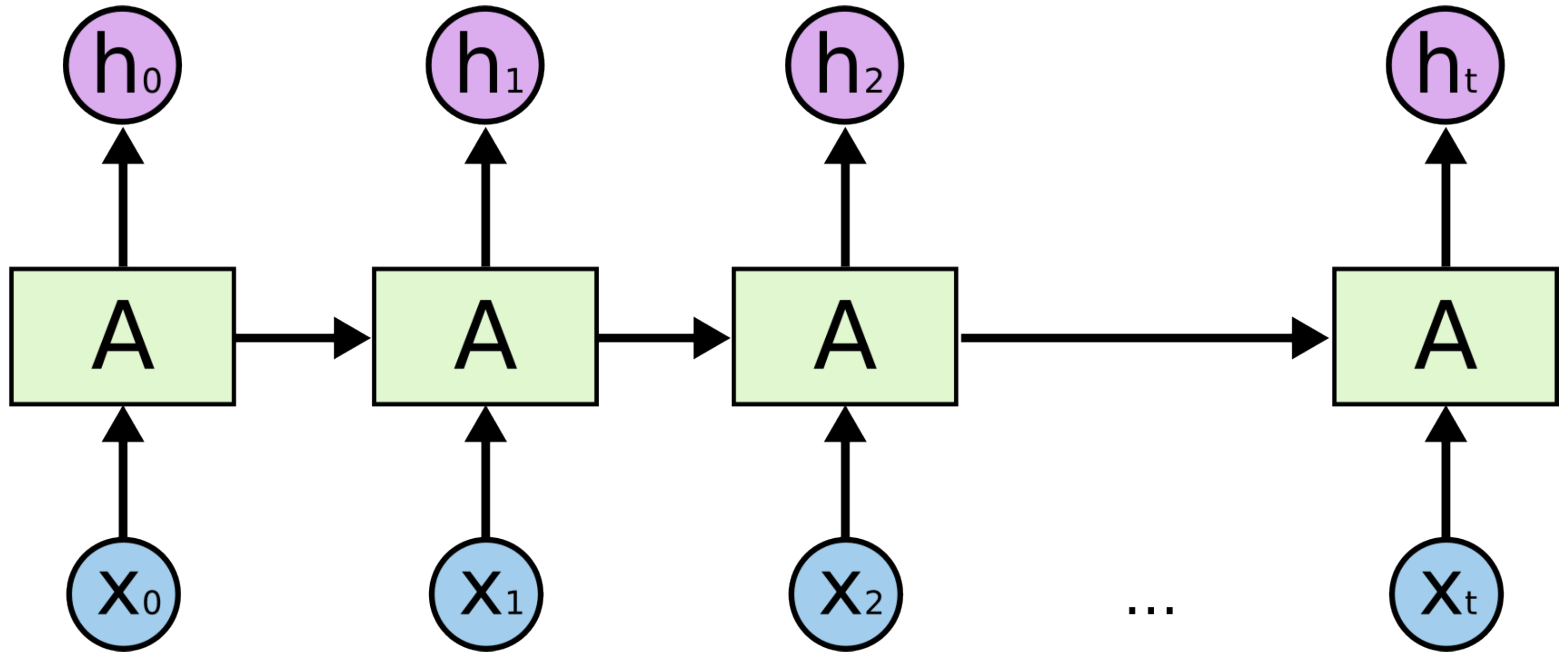
CONS

- Computation can be very slow

RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length
- Can model long-range dependencies
- Fixed model size regardless of input size



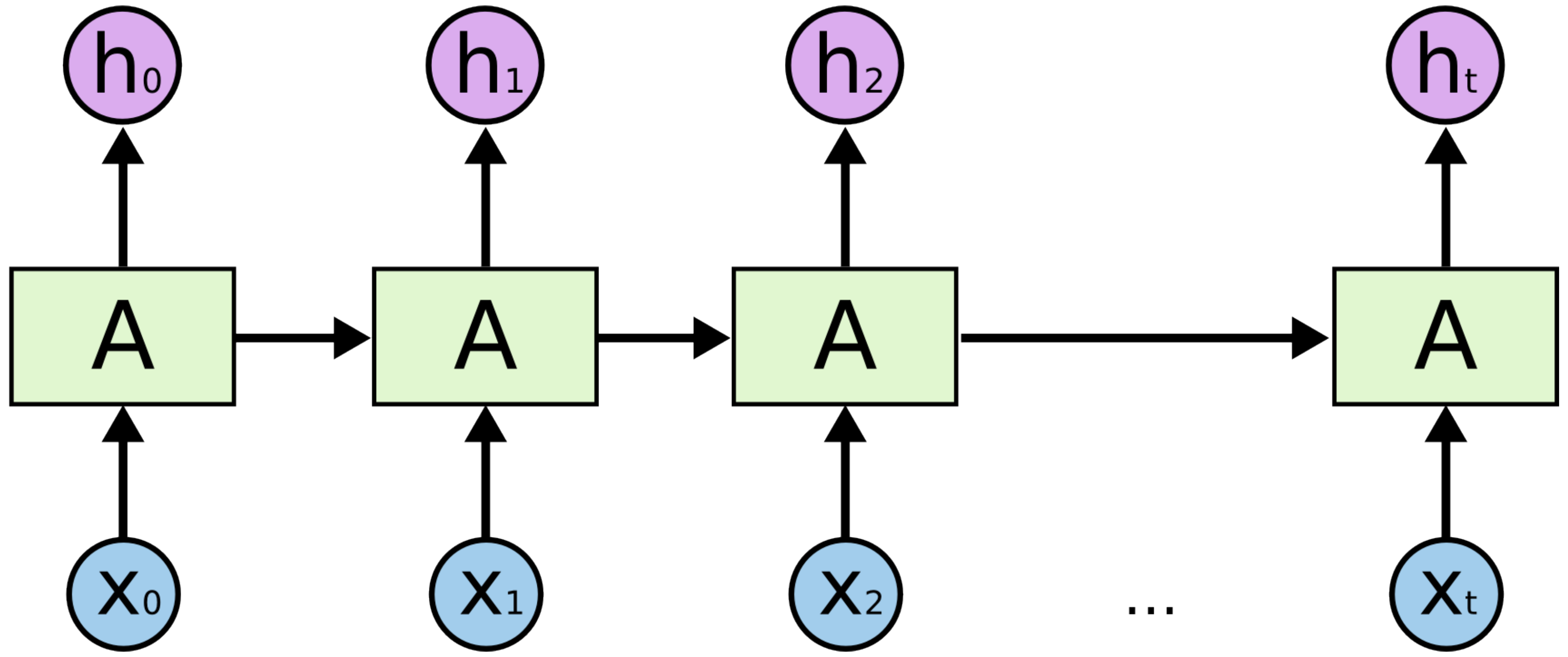
CONS

- Computation can be very slow
- Information degrades in every time step

RECURRENT NEURAL NETS: PROS AND CONS

PROS

- Can take inputs of variable (and potentially infinite) length
- Can model long-range dependencies
- Fixed model size regardless of input size



CONS

- Computation can be very slow
- Information degrades in every time step
- Exploding and vanishing gradients

RECURRENT NEURAL NETS: VANISHING GRADIENT PROBLEM

Analysis for simplified case ($\sigma =$ identity). General case follows similar proof.

(Whiteboard)

LSTM: LONG SHORT-TERM MEMORY NETWORK

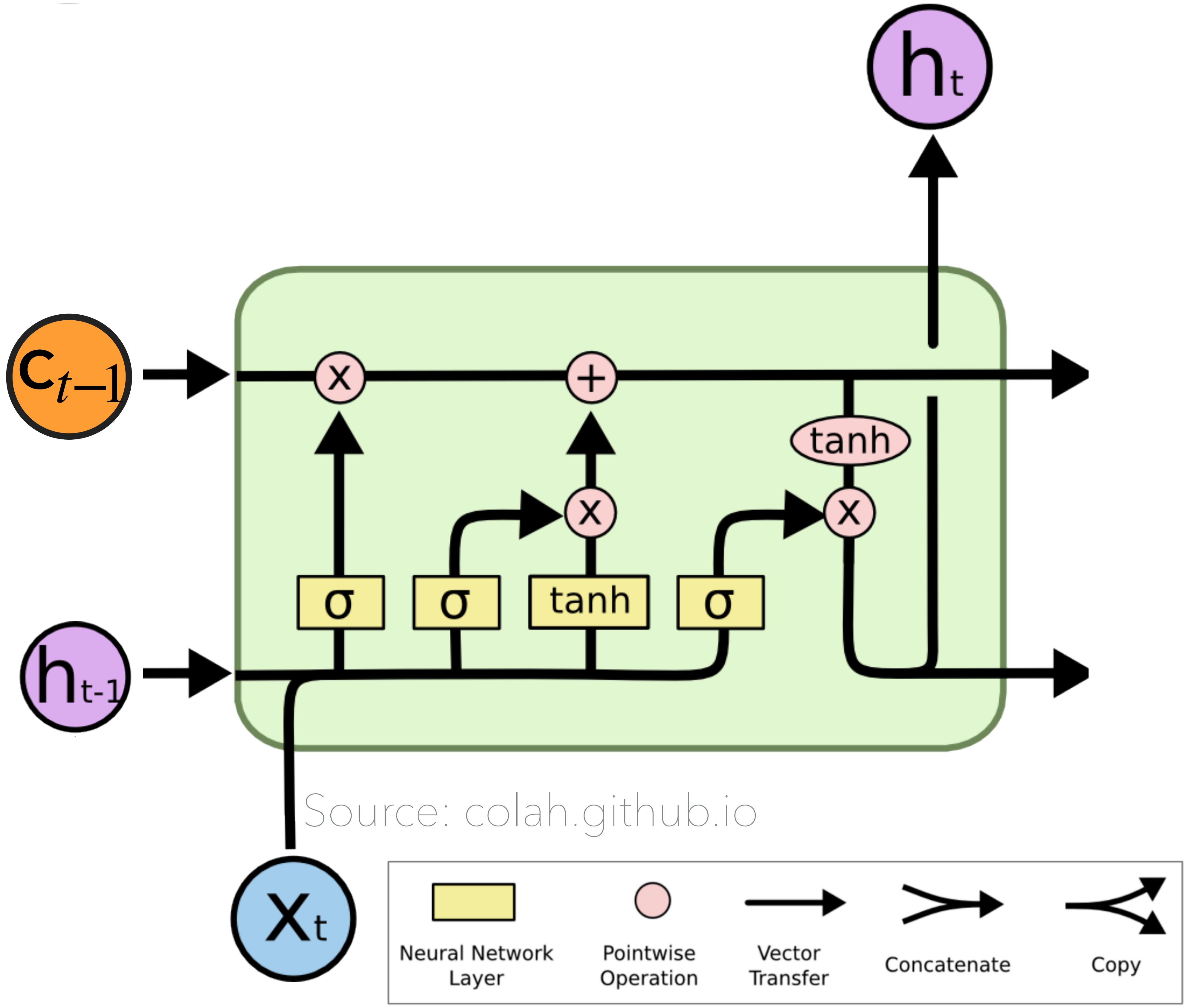
[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow

LSTM: LONG SHORT-TERM MEMORY NETWORK

[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow

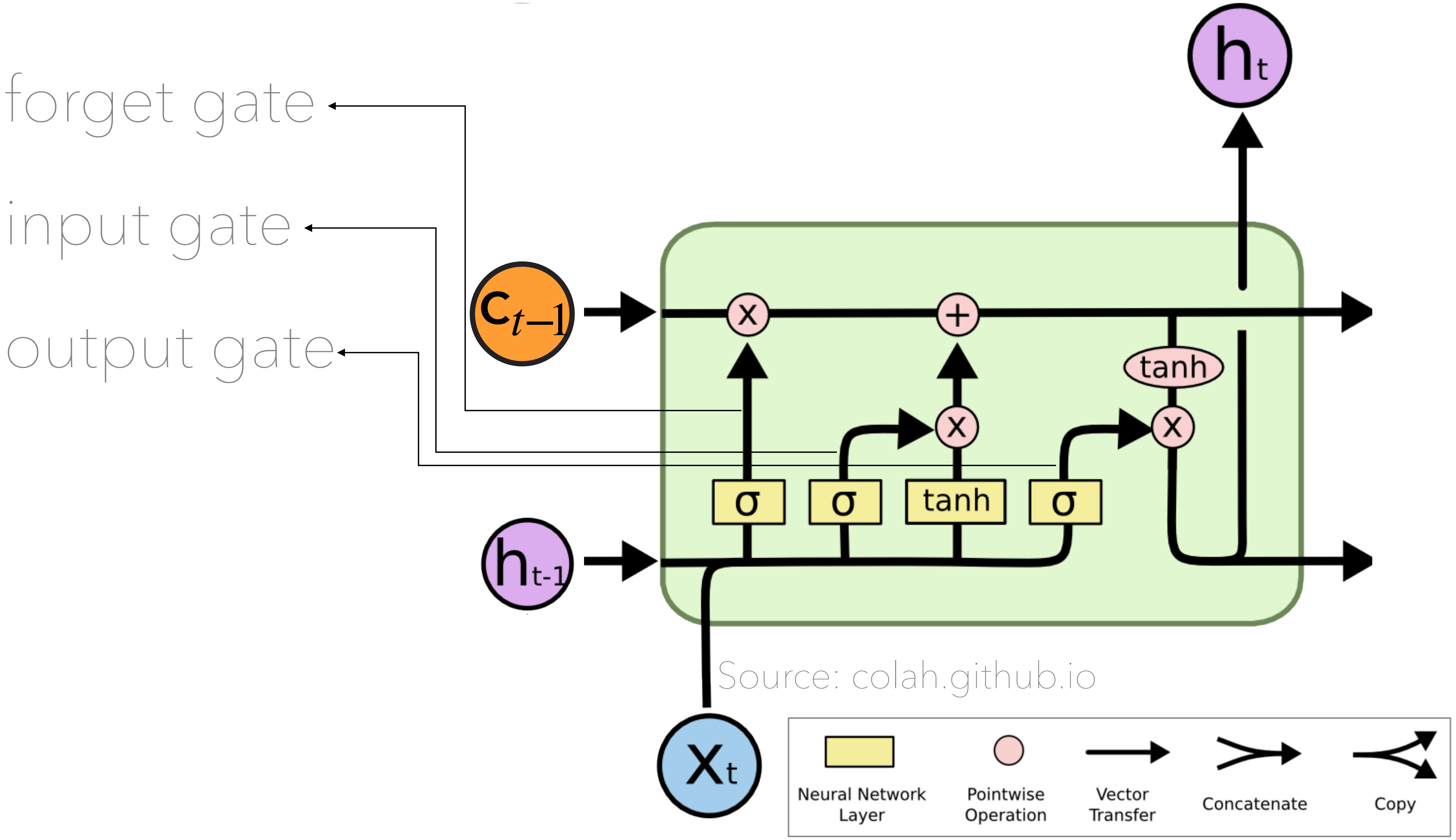


Source: colah.github.io

LSTM: LONG SHORT-TERM MEMORY NETWORK

[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow



Source: colah.github.io

LSTM: LONG SHORT-TERM MEMORY NETWORK

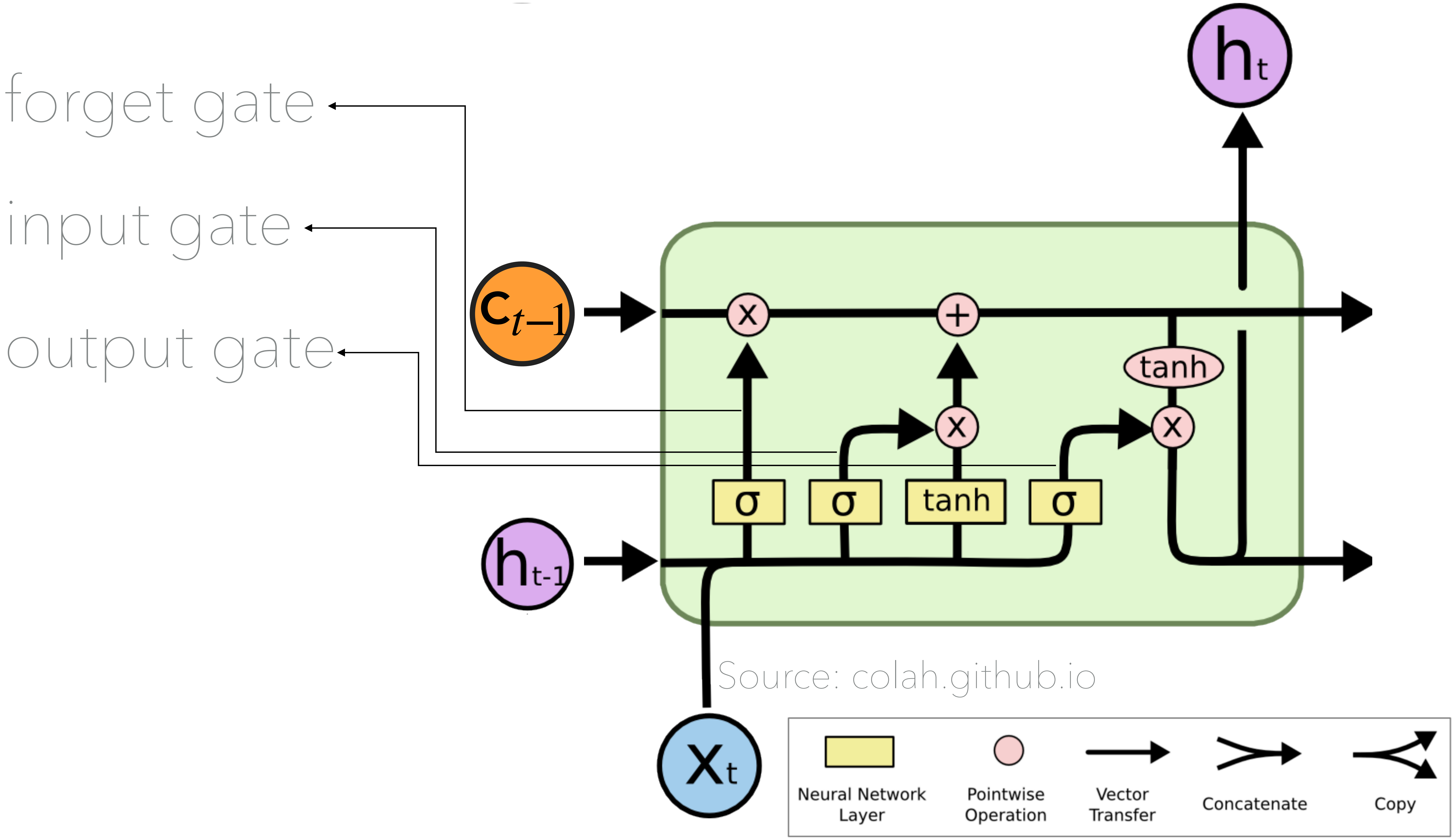
[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$



Source: colah.github.io

LSTM: LONG SHORT-TERM MEMORY NETWORK

[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$

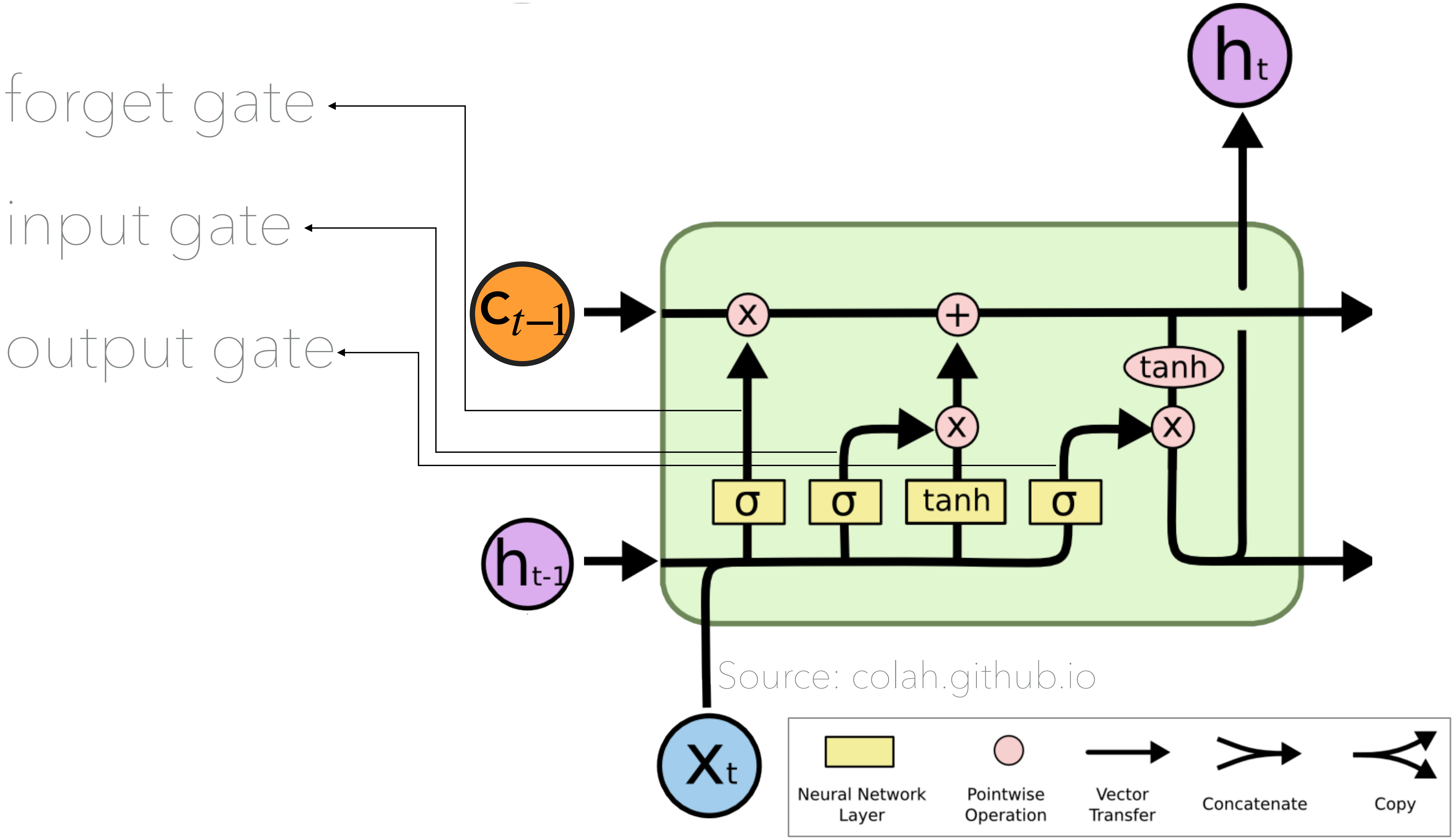
$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$



Source: colah.github.io

LSTM: LONG SHORT-TERM MEMORY NETWORK

[Schmidhuber et al. 1992]

Addresses the gradient problems by using 'gates' to control information flow

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$ forget gate

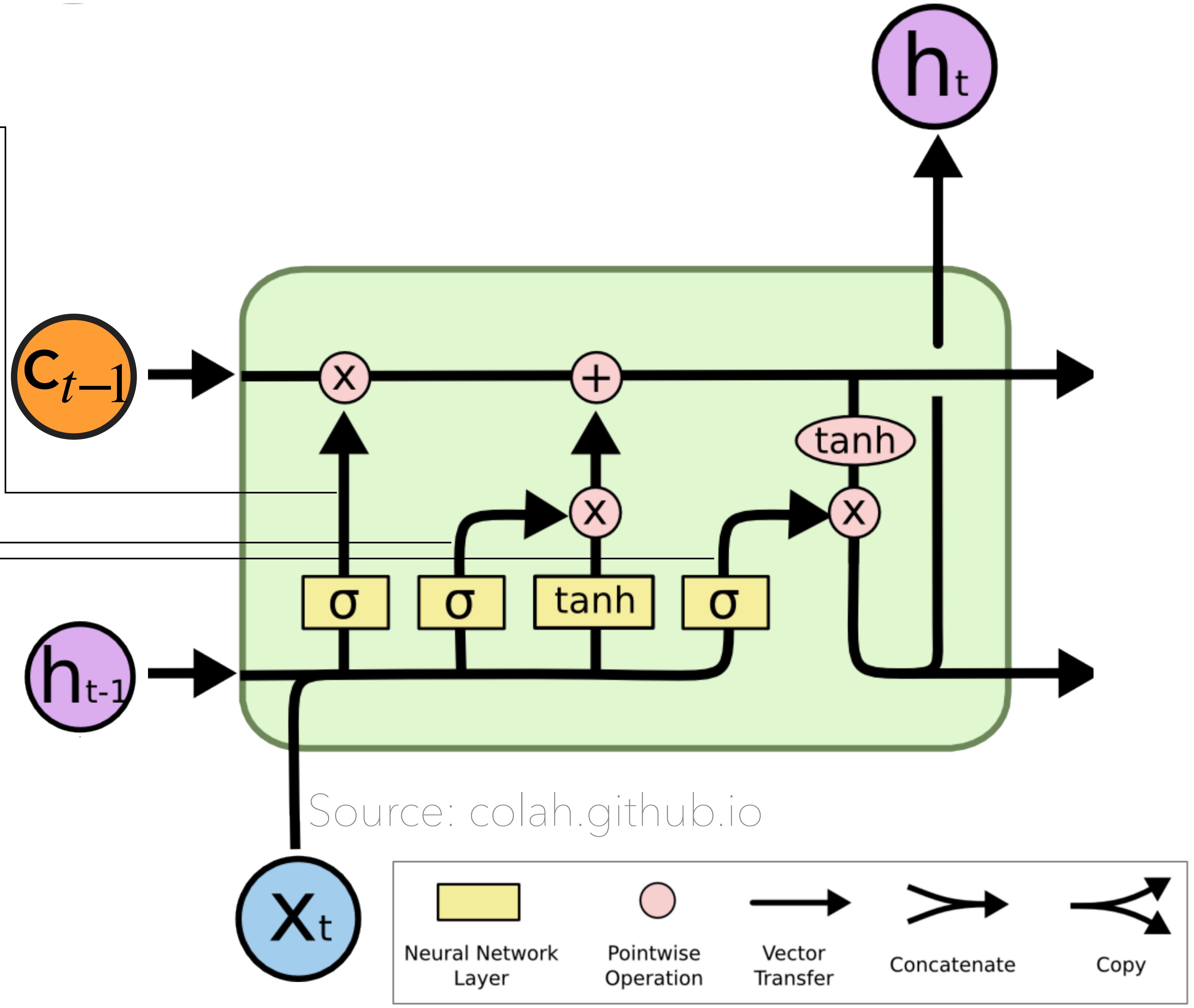
$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$ input gate

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$ output gate

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$ cell update

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$ cell state

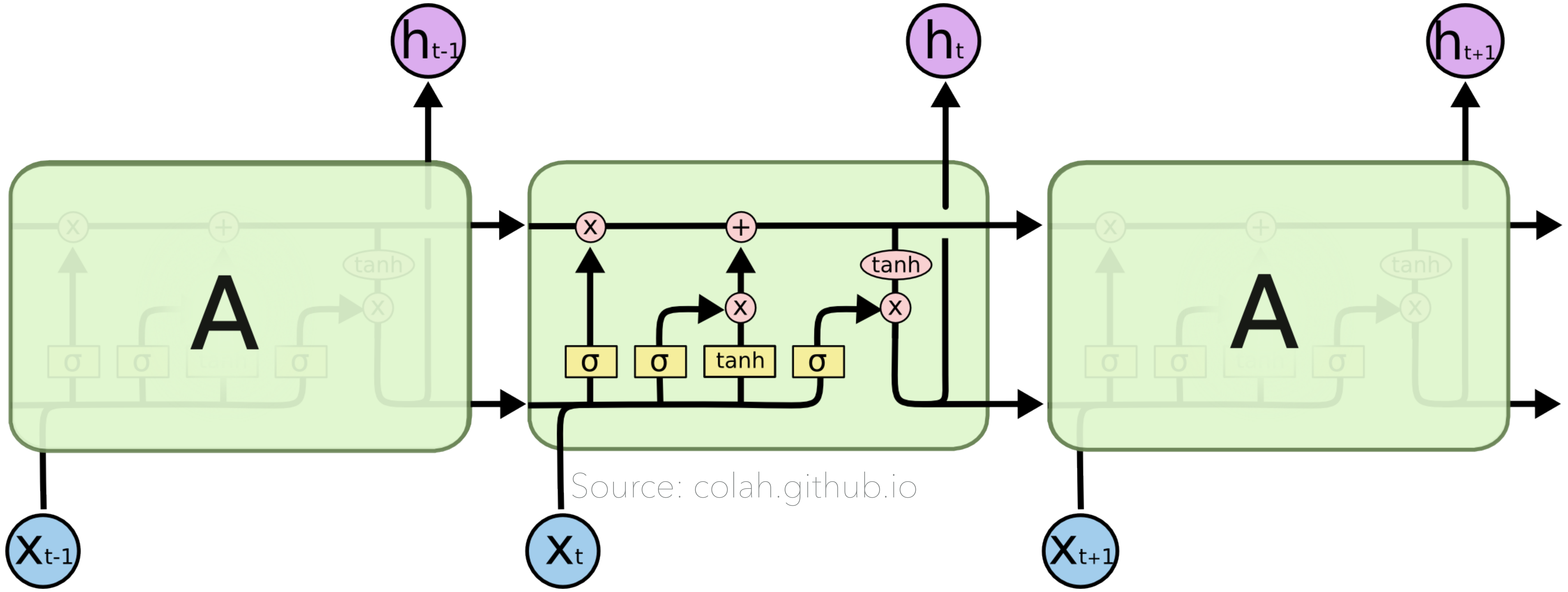
$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$ hidden state



Source: colah.github.io

LSTM: LONG SHORT-TERM MEMORY NETWORK

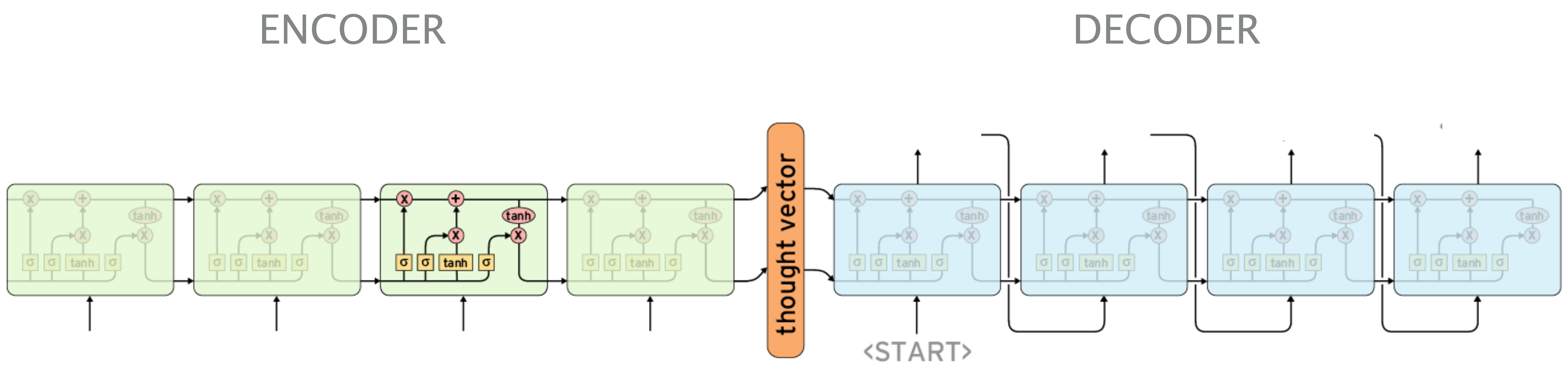
[Schmidhuber et al. 1992]



SEQ2SEQ: SEQUENCE TO SEQUENCE MODEL

[Sutskever et al. 2014]

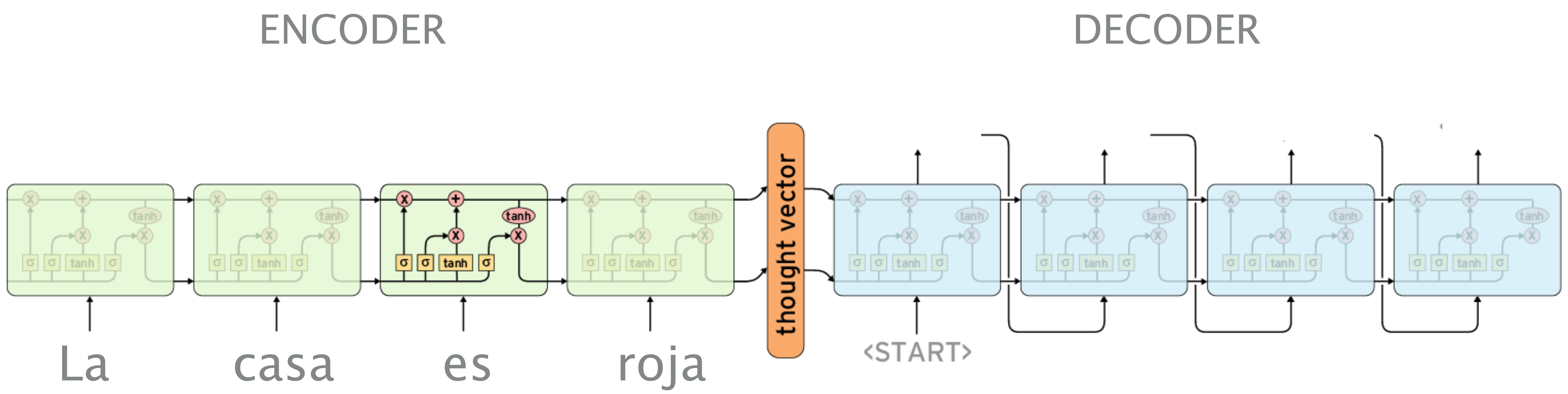
2 RNN's: encoder (processes input sentence) and decoder (generates output)



SEQ2SEQ: SEQUENCE TO SEQUENCE MODEL

[Sutskever et al. 2014]

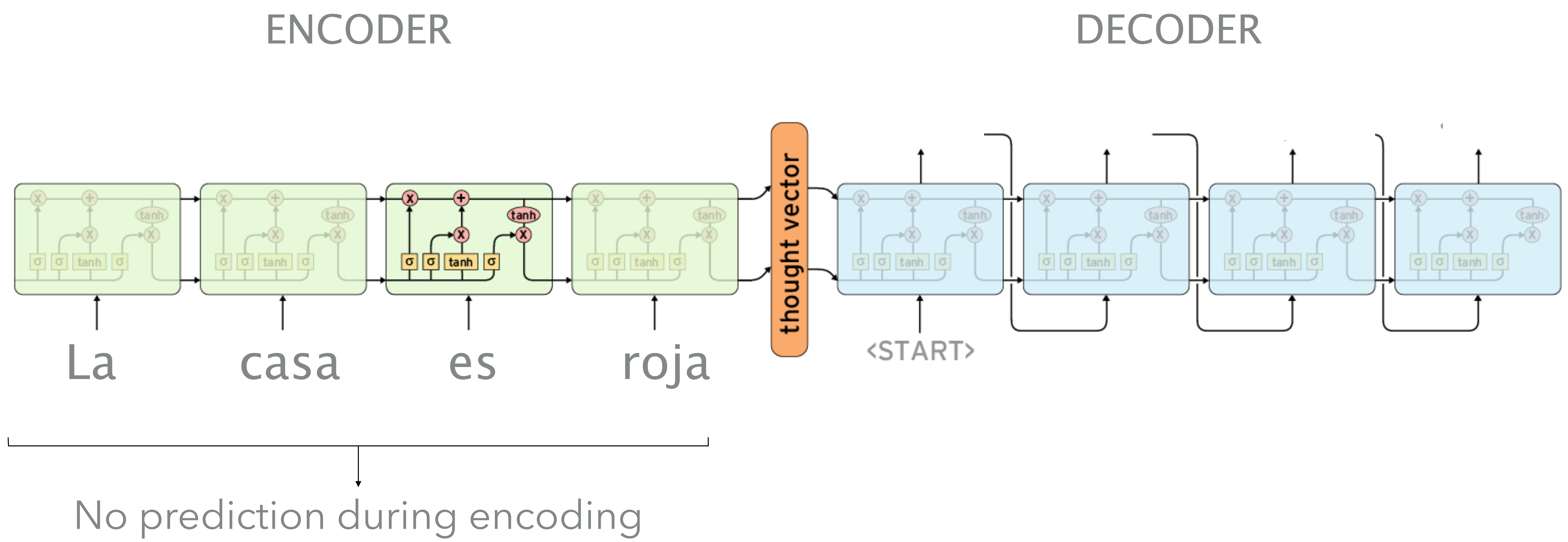
2 RNN’s: encoder (processes input sentence) and decoder (generates output)



SEQ2SEQ: SEQUENCE TO SEQUENCE MODEL

[Sutskever et al. 2014]

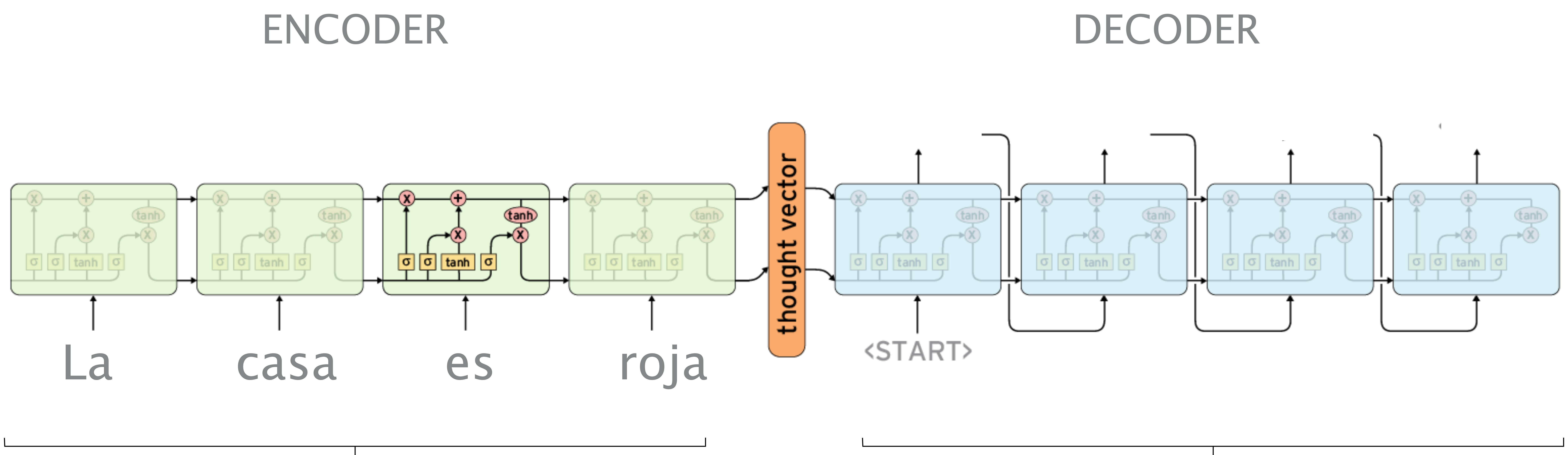
2 RNN's: encoder (processes input sentence) and decoder (generates output)



SEQ2SEQ: SEQUENCE TO SEQUENCE MODEL

[Sutskever et al. 2014]

2 RNN's: encoder (processes input sentence) and decoder (generates output)



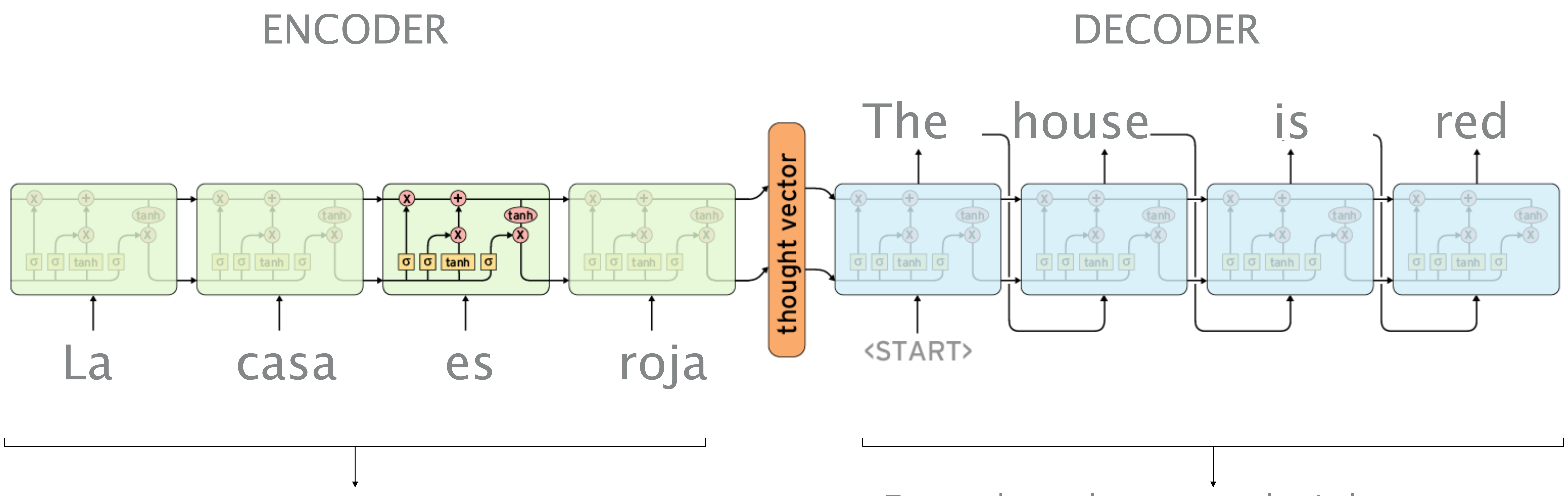
No prediction during encoding

Decoder takes encoder's last state + a special <start> token as inputs

SEQ2SEQ: SEQUENCE TO SEQUENCE MODEL

[Sutskever et al. 2014]

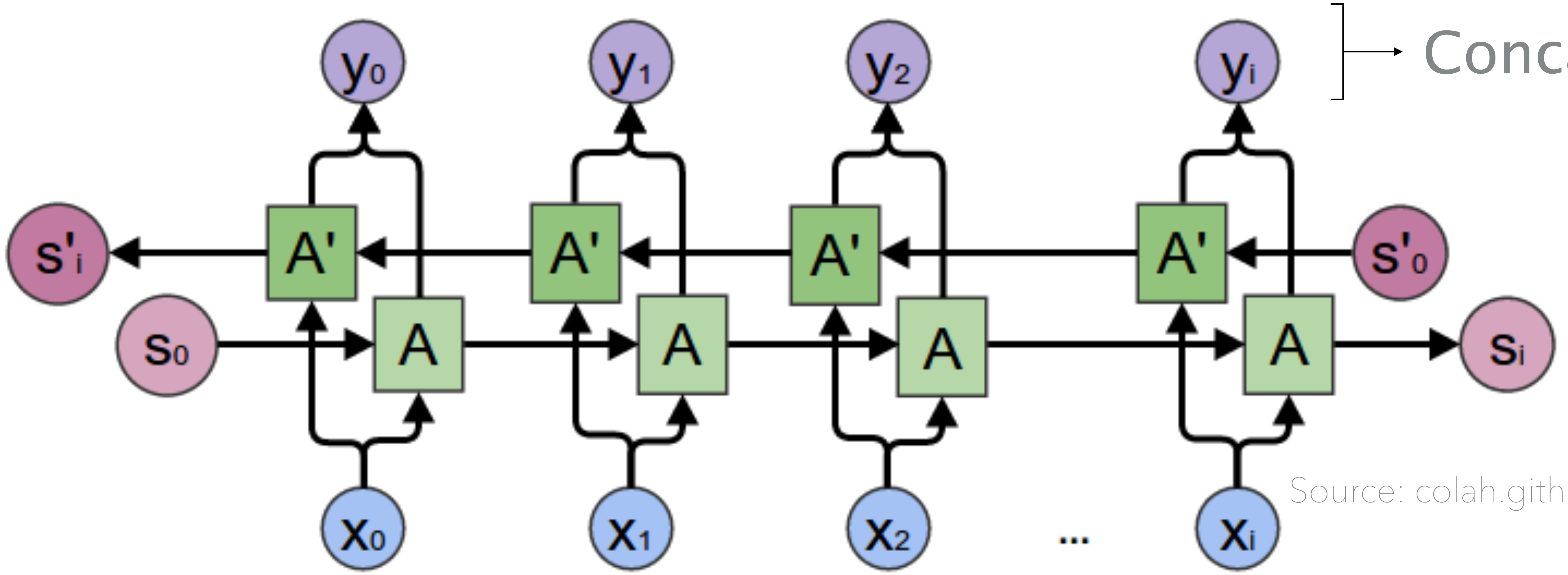
2 RNN's: encoder (processes input sentence) and decoder (generates output)



No prediction during encoding

Decoder takes encoder's last state + a special <start> token as inputs

BIDIRECTIONAL RNNS



Concatenate hidden states

Backward RNN

Forward RNN

Source: colah.github.io

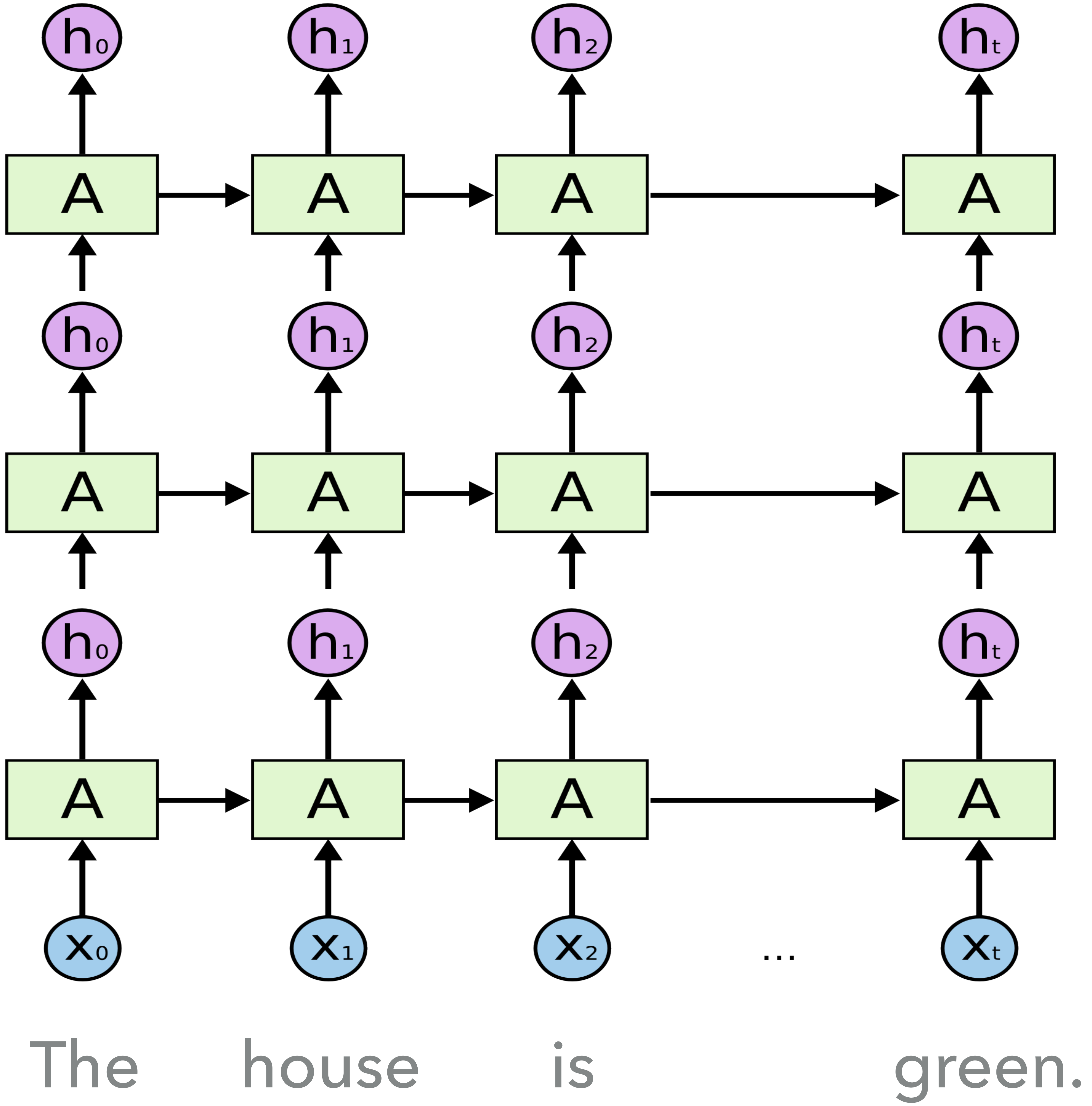
The house is red

Advantage: prediction can rely on both left and right context

Note: not applicable to Language Modeling! (Why?)

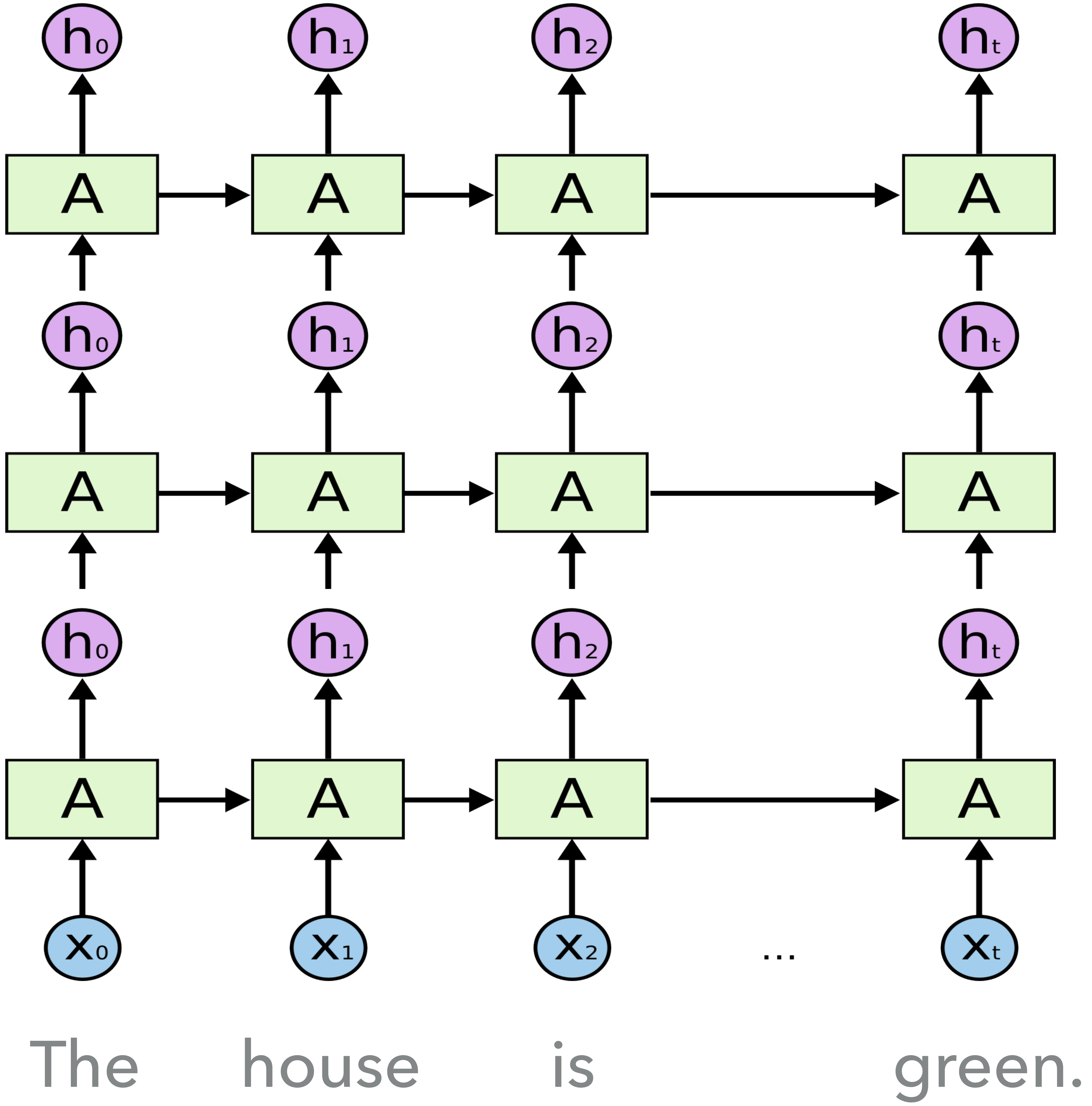
DEEP / STACKED / MULTI-LAYER RNNS

- Inputs to i-th RNN are hidden states of (i-1)-th RNN



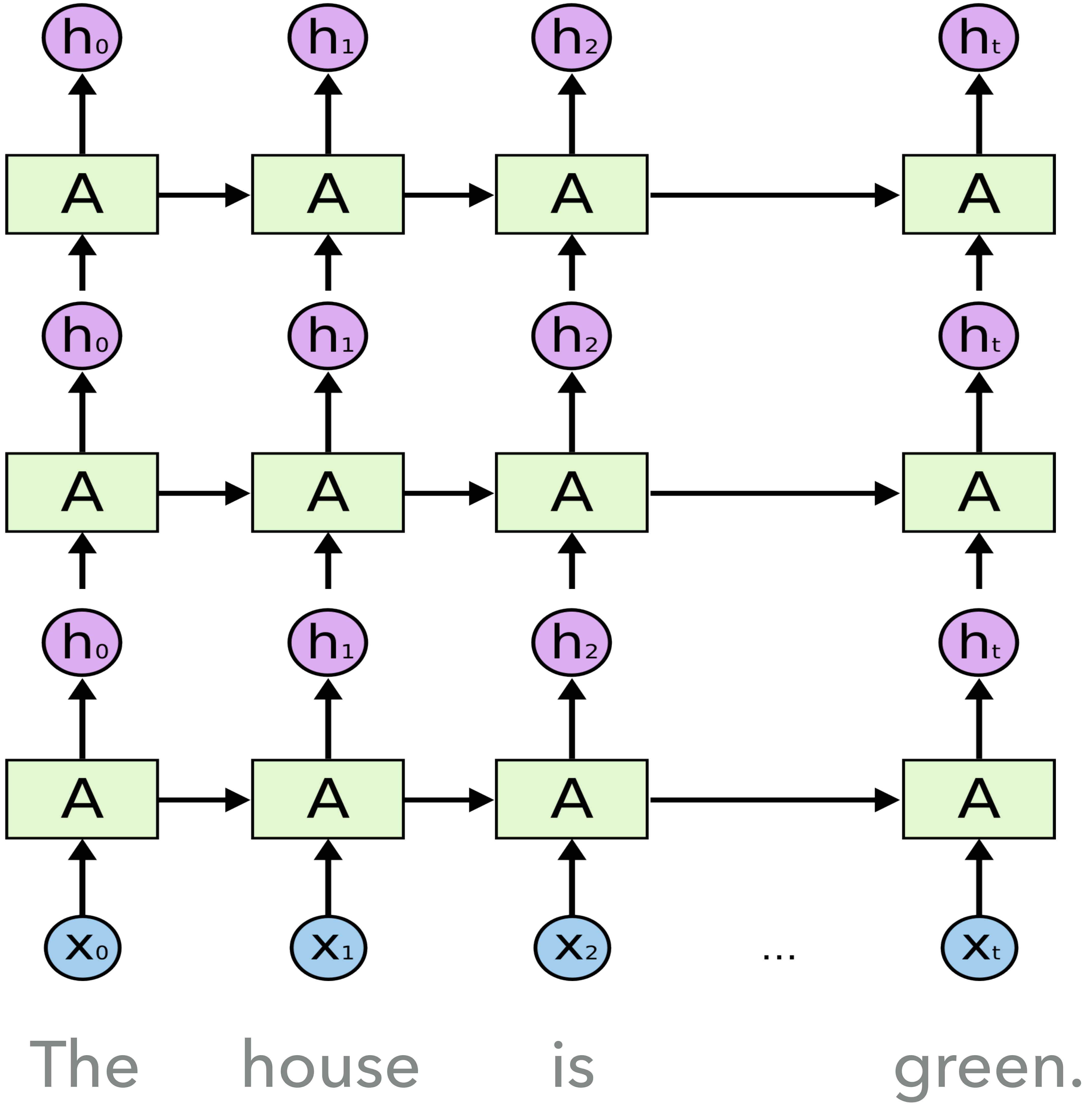
DEEP / STACKED / MULTI-LAYER RNNS

- Inputs to i-th RNN are hidden states of (i-1)-th RNN
- Allows RNN to learn more complex representations



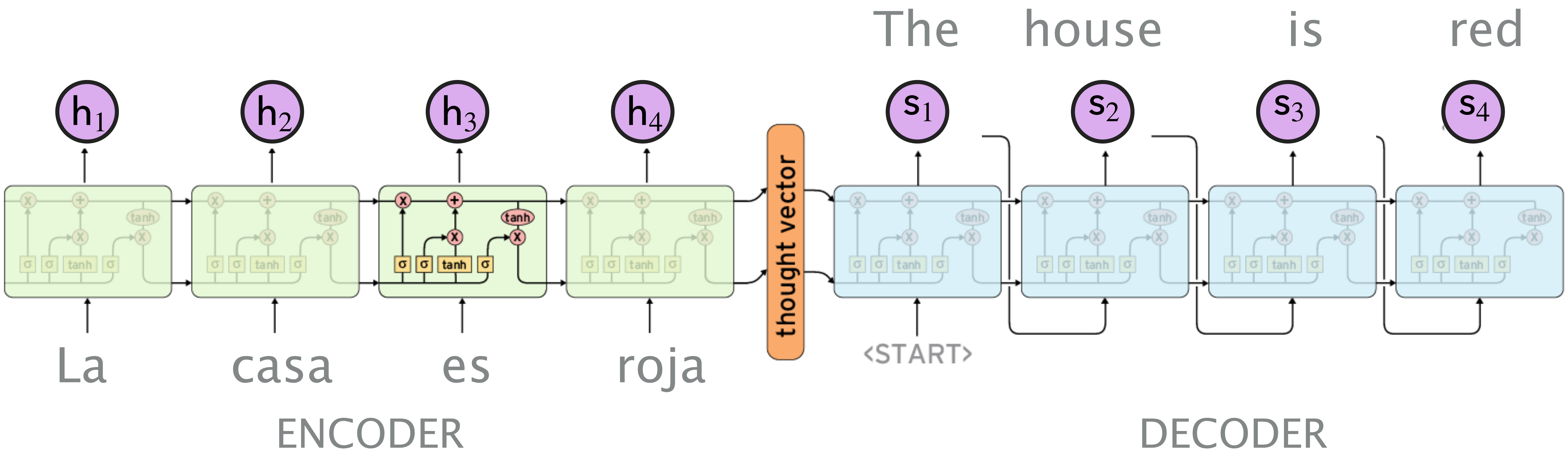
DEEP / STACKED / MULTI-LAYER RNNS

- Inputs to i-th RNN are hidden states of (i-1)-th RNN
- Allows RNN to learn more complex representations
- Typically: lower RNNs learn local/simpler features, higher RNNs learning global/abstract features



ATTENTION

Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

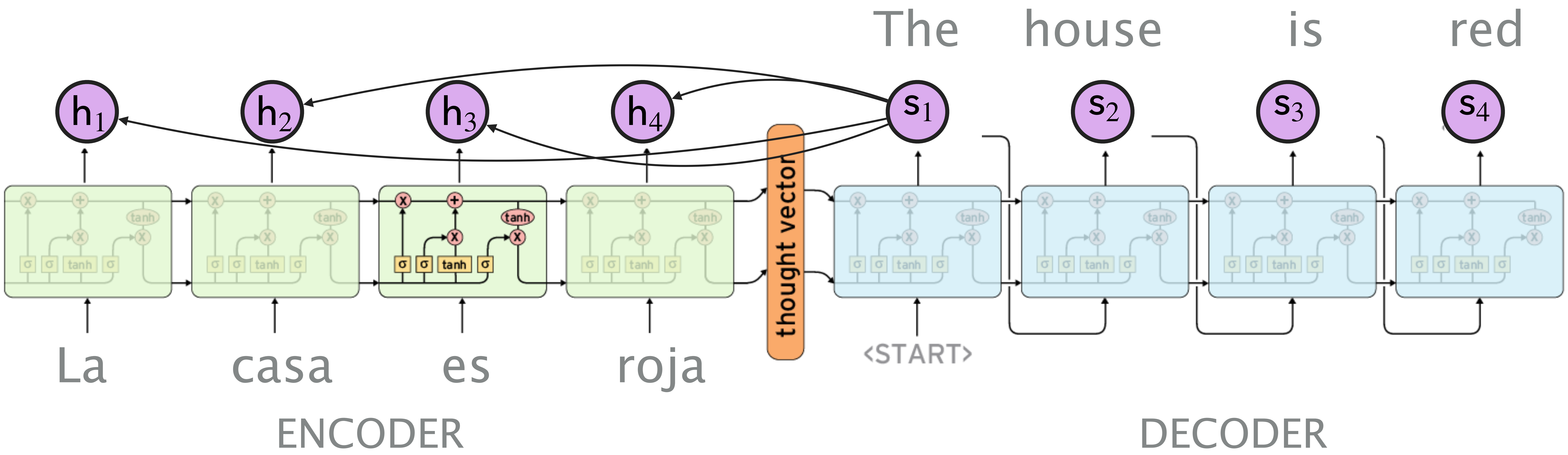


ATTENTION

Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

$$\mathbf{e}_t = [\mathbf{s}_t \cdot \mathbf{h}_1, \ldots, \mathbf{s}_t \cdot \mathbf{h}_N]$$

$\mathbf{s}_t$ is sometimes called the 'query'



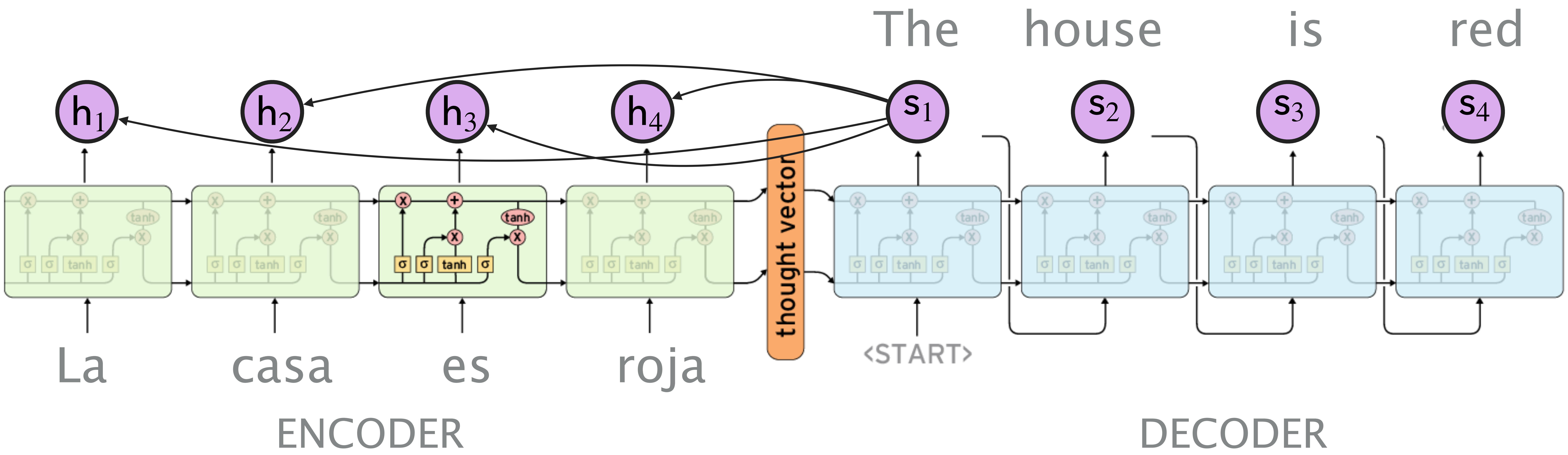
ATTENTION

Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

$$\mathbf{e}_t = [\mathbf{s}_t \cdot \mathbf{h}_1, \ldots, \mathbf{s}_t \cdot \mathbf{h}_N]$$

$\mathbf{s}_t$ is sometimes called the 'query'

$$\alpha_t = \text{softmax}(\mathbf{e}_t)$$



ATTENTION

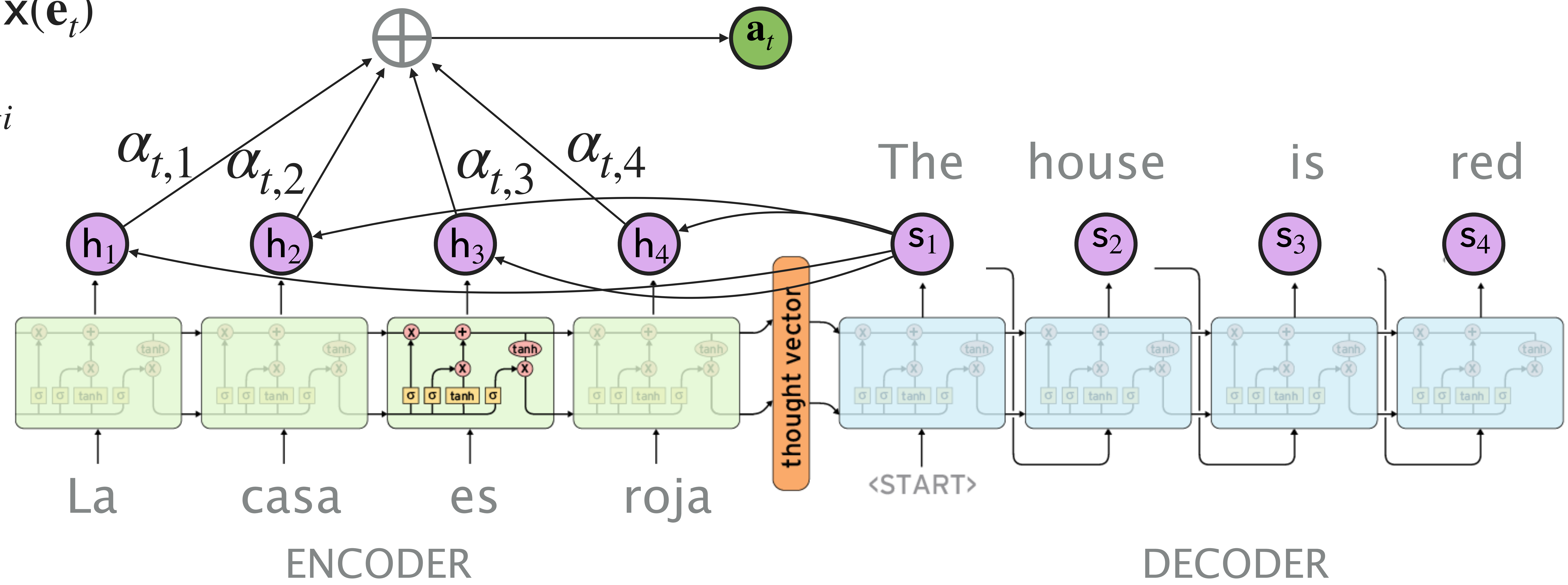
Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

$$\mathbf{e}_t = [\mathbf{s}_t \cdot \mathbf{h}_1, \ldots, \mathbf{s}_t \cdot \mathbf{h}_N]$$

$\mathbf{s}_t$ is sometimes called the 'query'

$$\alpha_t = \text{softmax}(\mathbf{e}_t)$$

$$\mathbf{a}_t = \sum_{i=1}^{N} \alpha_{t,i} \mathbf{h}_i$$



ATTENTION

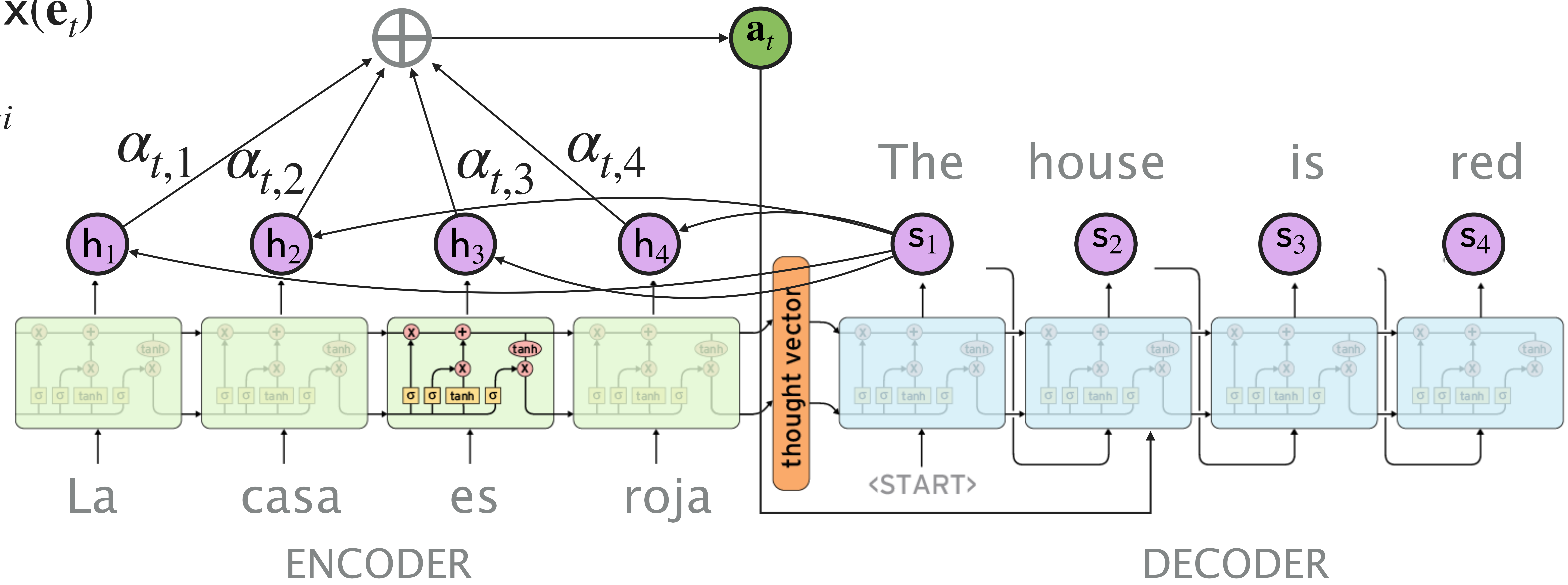
Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

$$\mathbf{e}_t = [\mathbf{s}_t \cdot \mathbf{h}_1, \ldots, \mathbf{s}_t \cdot \mathbf{h}_N]$$

$\mathbf{s}_t$ is sometimes called the 'query'

$$\alpha_t = \text{softmax}(\mathbf{e}_t)$$

$$\mathbf{a}_t = \sum_{i=1}^{N} \alpha_{t,i} \mathbf{h}_i$$



ATTENTION

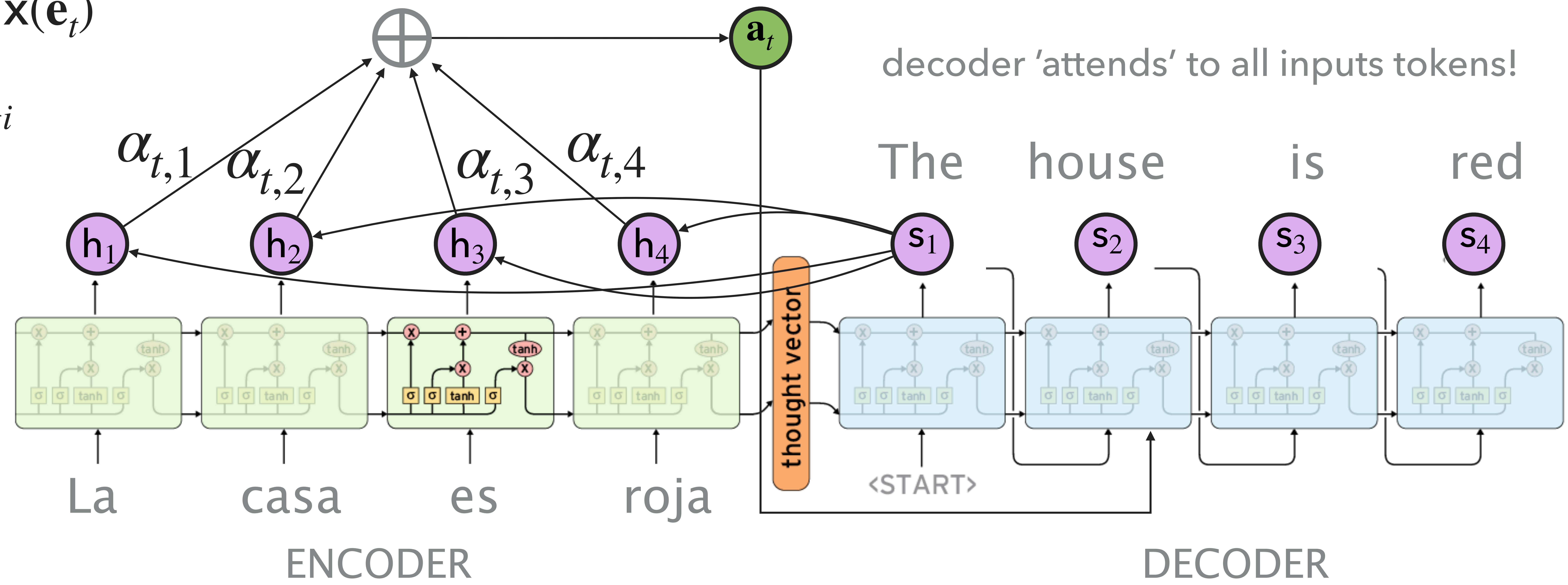
Motivation: entire meaning of source sentence encoded in one vector! (the '**bottleneck** problem')

$$\mathbf{e}_t = [\mathbf{s}_t \cdot \mathbf{h}_1, \ldots, \mathbf{s}_t \cdot \mathbf{h}_N]$$

$\mathbf{s}_t$ is sometimes called the 'query'

$$\alpha_t = \text{softmax}(\mathbf{e}_t)$$

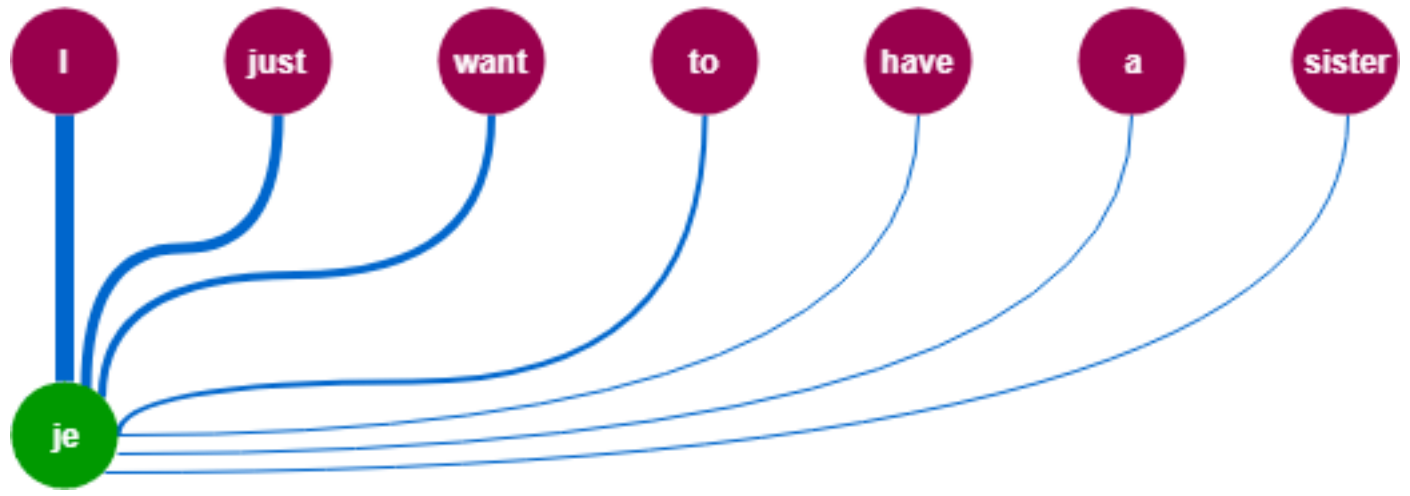
$$\mathbf{a}_t = \sum_{i=1}^{N} \alpha_{t,i} \mathbf{h}_i$$



ATTENTION

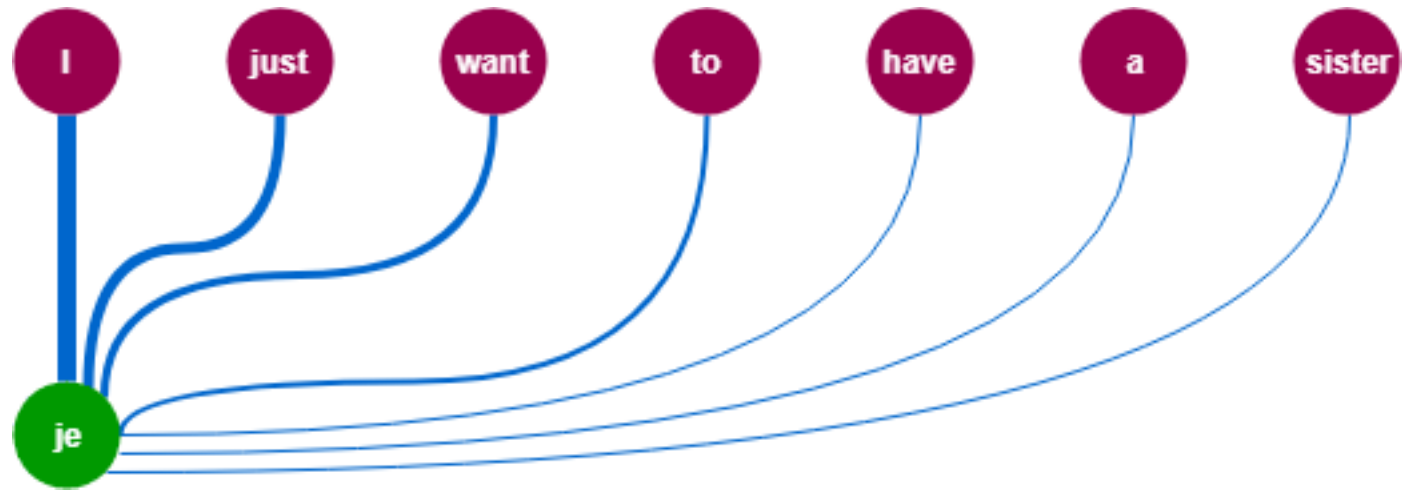
…for machine translation:

Source: trungtran.io



ATTENTION

…for machine translation:

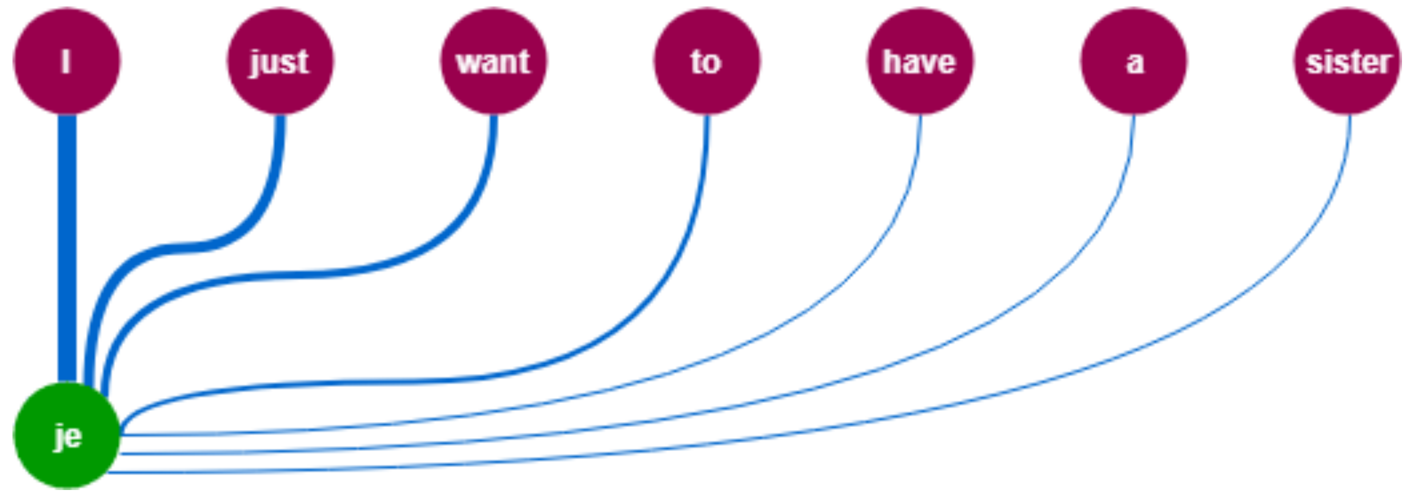


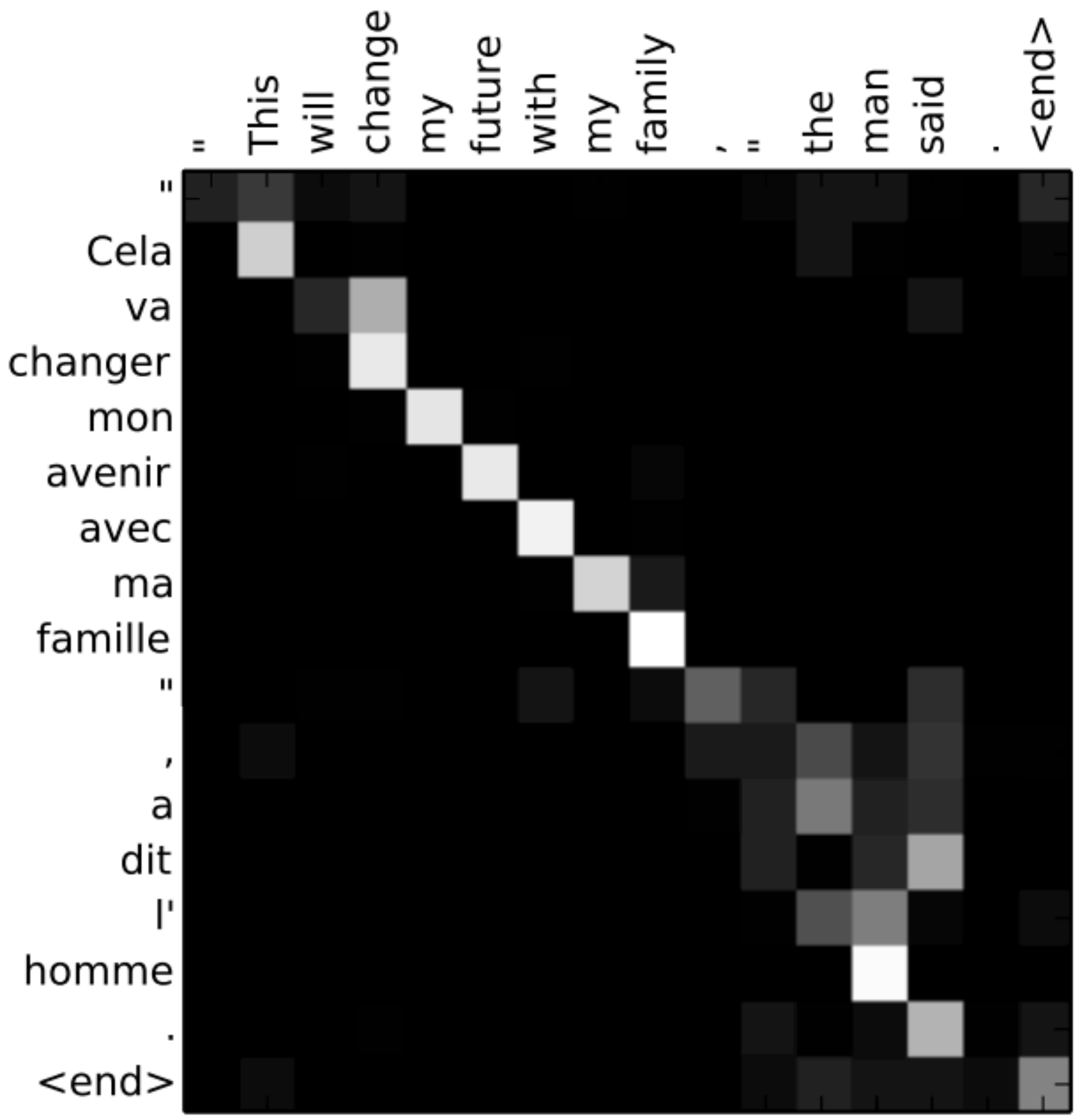
Source: trungtran.io

ATTENTION

…for machine translation:

Source: trungtran.io

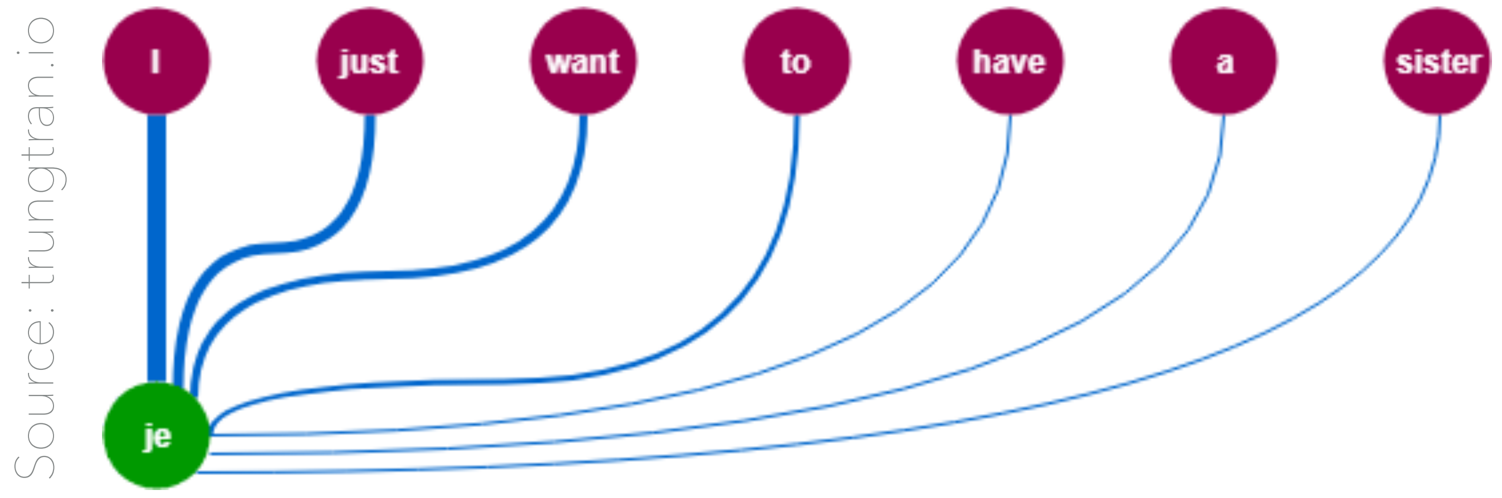


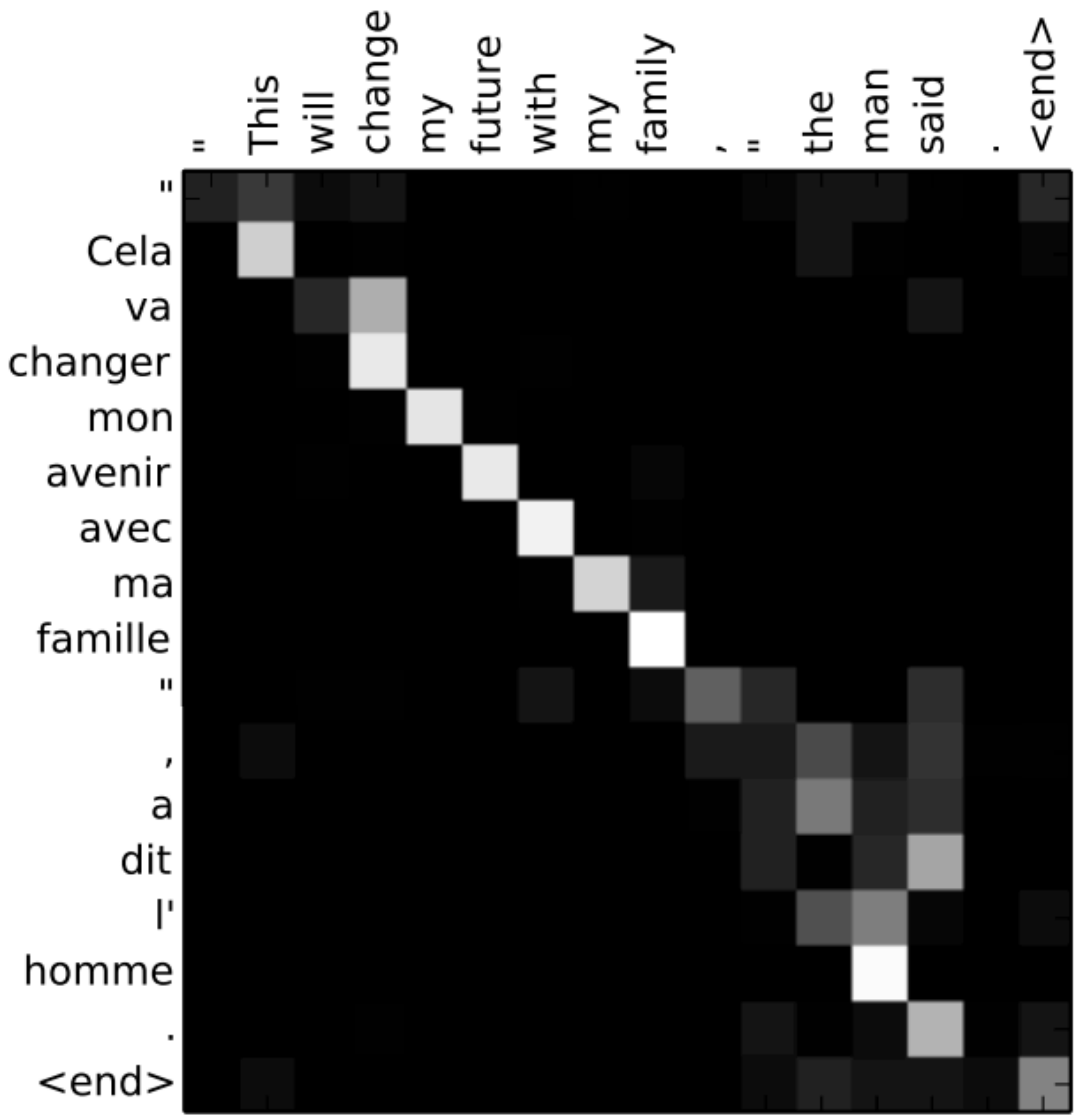


[Bahdanau et al., 2015]

ATTENTION

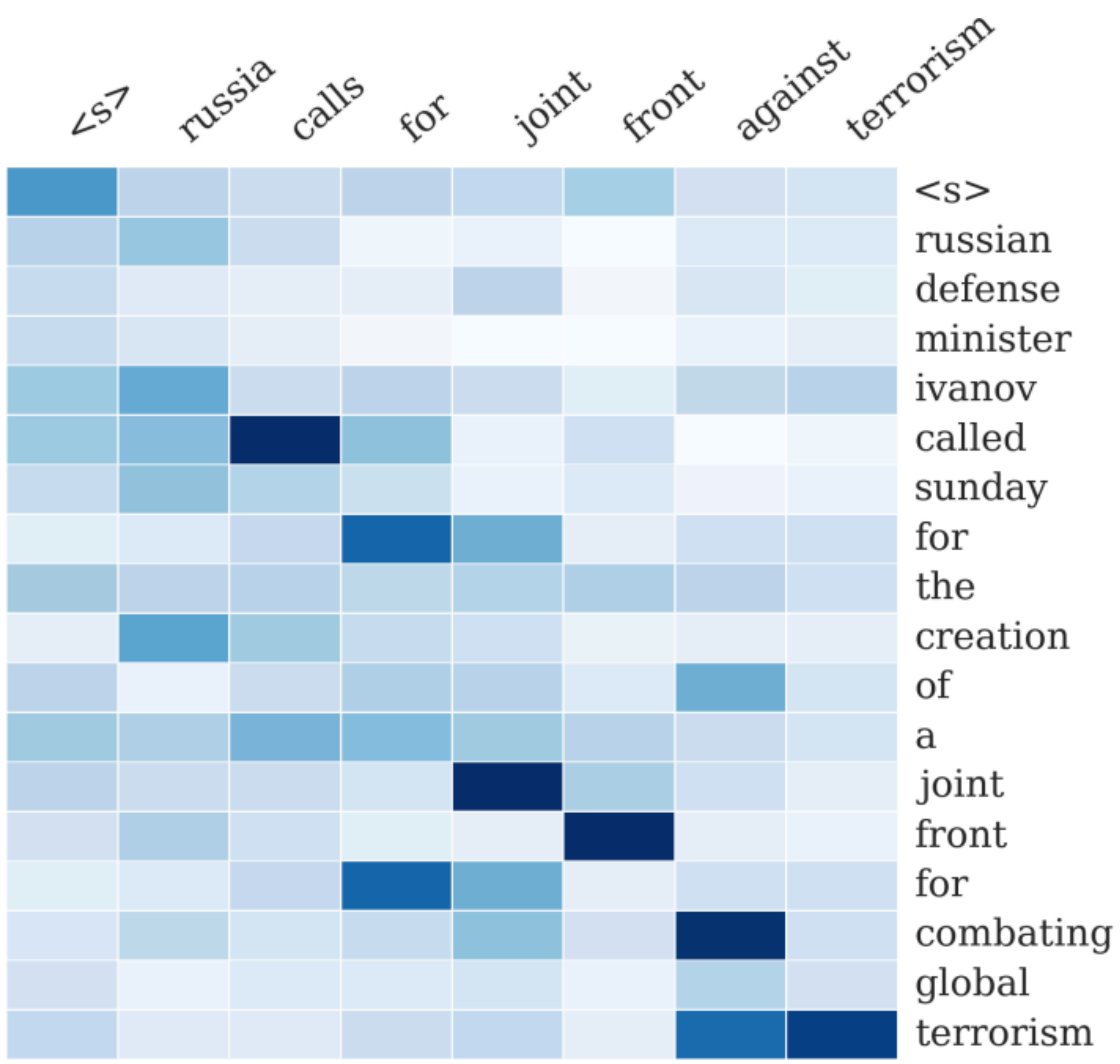
…for machine translation:





[Bahdanau et al., 2015]

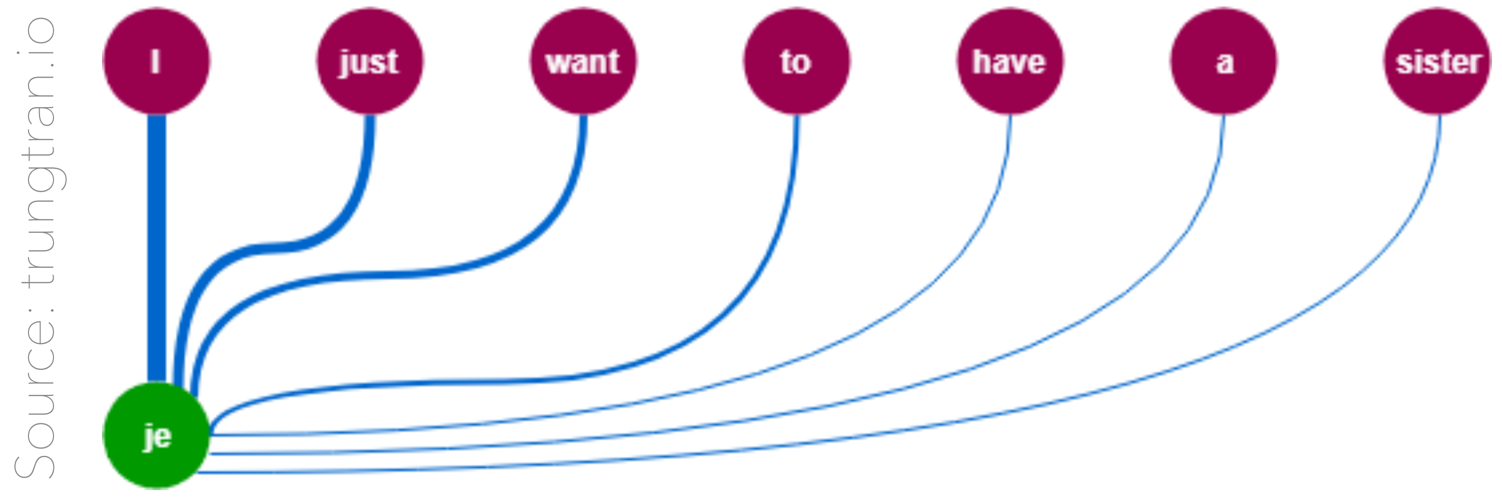
… for summarization:

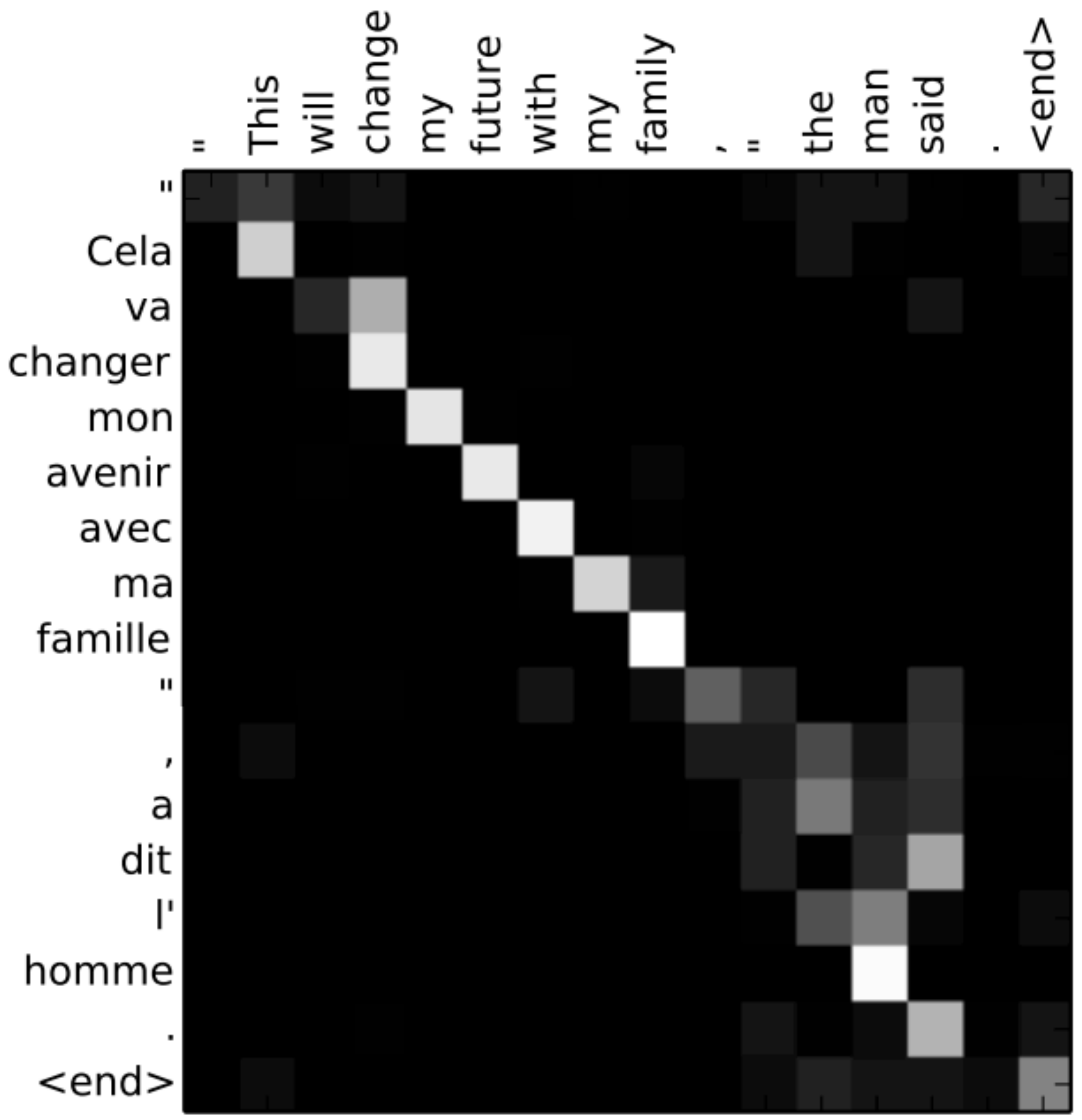


[Rush et al., 2015]

ATTENTION

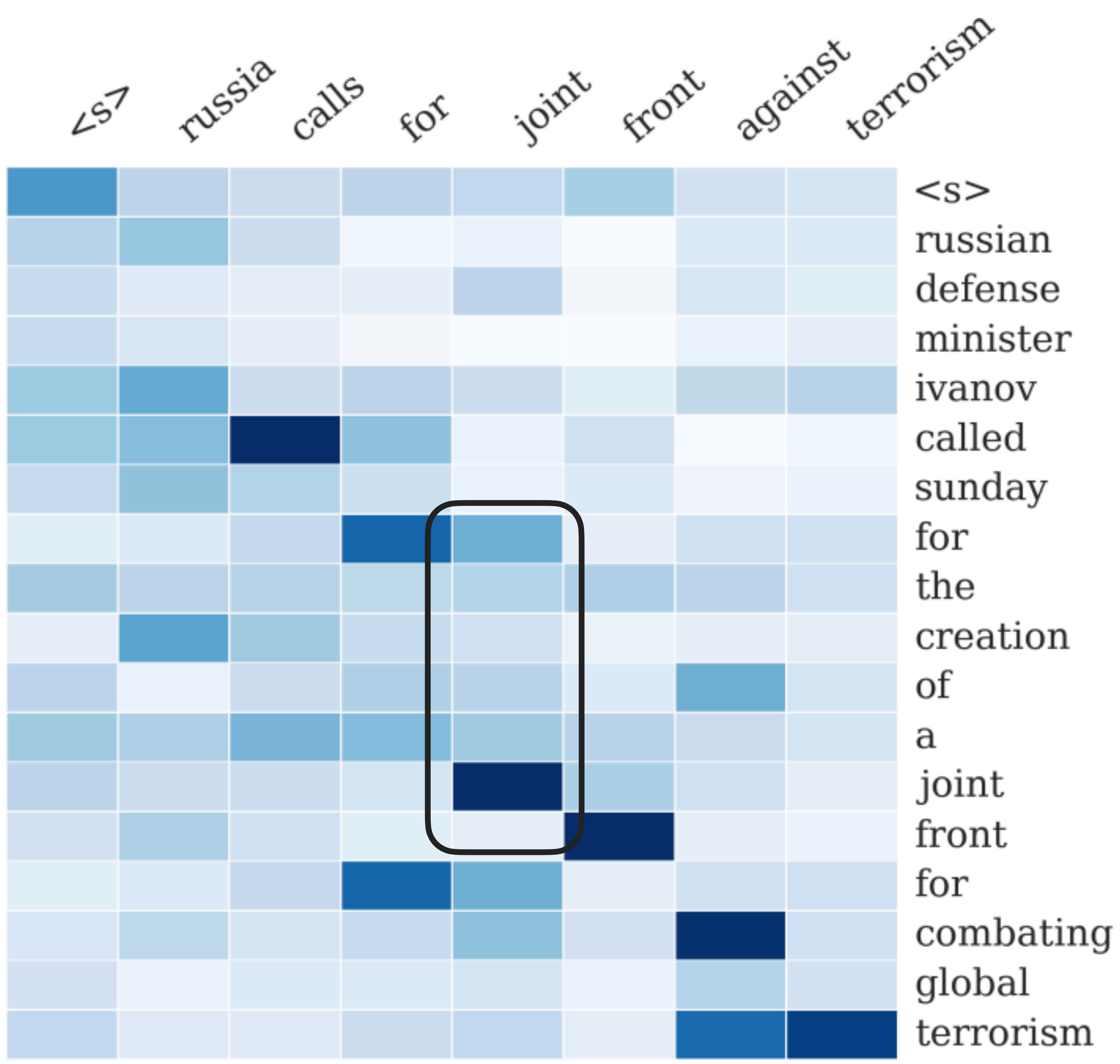
…for machine translation:





[Bahdanau et al., 2015]

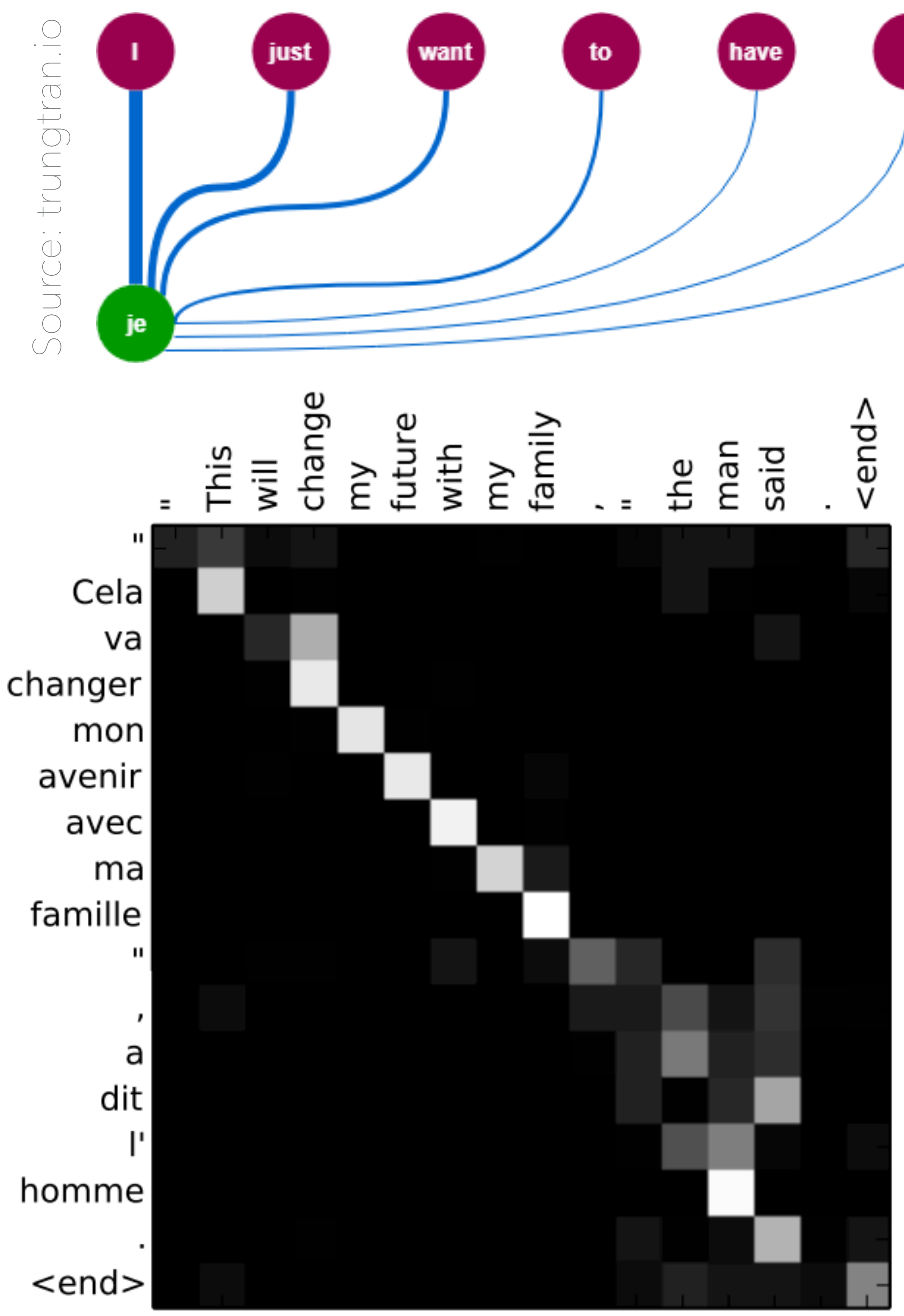
… for summarization:



[Rush et al., 2015]

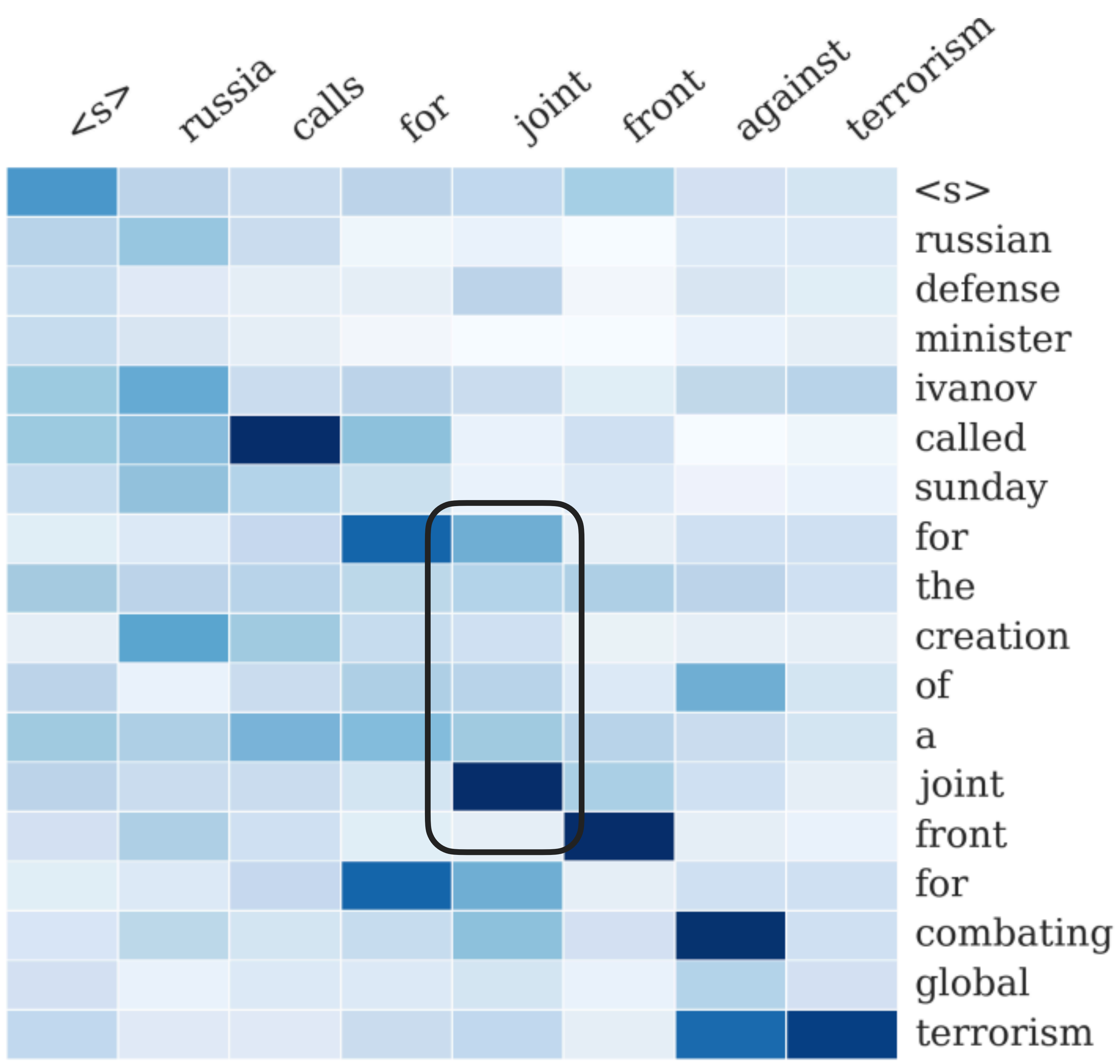
ATTENTION

…for machine translation:



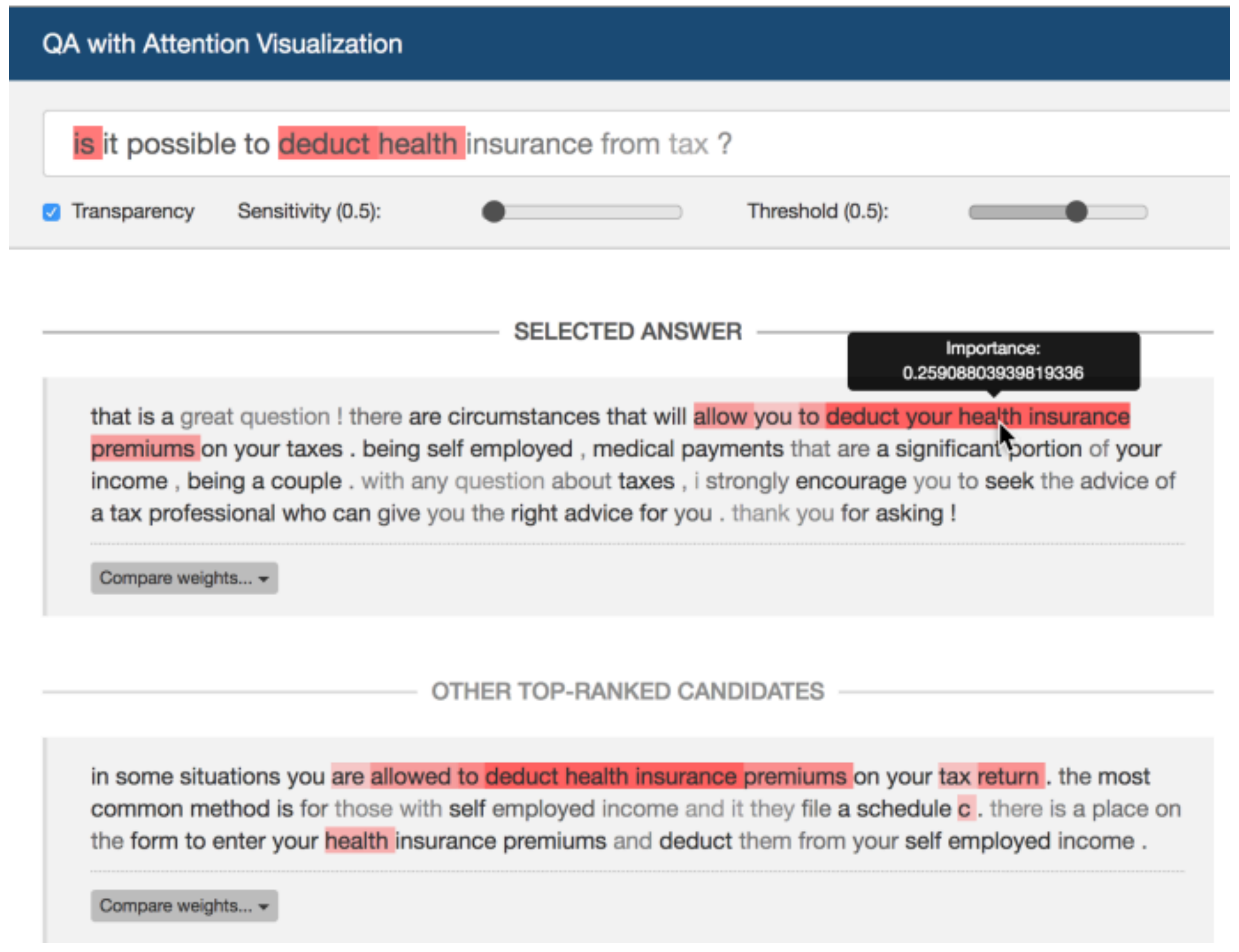
[Bahdanau et al., 2015]

… for summarization:



[Rush et al., 2015]

… for Question Answering:



[Rucke et al., 2017]

PART 3:

LARGE LANGUAGE MODELS:

SELF-ATTENTION, TRANSFORMERS, PRETRAINING

FROM RNN TO ATTENTION-BASED MODELS

FROM RNN TO ATTENTION-BASED MODELS

- RNNs (even LSTMs) struggle with very long-range dependencies

FROM RNN TO ATTENTION-BASED MODELS

- RNNs (even LSTMs) struggle with very long-range dependencies
- So far: linear interaction between individual words. Language isn't linear!

FROM RNN TO ATTENTION-BASED MODELS

- RNNs (even LSTMs) struggle with very long-range dependencies
- So far: linear interaction between individual words. Language isn't linear!
- Starting in 2017, new increasingly larger models have taken over NLP

FROM RNN TO ATTENTION-BASED MODELS

- RNNs (even LSTMs) struggle with very long-range dependencies
- So far: linear interaction between individual words. Language isn't linear!
- Starting in 2017, new increasingly larger models have taken over NLP
- Almost human-quality text generation, state-of-the-art in many tasks

FROM RNN TO ATTENTION-BASED MODELS

- RNNs (even LSTMs) struggle with very long-range dependencies
- So far: linear interaction between individual words. Language isn't linear!
- Starting in 2017, new increasingly larger models have taken over NLP
- Almost human-quality text generation, state-of-the-art in many tasks
- Two key ideas behind them: **attention-based** architectures and **pre-training**

SELF-ATTENTION

Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

- We saw how attention helps in seq2seq models (decoder attends to encoder input)

Source: jalammar.github.io/illustrated-transformer/

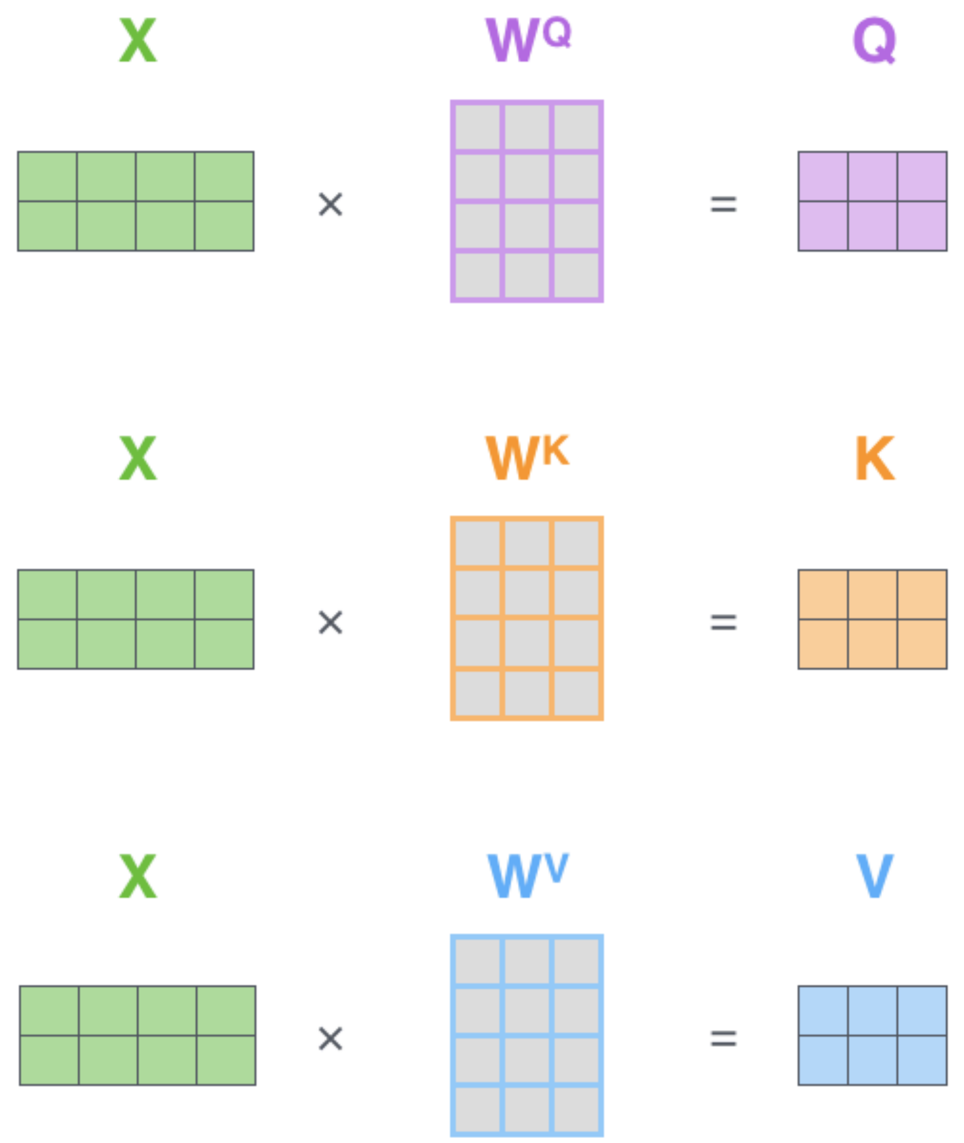
SELF-ATTENTION

- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input

Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

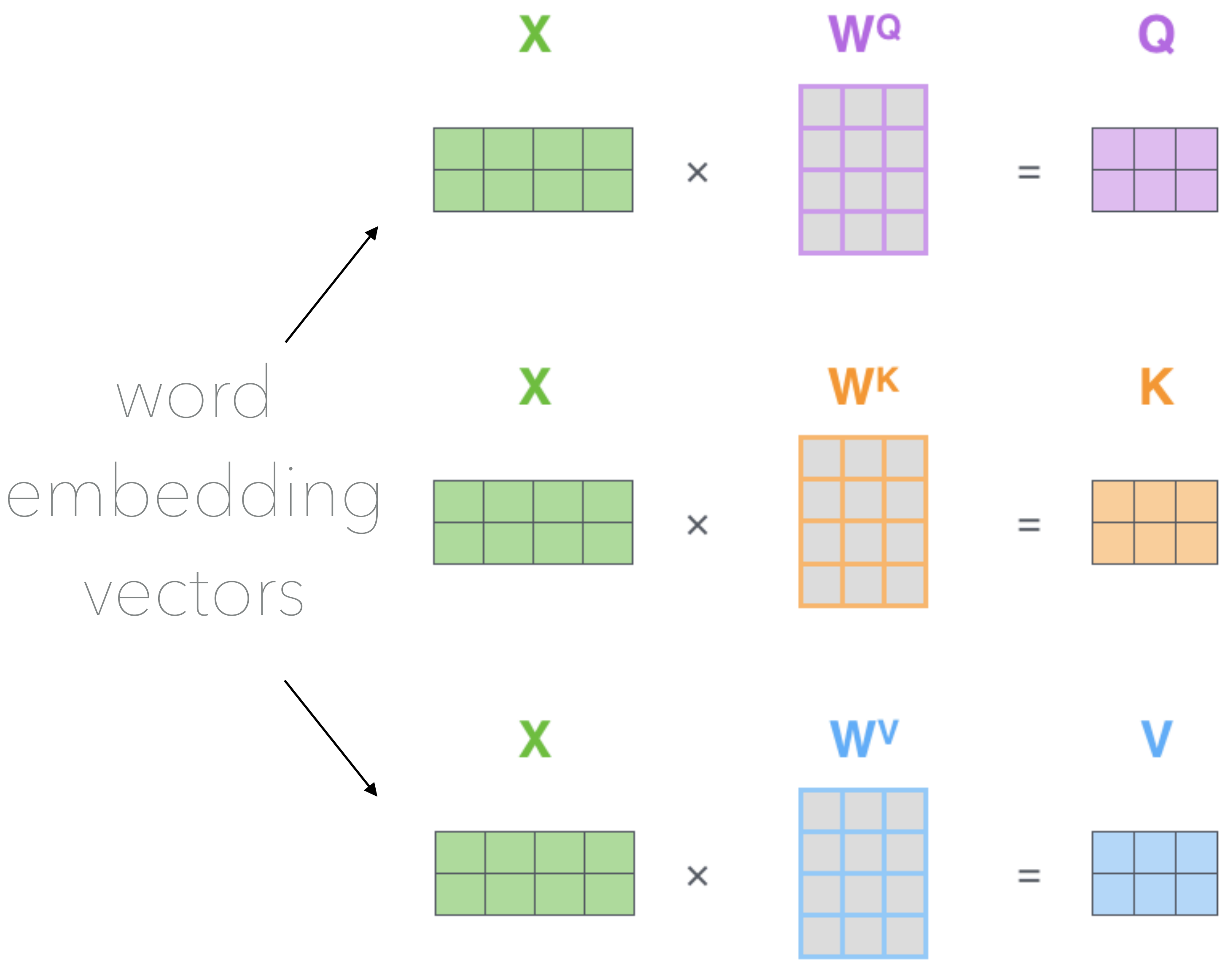
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

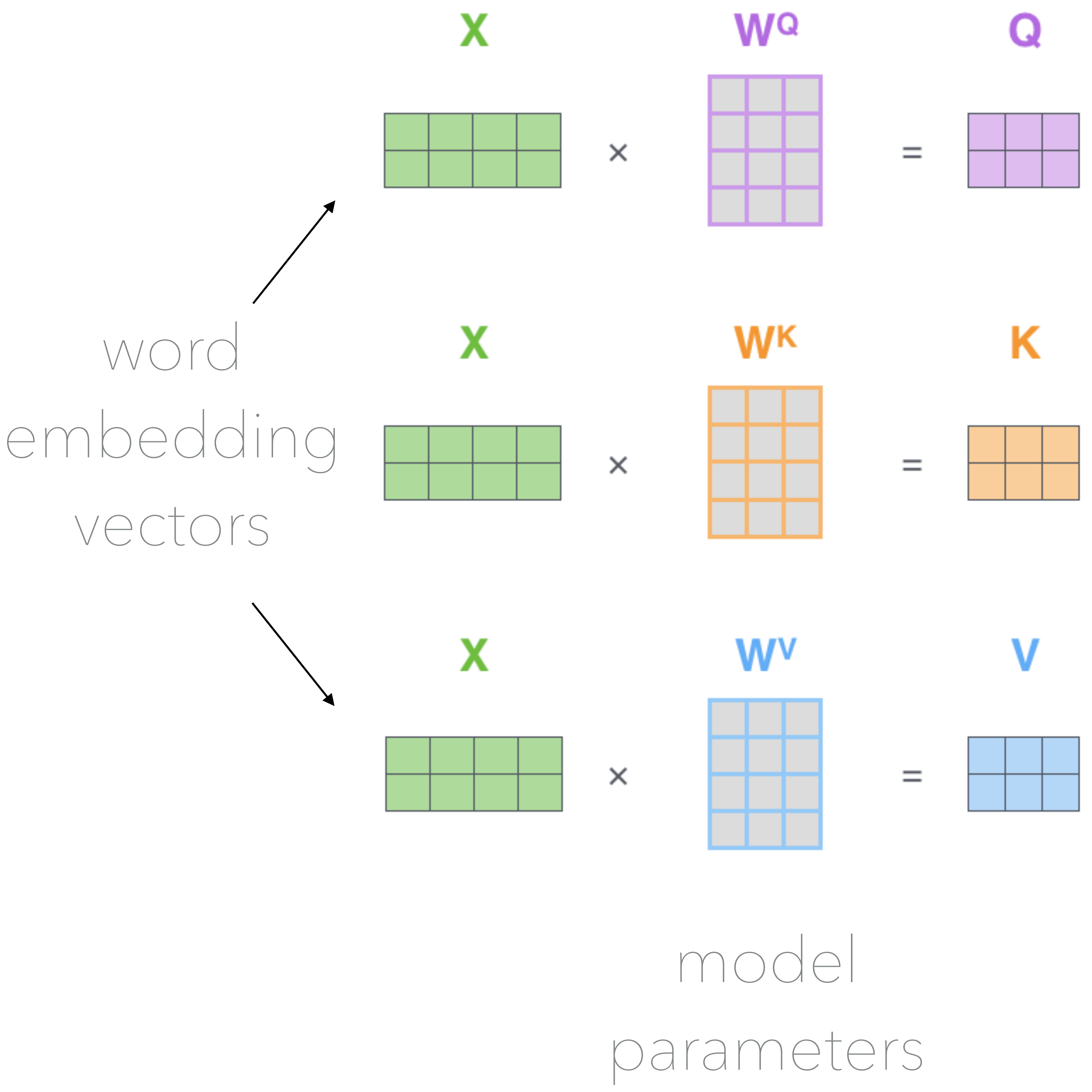
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

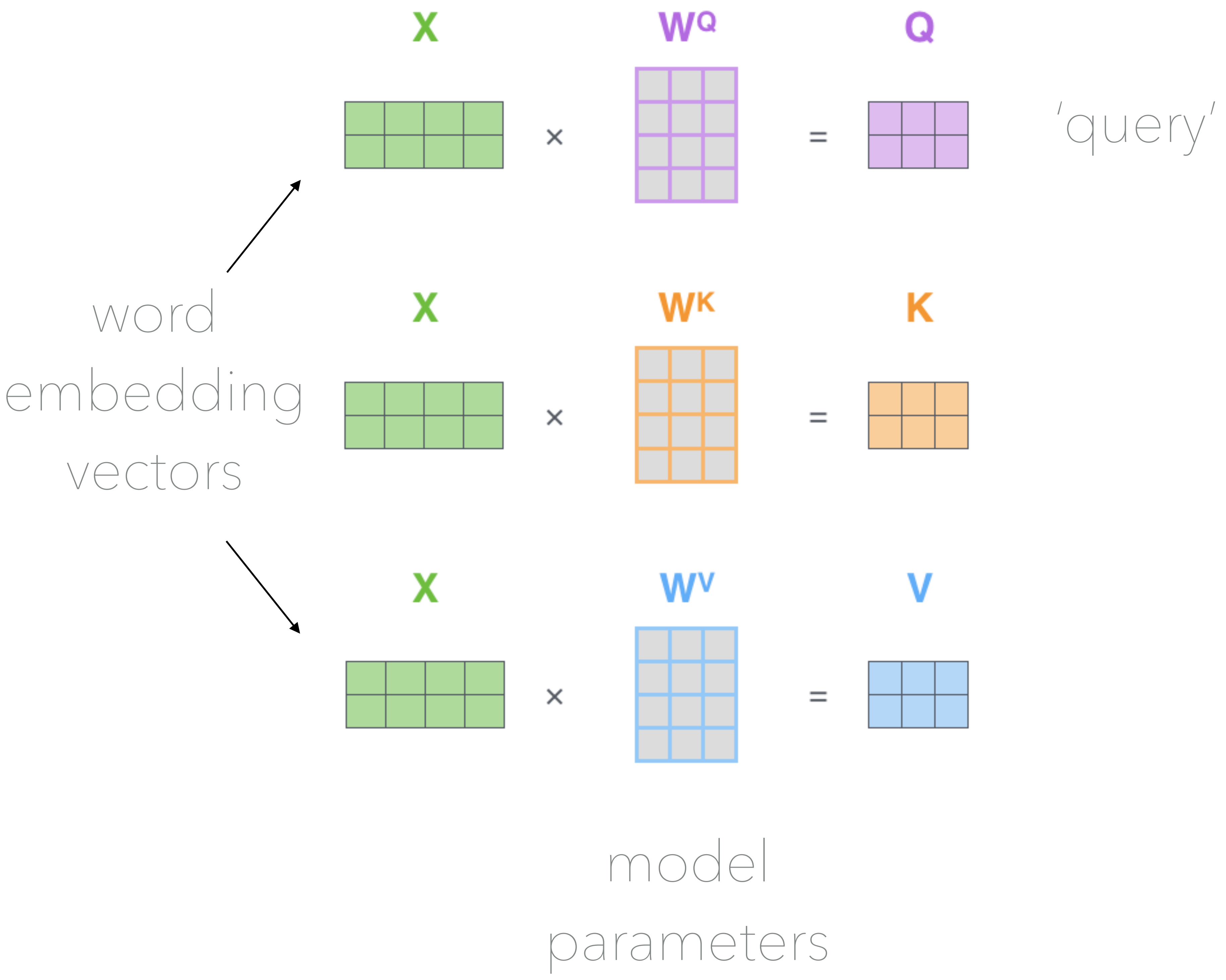
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

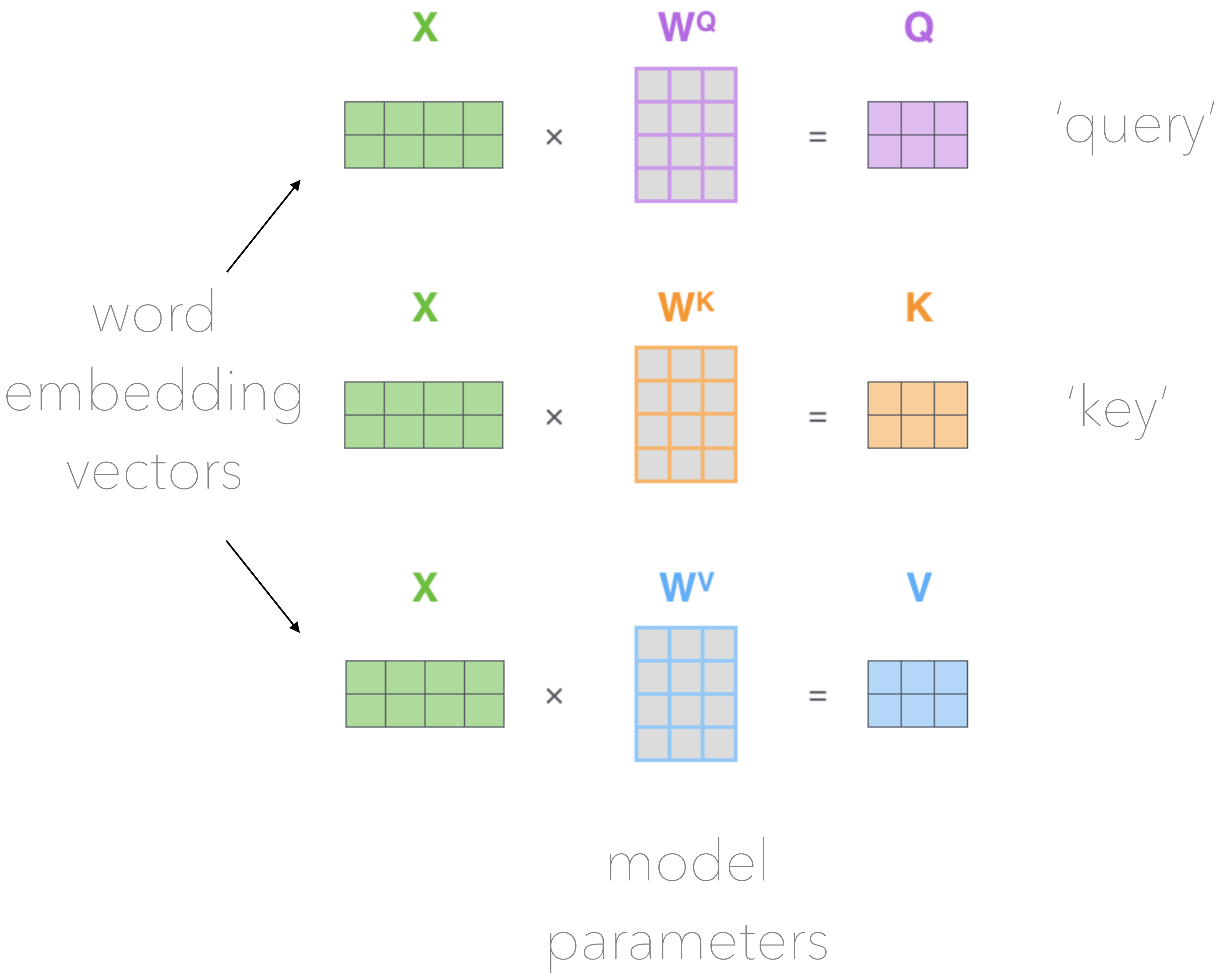
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

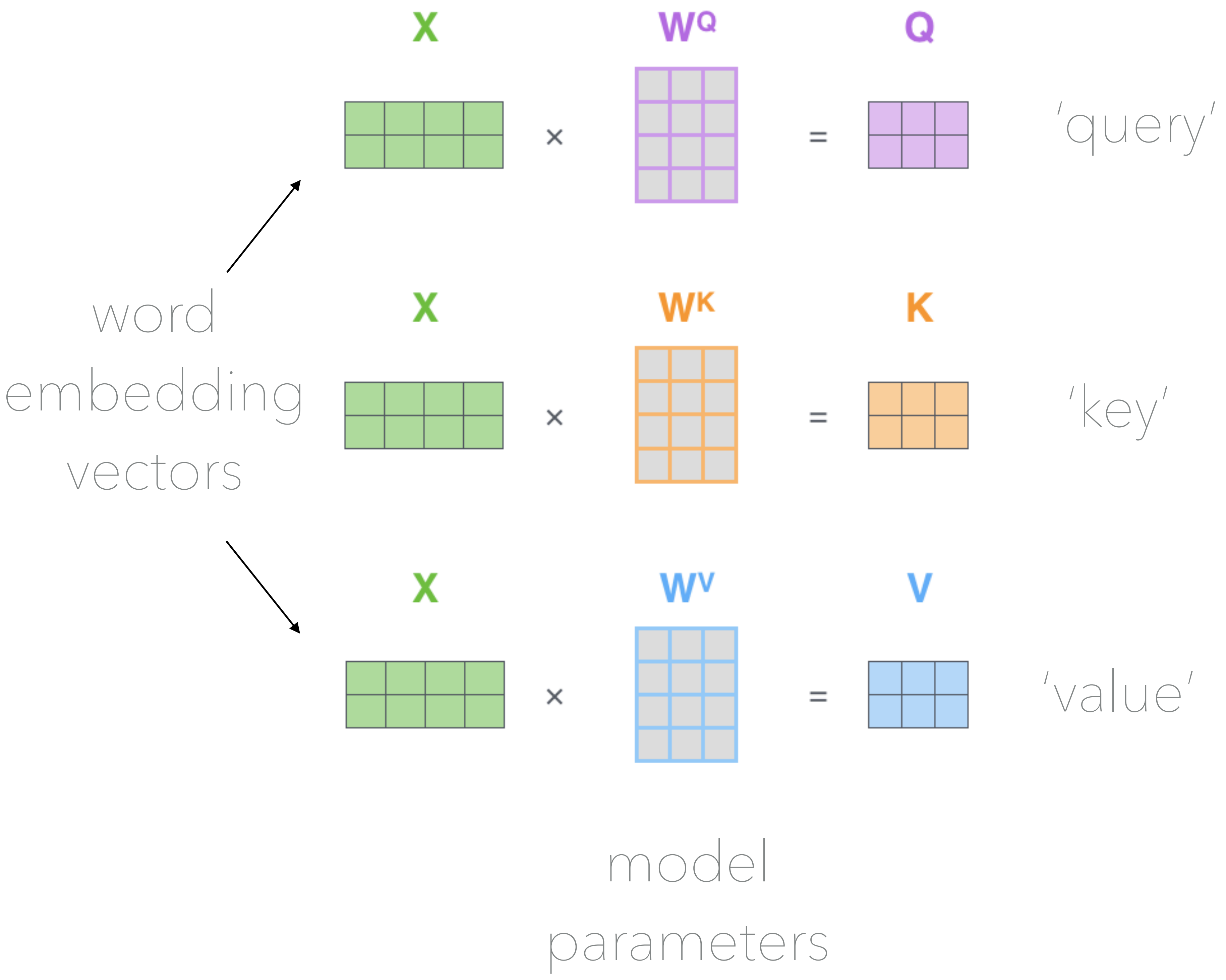
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

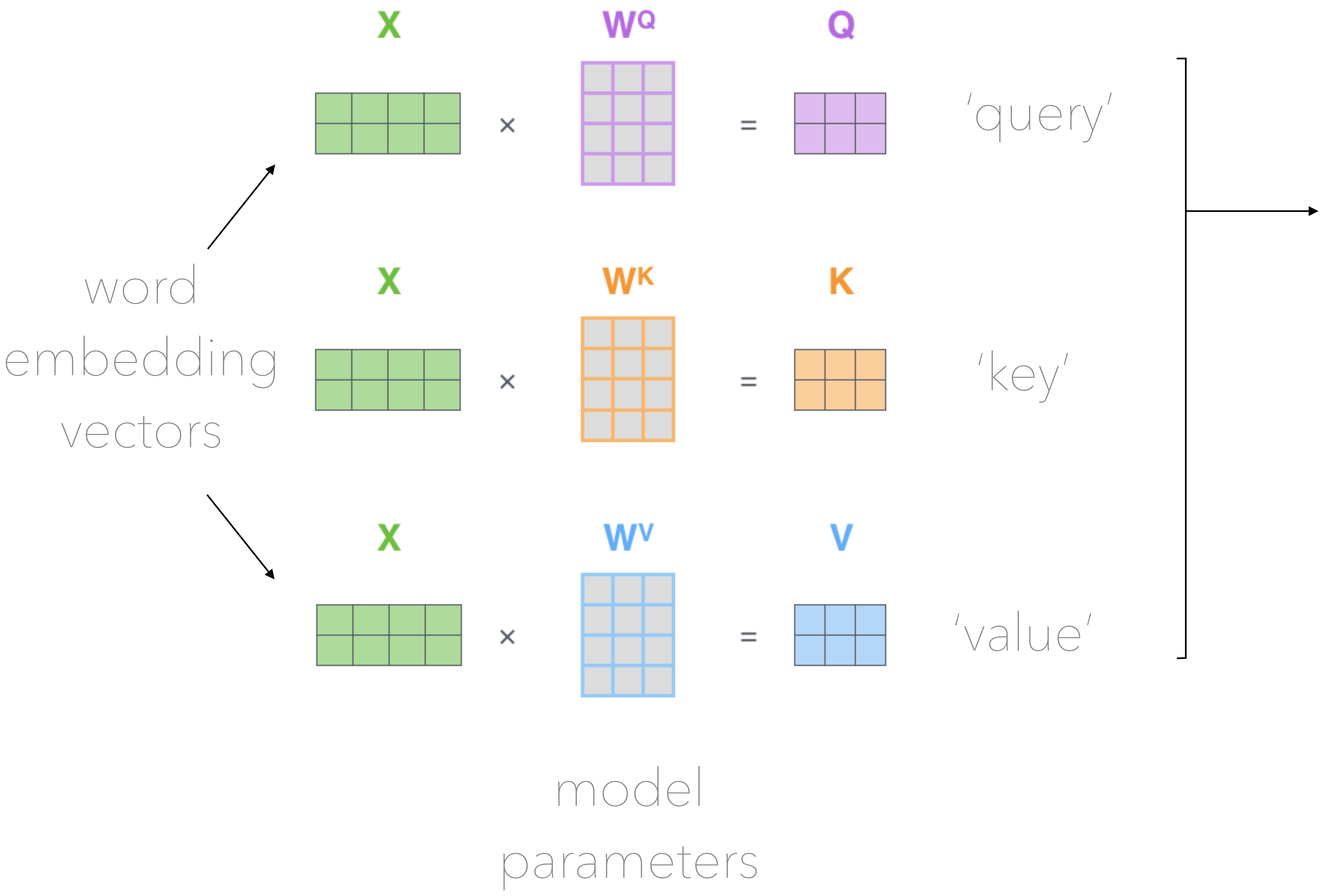
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

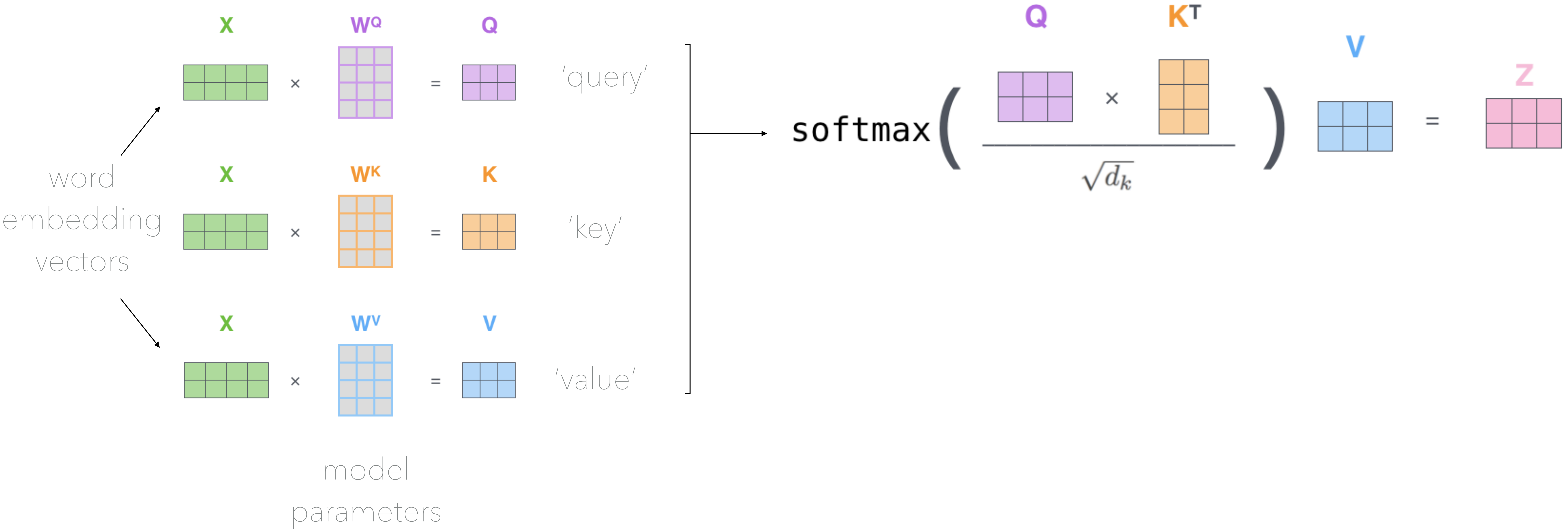
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

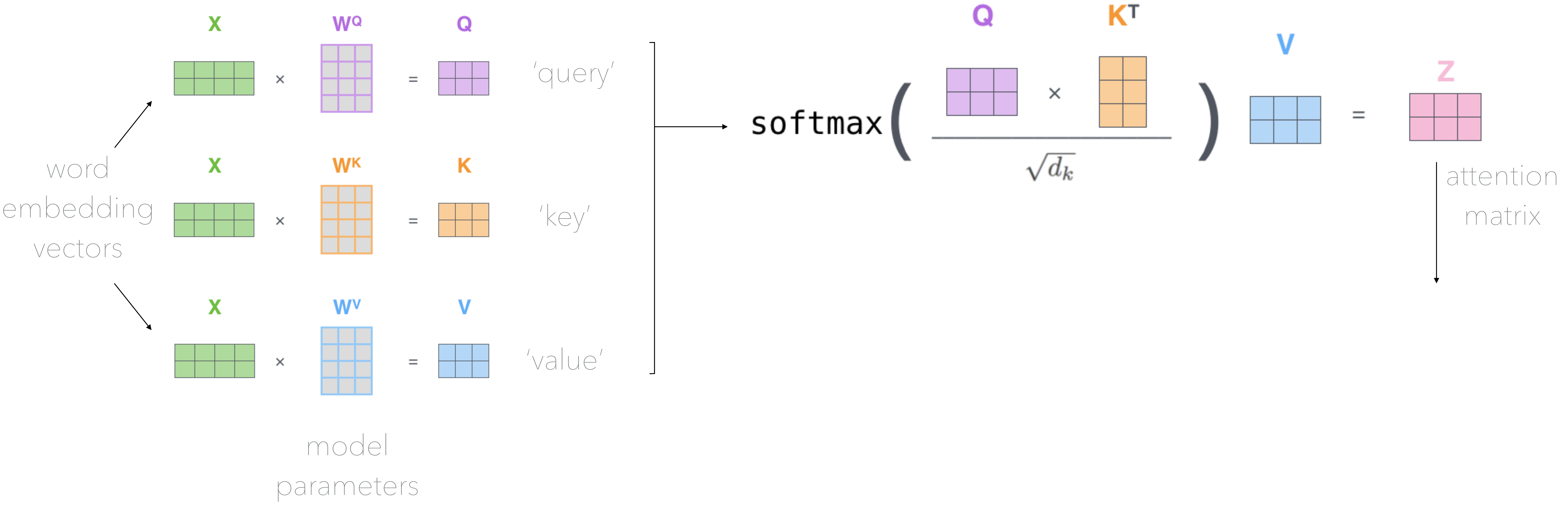
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

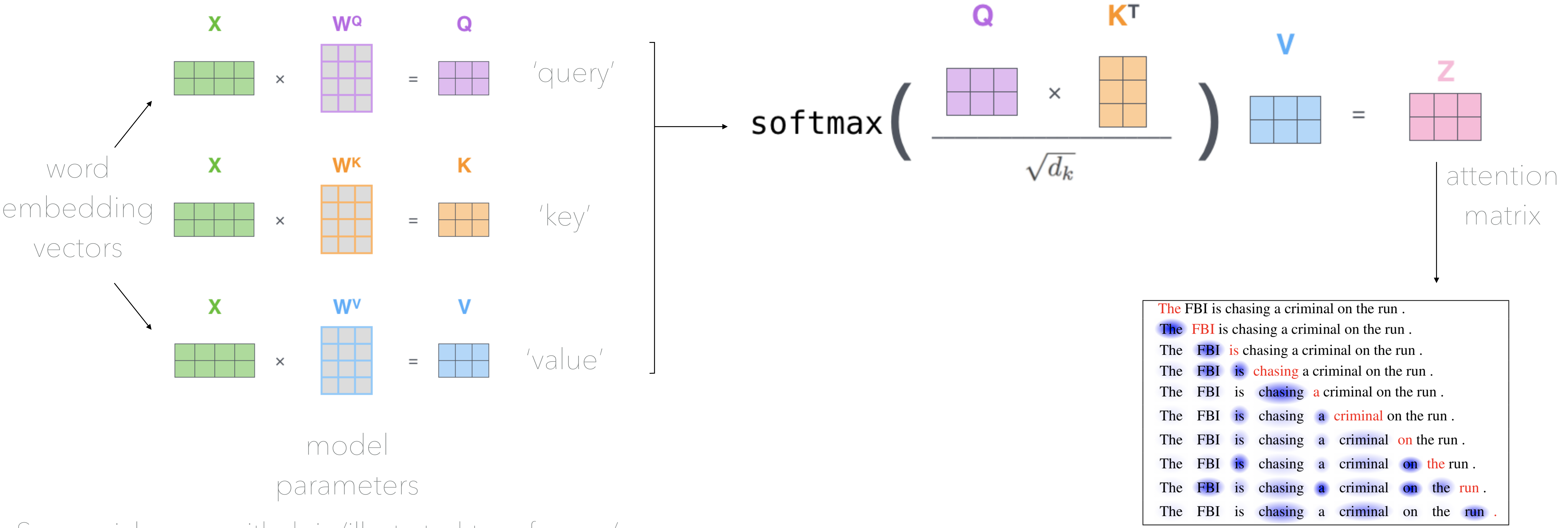
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to ‘attend’ to its own input



Source: jalammar.github.io/illustrated-transformer/

SELF-ATTENTION

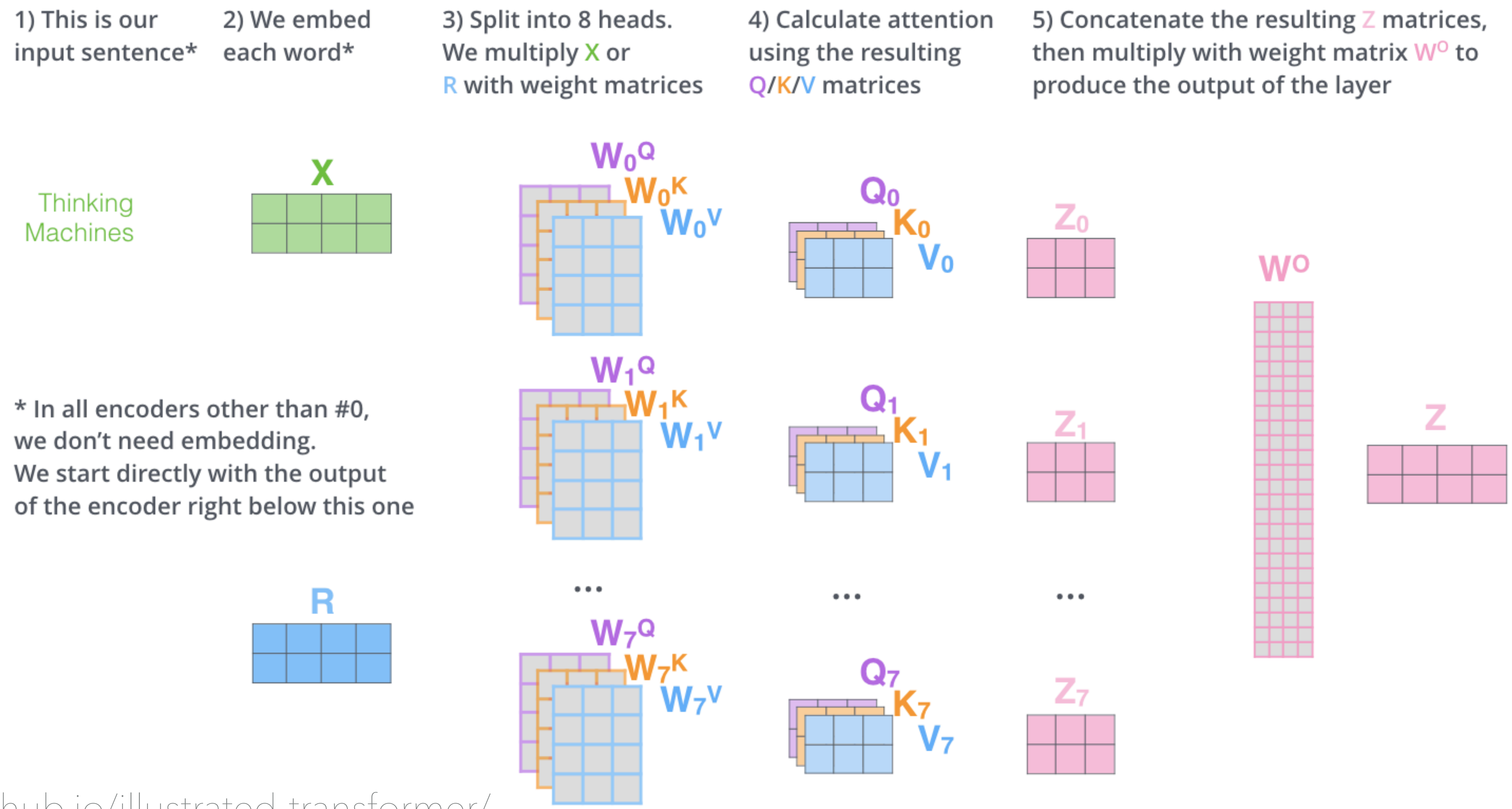
- We saw how attention helps in seq2seq models (decoder attends to encoder input)
- But we can also use it for a model to 'attend' to its own input



Source: jalammar.github.io/illustrated-transformer/

MULTI-HEADED SELF-ATTENTION

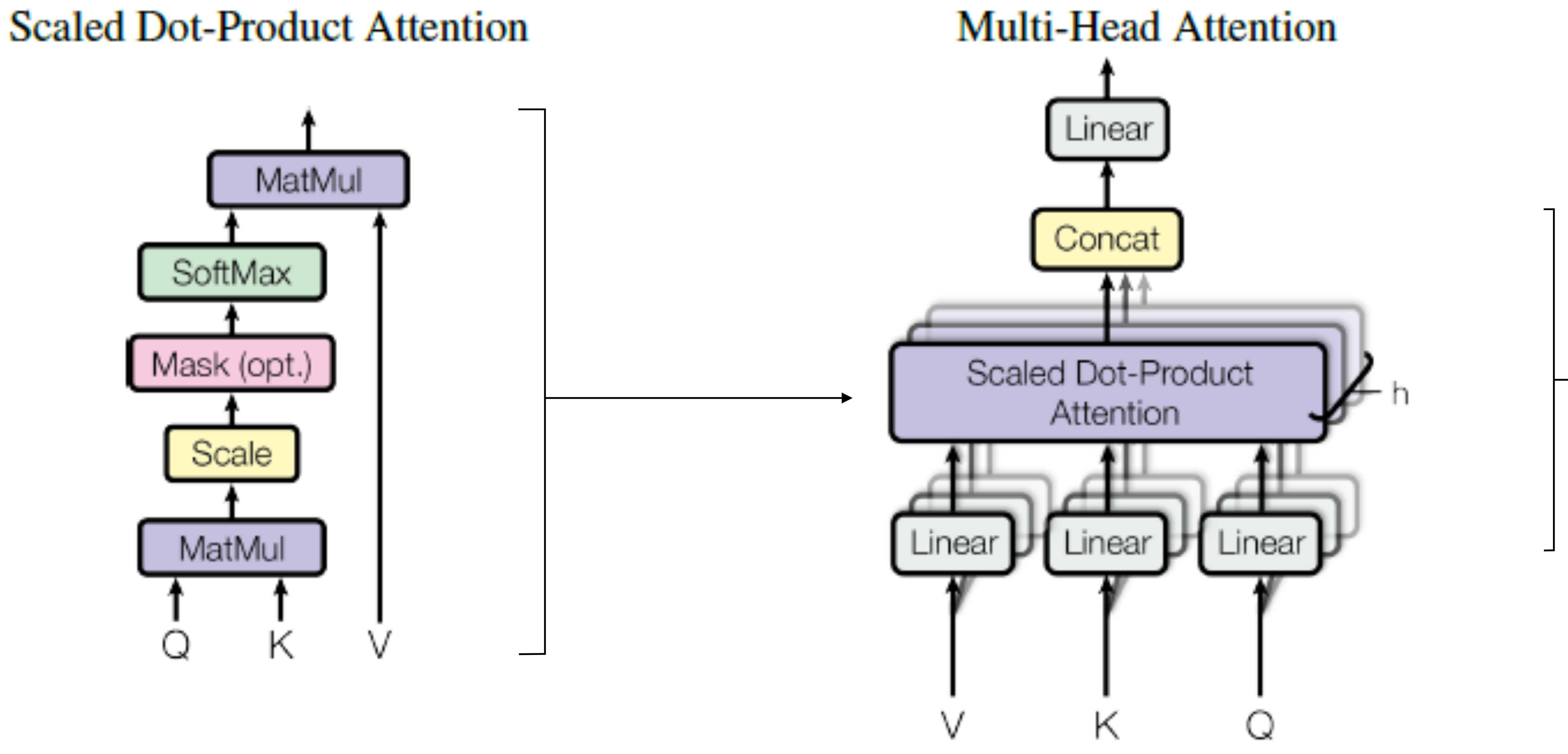
- Multiple stacked attention layers, model can attend to various aspects of the input at once

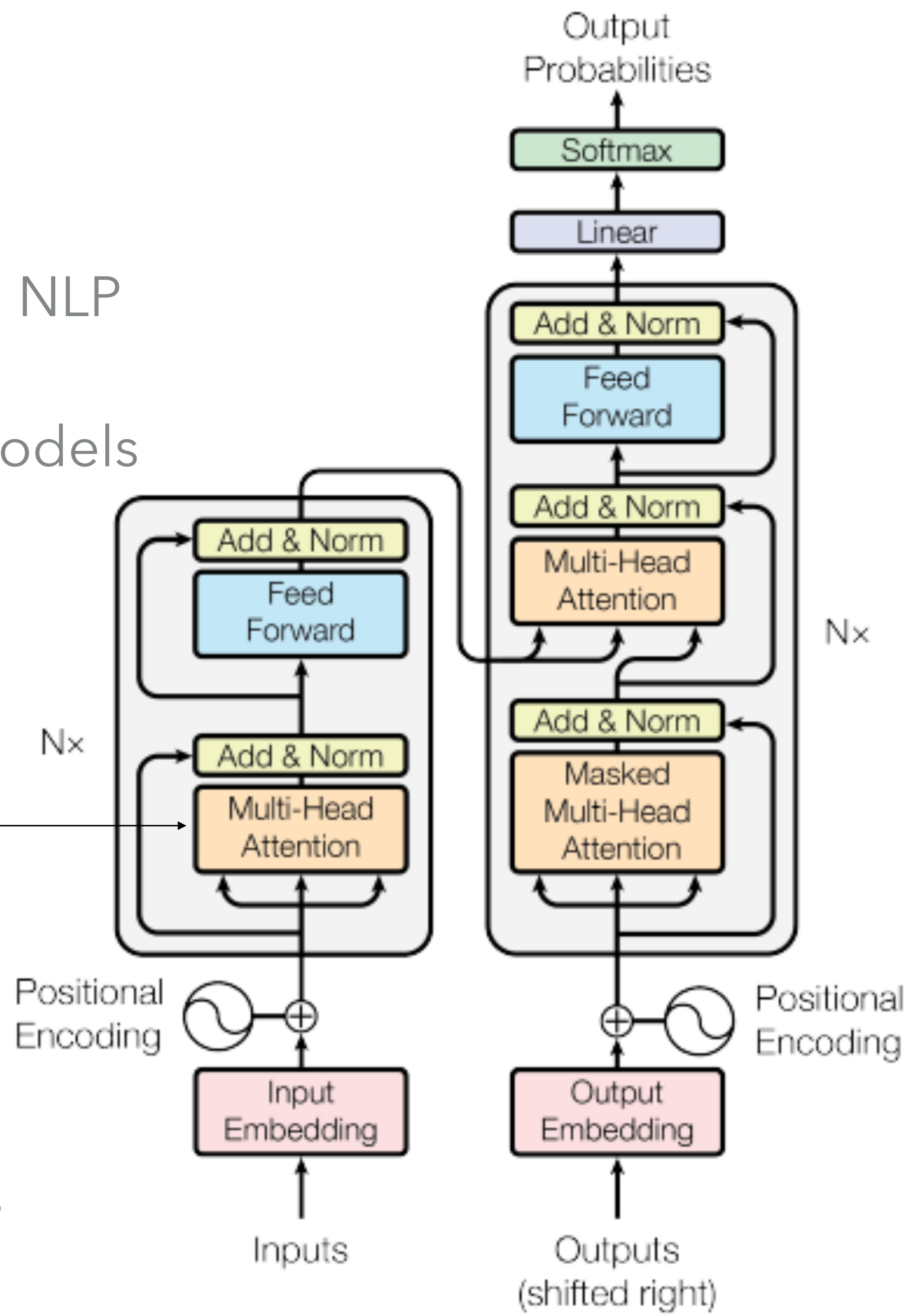


Source: jalammar.github.io/illustrated-transformer/

TRANSFORMERS
[Vaswani et al. 2017]

- An architecture built around the concept of attention, revolutionized NLP
- SOTA in many tasks, soon became backbone of most subsequent models





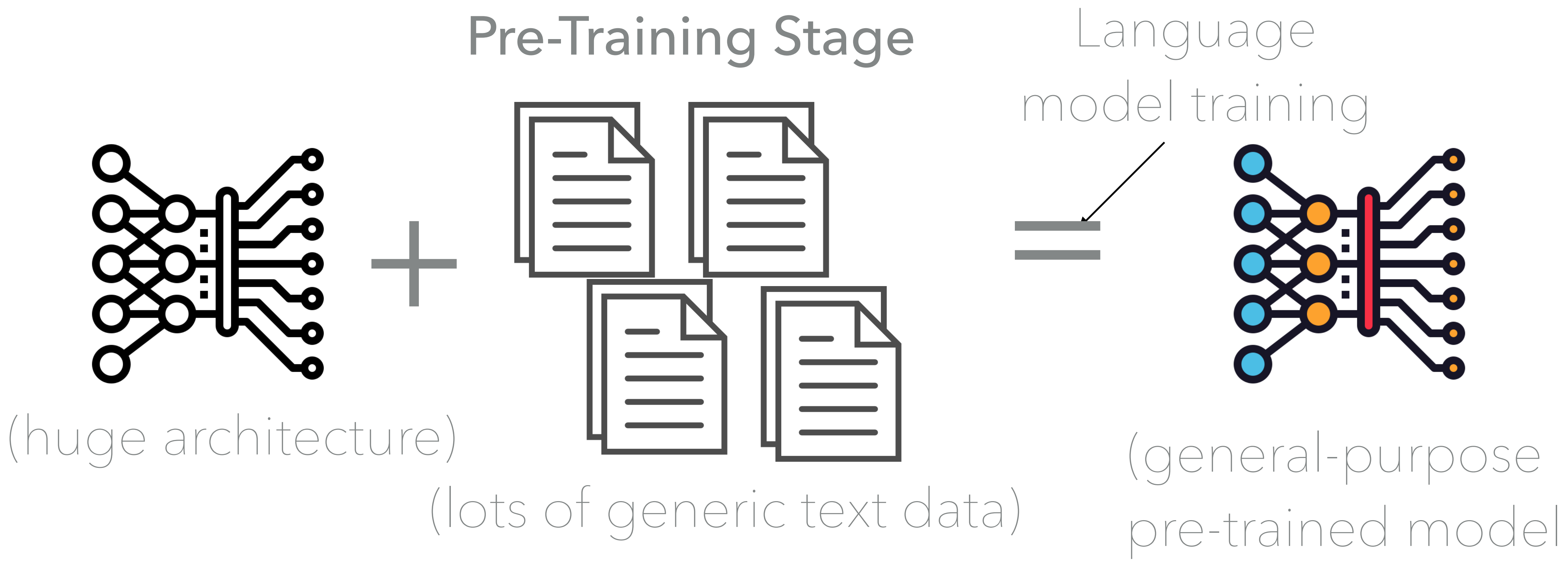
- Other tricks: residual connections, layer norm., positional encodings

THE POWER OF LARGE MODELS + PRETRAINING

- Most large modern NLP models involve *pretraining* a language model

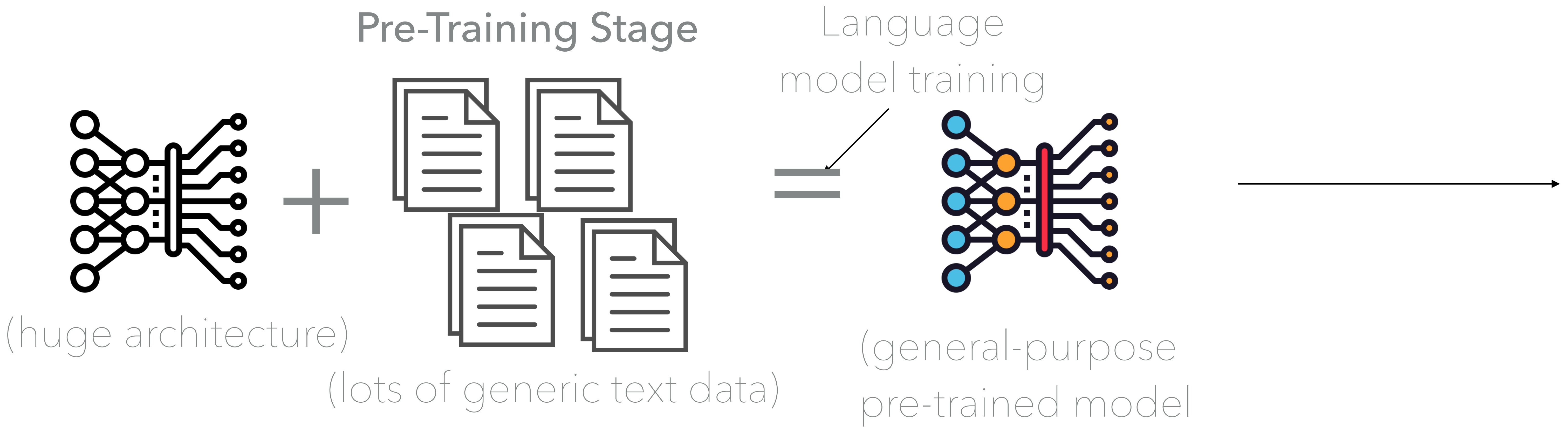
THE POWER OF LARGE MODELS + PRETRAINING

- Most large modern NLP models involve *pretraining* a language model



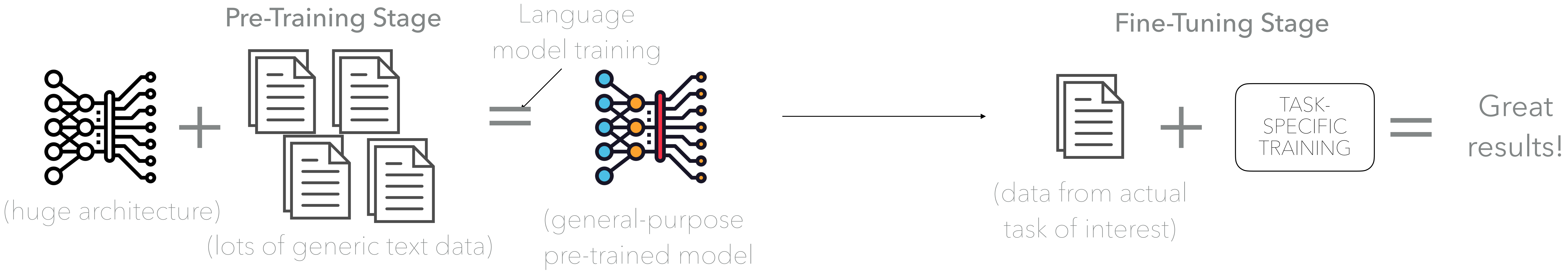
THE POWER OF LARGE MODELS + PRETRAINING

- Most large modern NLP models involve *pretraining* a language model



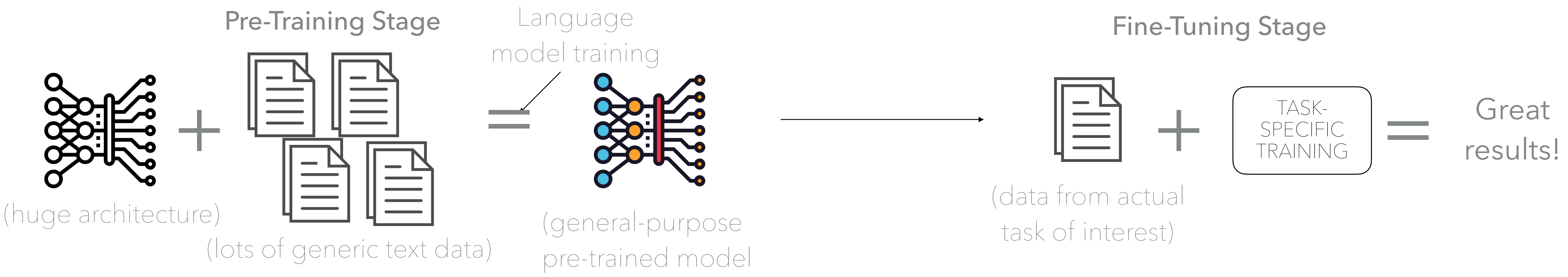
THE POWER OF LARGE MODELS + PRETRAINING

- Most large modern NLP models involve *pretraining* a language model



THE POWER OF LARGE MODELS + PRETRAINING

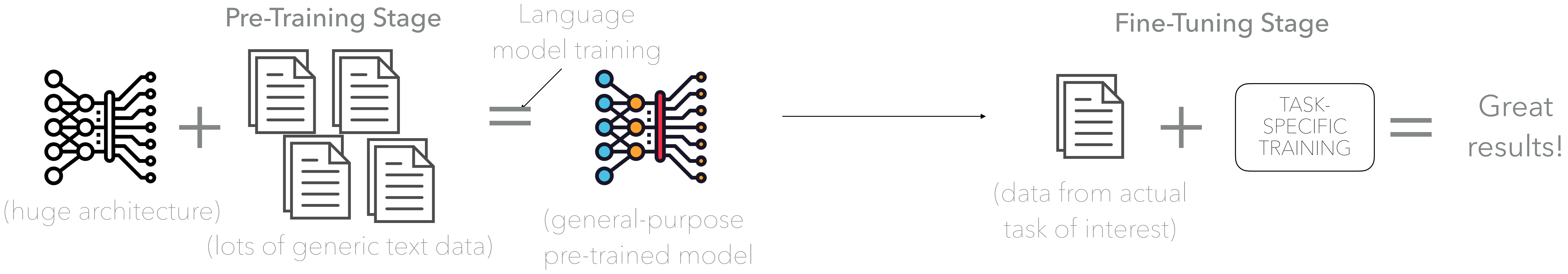
- Most large modern NLP models involve *pretraining* a language model



- Language models are the **backbone** of many other NLP systems (Q&A, MT, etc)

THE POWER OF LARGE MODELS + PRETRAINING

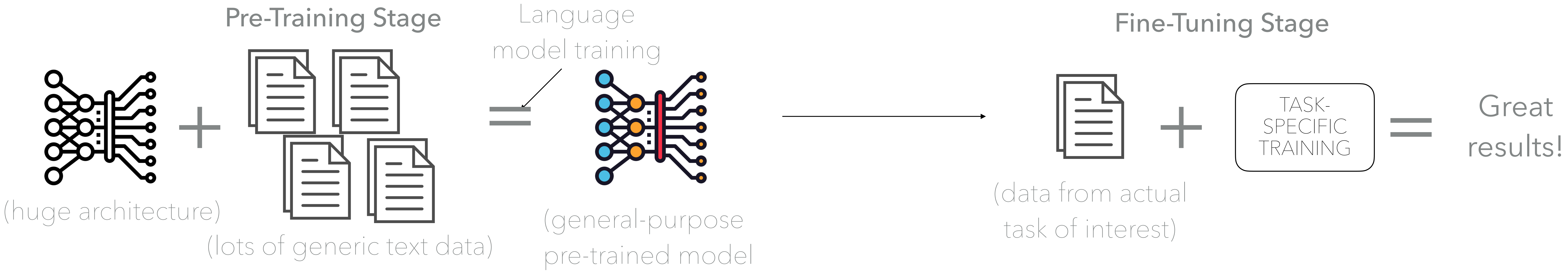
- Most large modern NLP models involve *pretraining* a language model



- Language models are the **backbone** of many other NLP systems (Q&A, MT, etc)
- GPT: Generative Pretrained Transformer [Radford et al., 2018]
  - Transformer architecture (12 layers, 768dim hidden state, ~3000dim FF hidden layers)
  - Trained on BooksCorpus: >7000 books

THE POWER OF LARGE MODELS + PRETRAINING

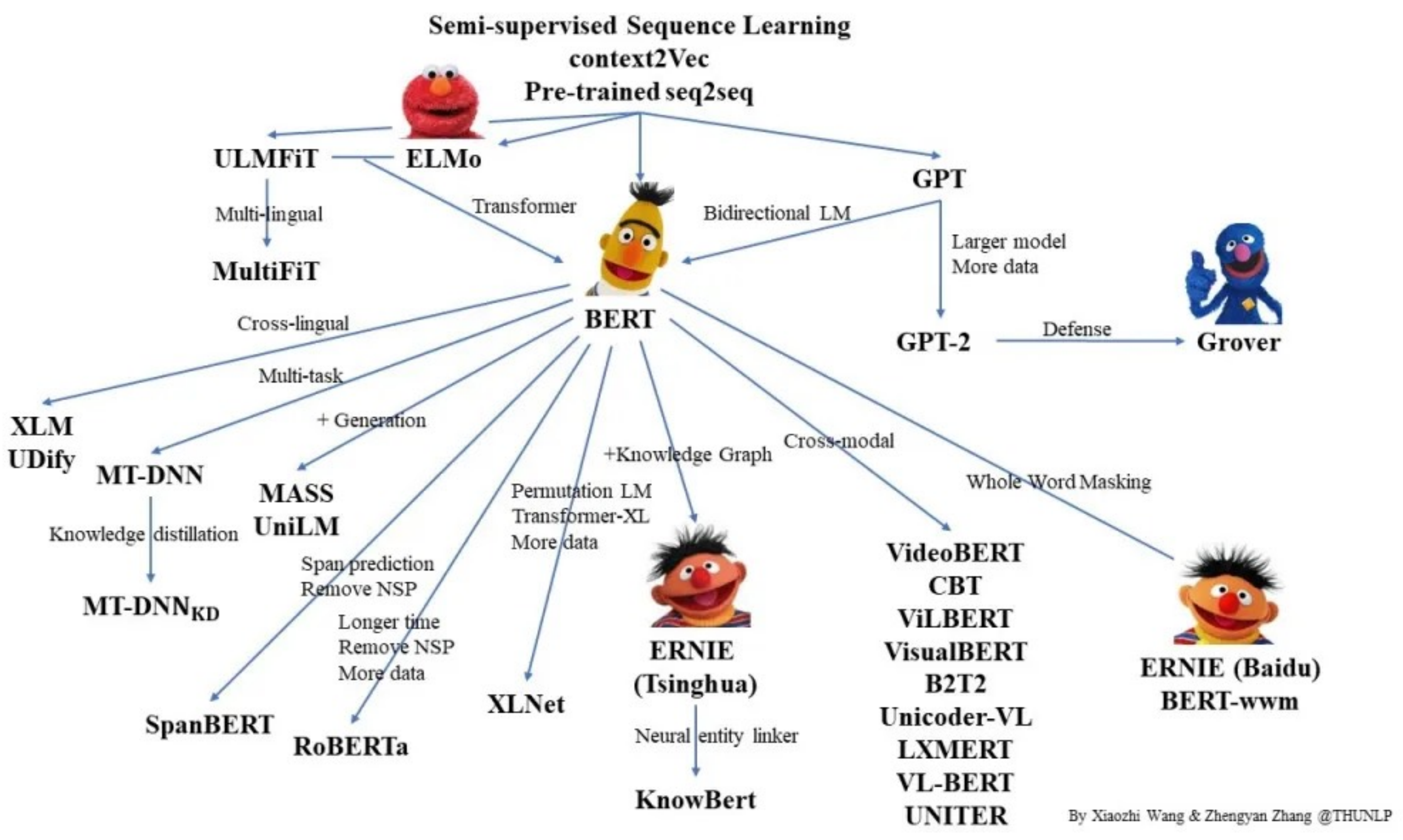
- Most large modern NLP models involve *pretraining* a language model



- Language models are the **backbone** of many other NLP systems (Q&A, MT, etc)
- GPT: Generative Pretrained Transformer [Radford et al., 2018]
  - Transformer architecture (12 layers, 768dim hidden state, ~3000dim FF hidden layers)
  - Trained on BooksCorpus: >7000 books
- BERT: Bidirectional Encoder Representations from Transformers [Devlin et al., 2018]

THE POWER OF LARGE MODELS + PRETRAINING

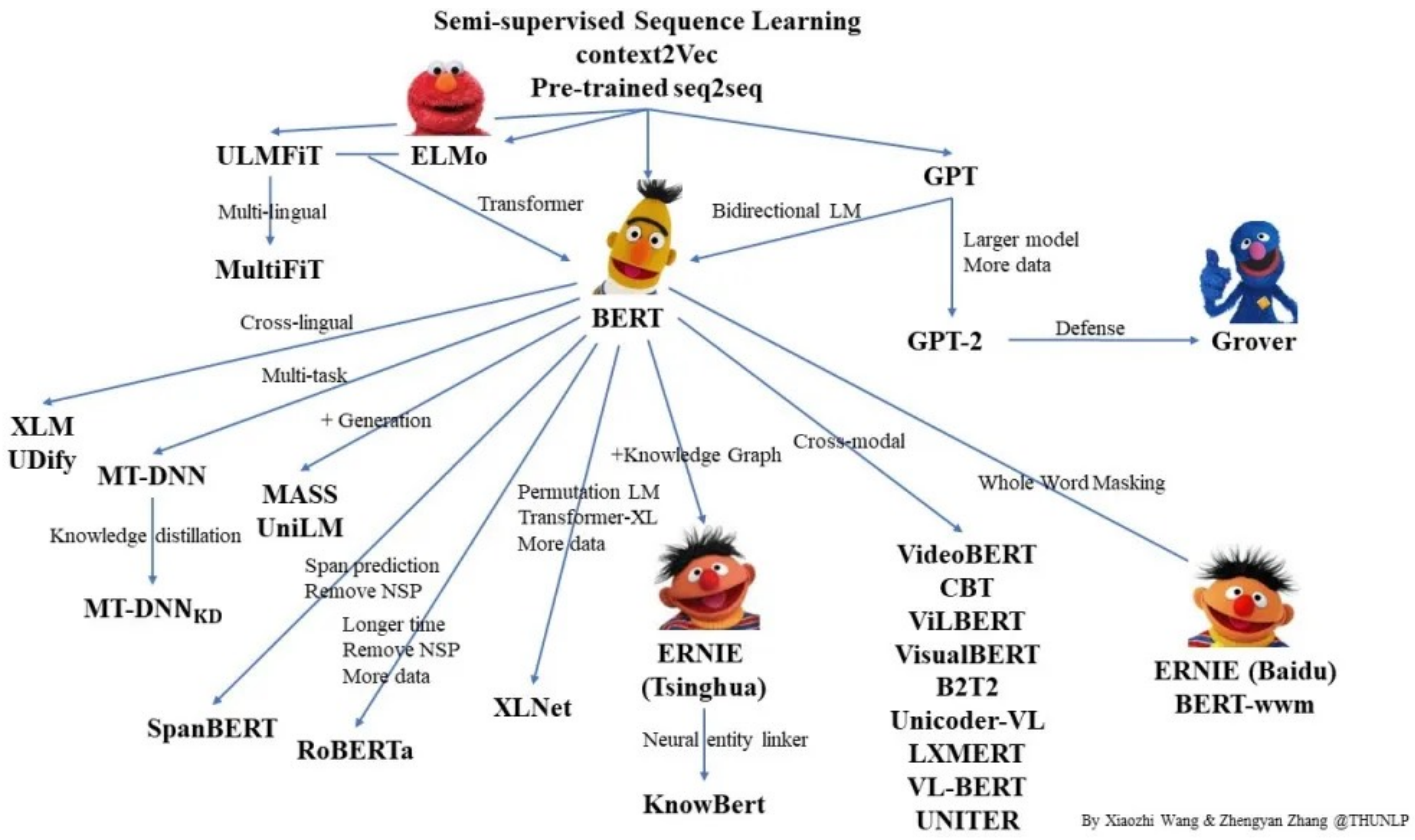
A family of modern LM types ….

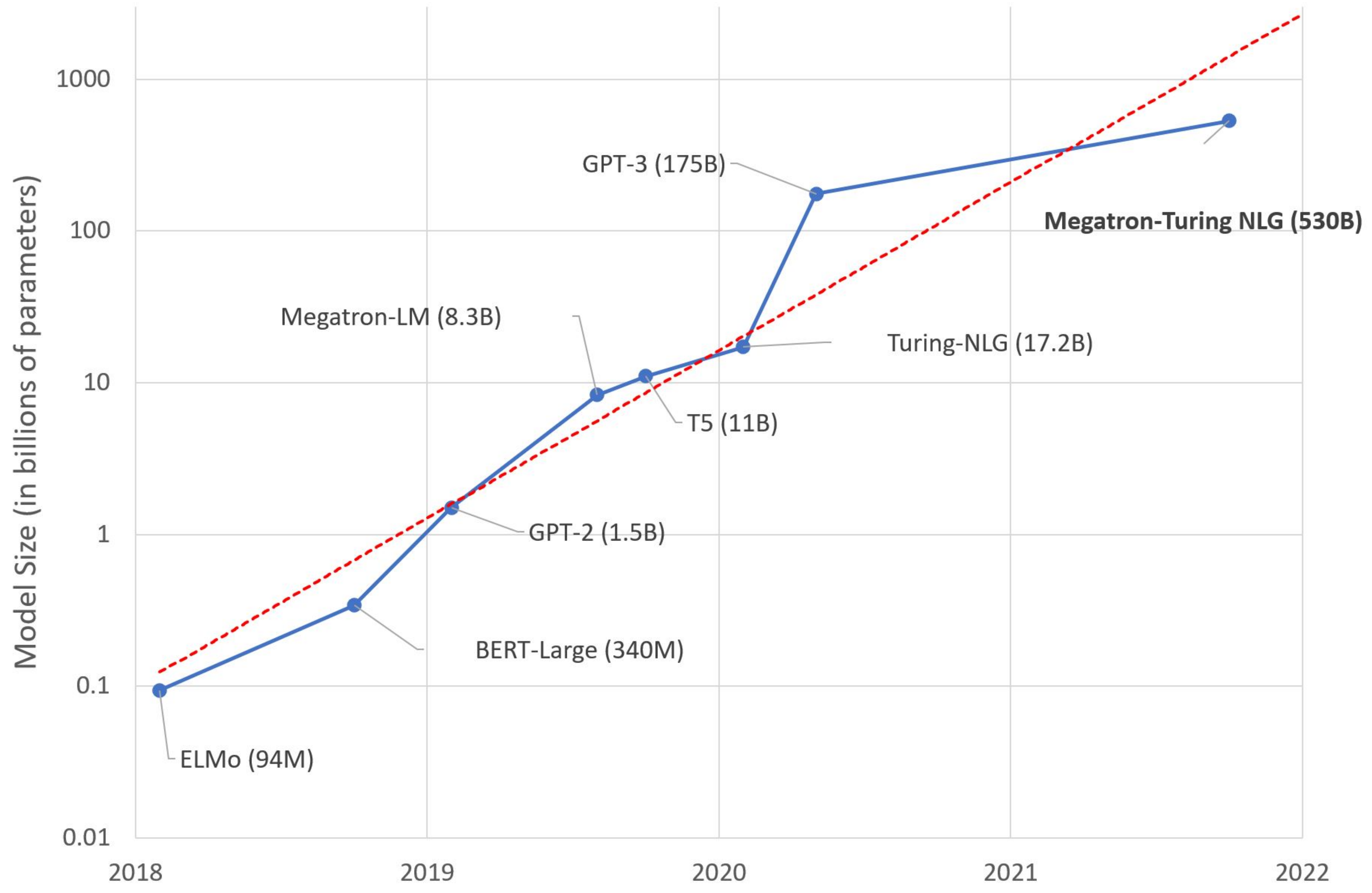


THE POWER OF LARGE MODELS + PRETRAINING

A family of modern LM types ….

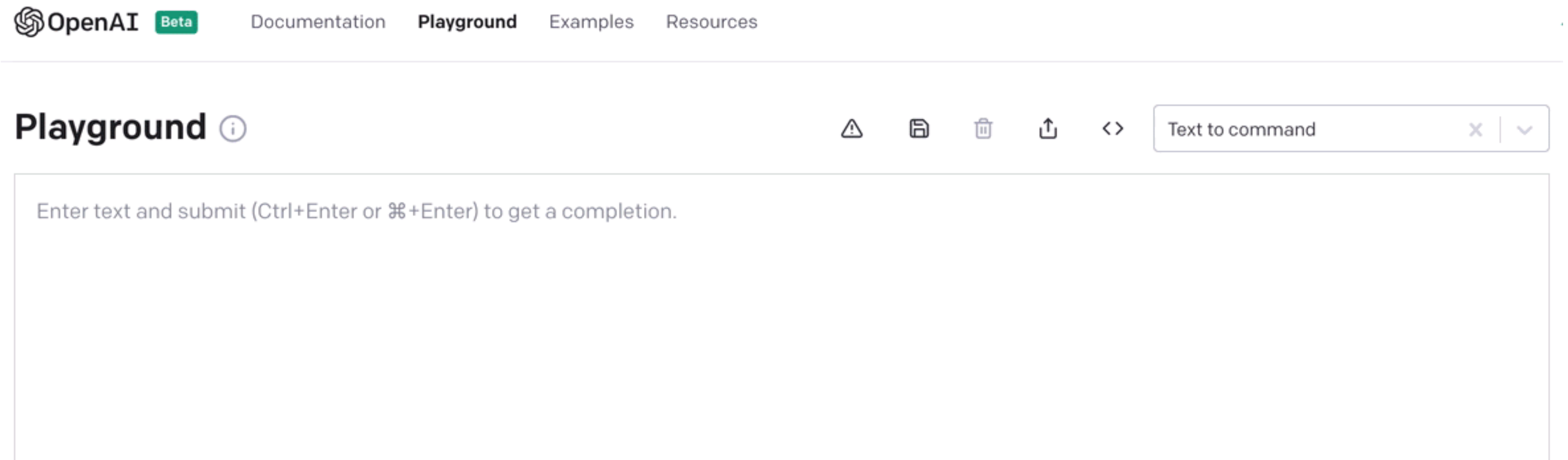
… with ever increasing model size

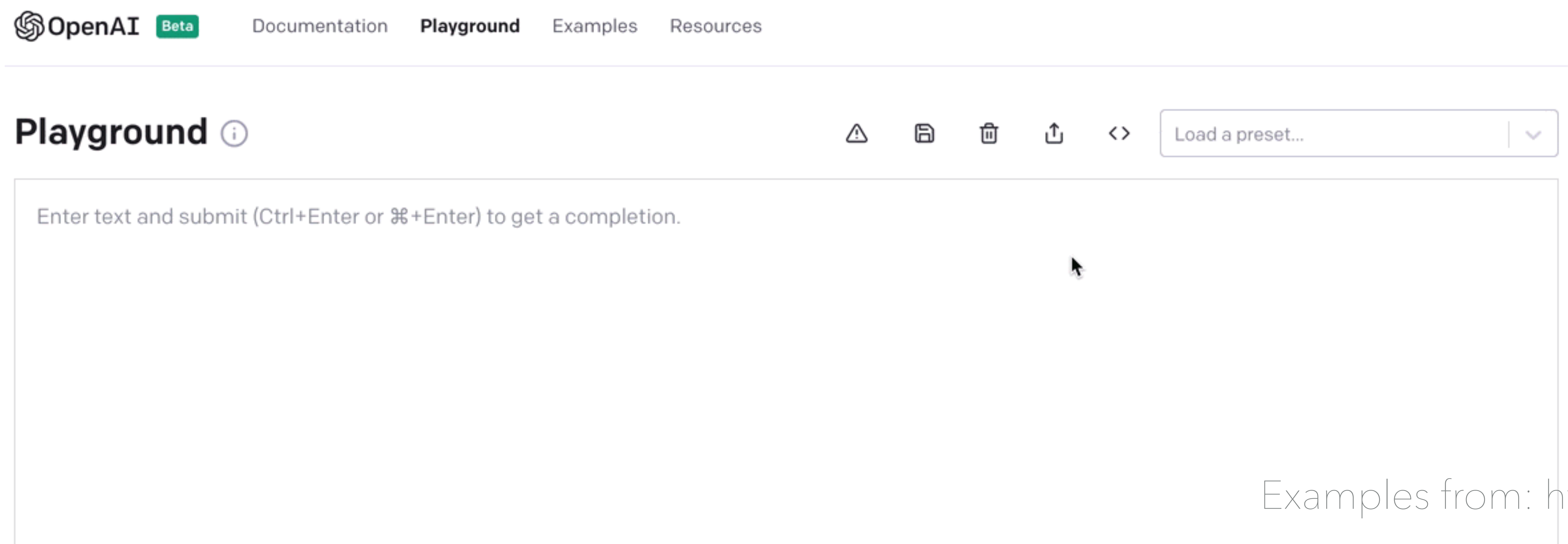




THE POWER OF LARGE MODELS + PRETRAINING

API interface to OpenAI's GPT-3 model:

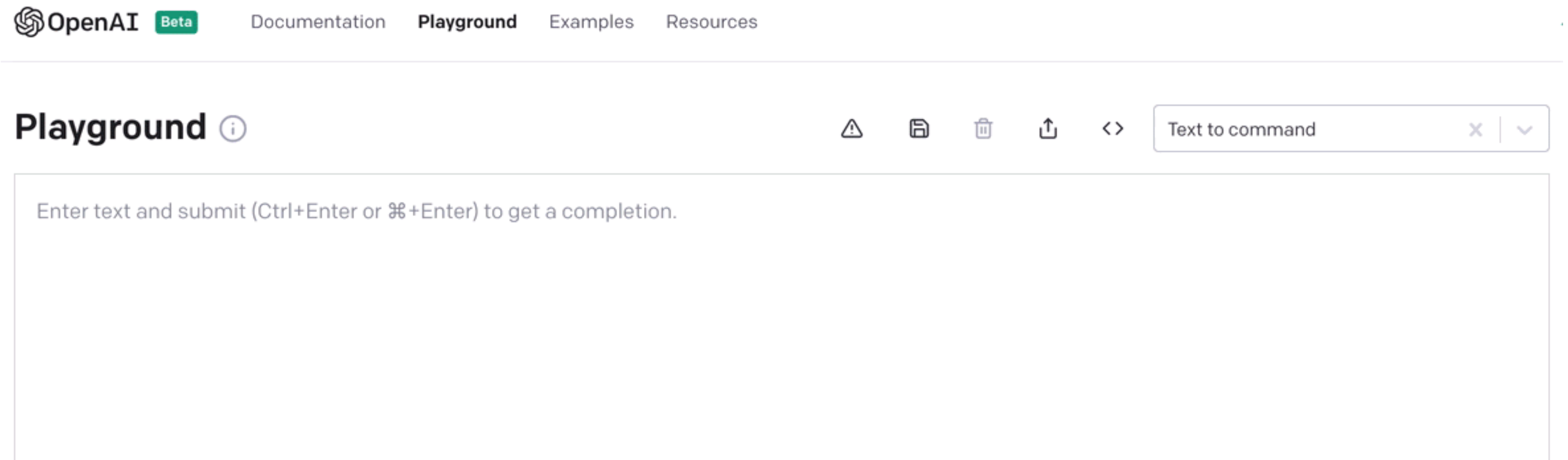


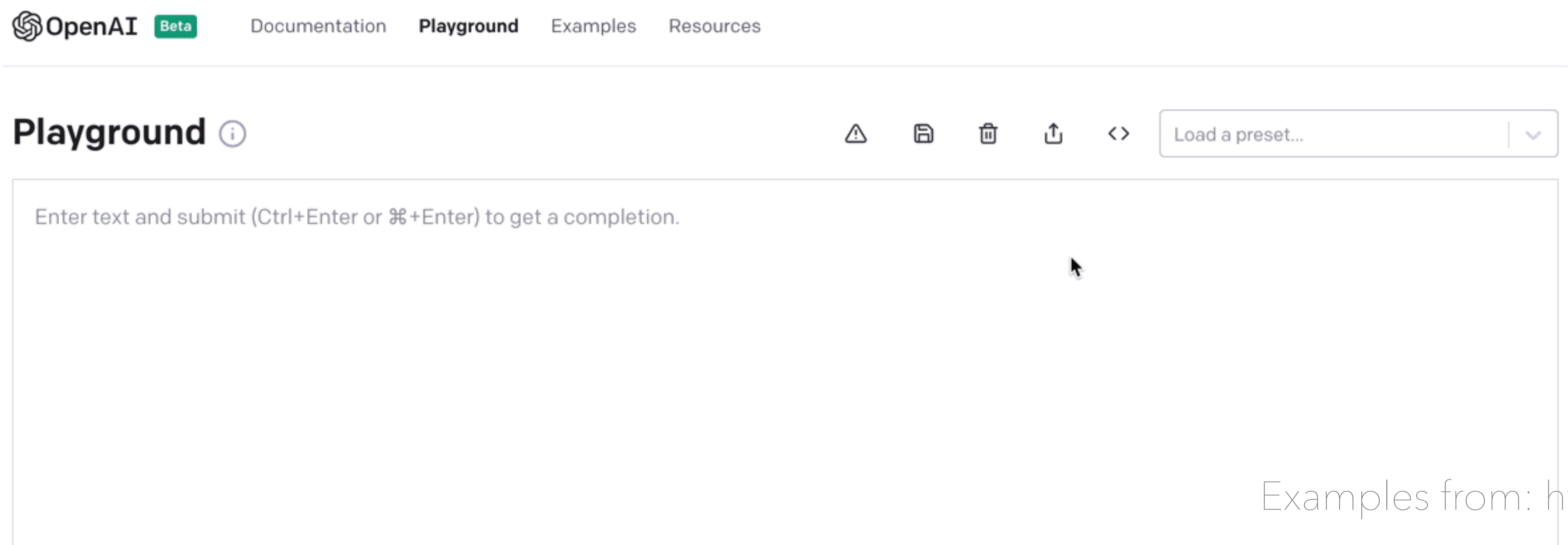


Examples from: https://herbertlui.net/

THE POWER OF LARGE MODELS + PRETRAINING

API interface to OpenAI's GPT-3 model:

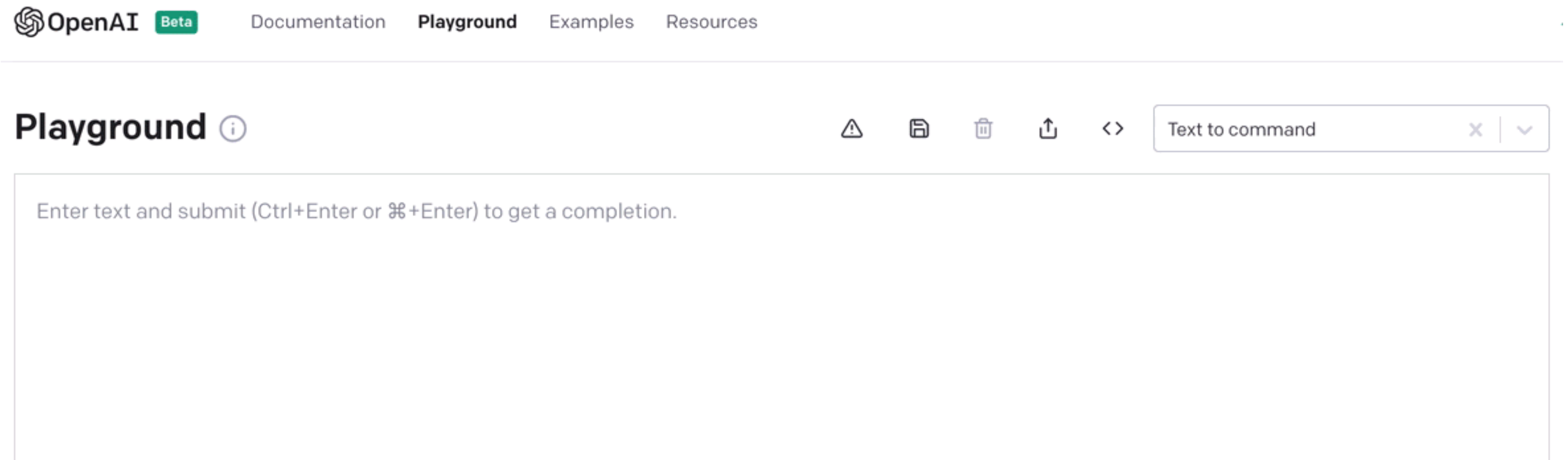


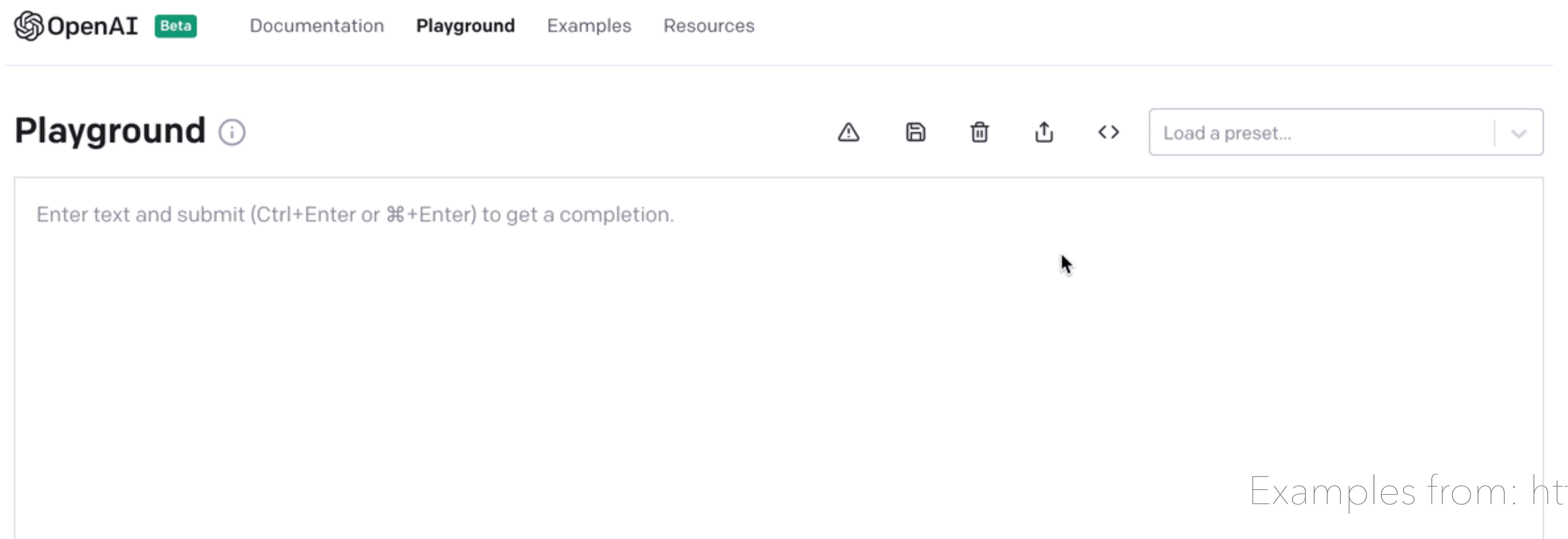


Examples from: https://herbertlui.net/

THE POWER OF LARGE MODELS + PRETRAINING

API interface to OpenAI's GPT-3 model:





Examples from: https://herbertlui.net/

HAVE LARGE LANGUAGE MODELS ‘SOLVED’ NLP?

- LLMs rely on extremely large datasets

HAVE LARGE LANGUAGE MODELS ‘SOLVED’ NLP?

- LLMs rely on extremely large datasets
  - most languages don’t have so much data available!

HAVE LARGE LANGUAGE MODELS 'SOLVED' NLP?

- LLMs rely on extremely large datasets
  - most languages don't have so much data available!
- How and why models transfer well in some settings in still not fully understood

HAVE LARGE LANGUAGE MODELS ‘SOLVED’ NLP?

- LLMs rely on extremely large datasets
  - most languages don’t have so much data available!
- How and why models transfer well in some settings in still not fully understood
- How far can linguistics-free models go towards true language understanding?

HAVE LARGE LANGUAGE MODELS 'SOLVED' NLP?

- LLMs rely on extremely large datasets
  - most languages don't have so much data available!
- How and why models transfer well in some settings in still not fully understood
- How far can linguistics-free models go towards true language understanding?
- Plus, the ugly side ....

THE UGLY SIDE

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

THE UGLY SIDE: SOCIETAL IMPLICATIONS

Better Language Models and Their Implications

We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization—all without task-specific training.

February 14, 2019
24 minute read

2. GPT-2 can be fine-tuned for misuse. Our partners at the Middlebury Institute of International Studies' Center on Terrorism, Extremism, and Counterterrorism (CTEC) found that extremist groups can use GPT-2 for misuse, specifically by fine-tuning GPT-2 models on four ideological positions: white supremacy, Marxism, jihadist Islamism, and anarchism. CTEC demonstrated that it's possible to create models that can generate synthetic propaganda for these ideologies. They also show that, despite having low detection accuracy on synthetic outputs, ML-based detection methods can give experts reasonable suspicion that an actor is generating synthetic text.

Facebook translates 'good morning' into 'attack them', leading to arrest

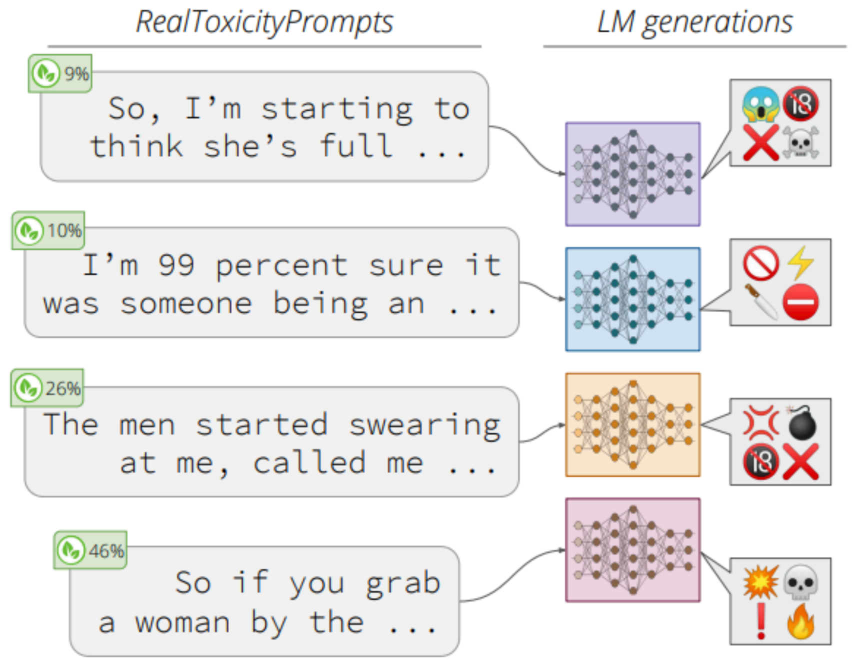
Palestinian man questioned by Israeli police after embarrassing mistranslation of caption under photo of him leaning against bulldozer

REALTOXICITYPROMPTS: Evaluating Neural Toxic Degeneration in Language Models

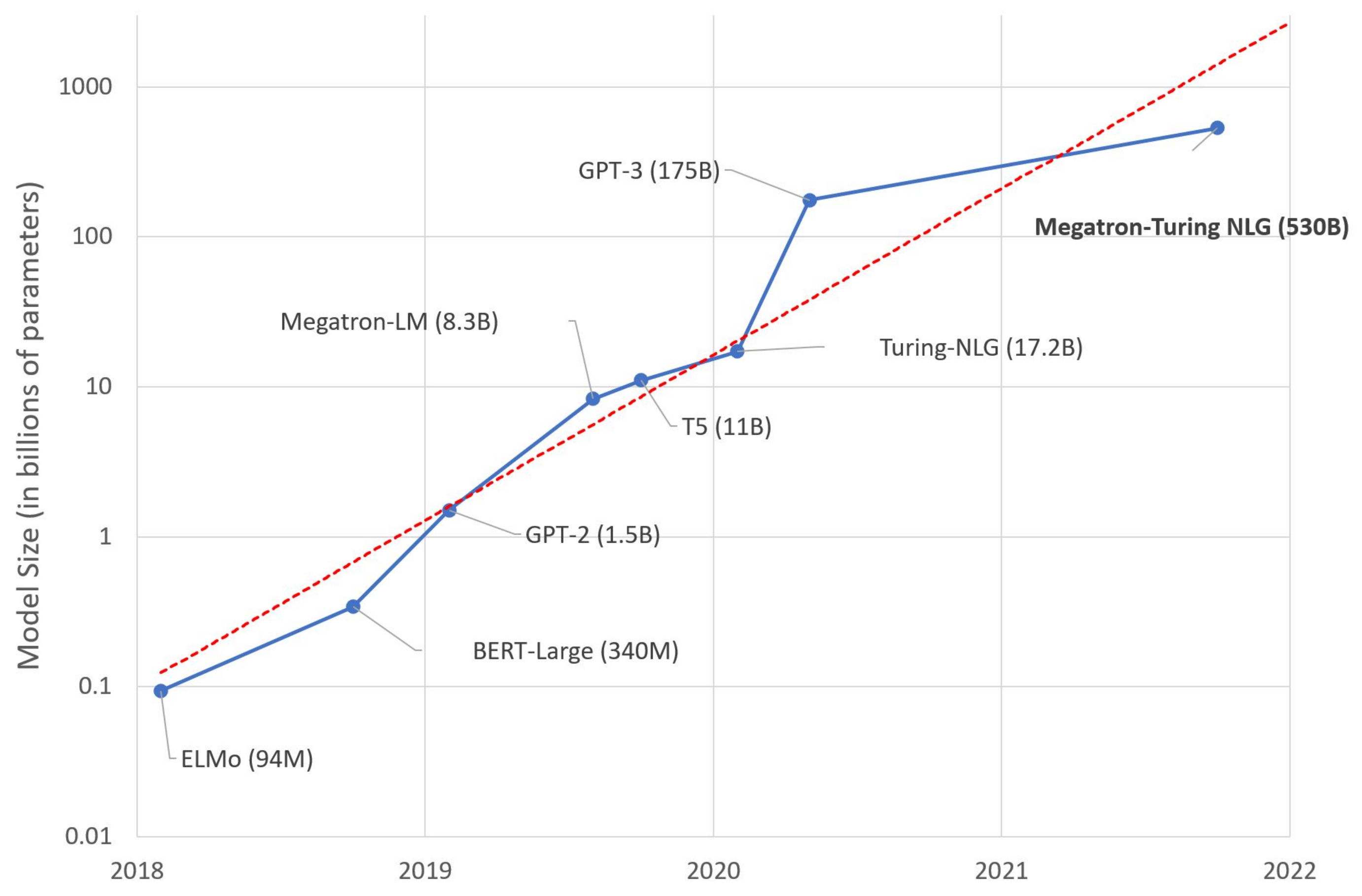
Samuel Gehman[◇] **Suchin Gururangan**[◇†] **Maarten Sap**[◇] **Yejin Choi**[◇†] **Noah A. Smith**[◇†]
[◇]Paul G. Allen School of Computer Science & Engineering, University of Washington
[†]Allen Institute for Artificial Intelligence
Seattle, USA
{sgehman,sg01,msap,yejin,nasmith}@cs.washington.edu

Abstract

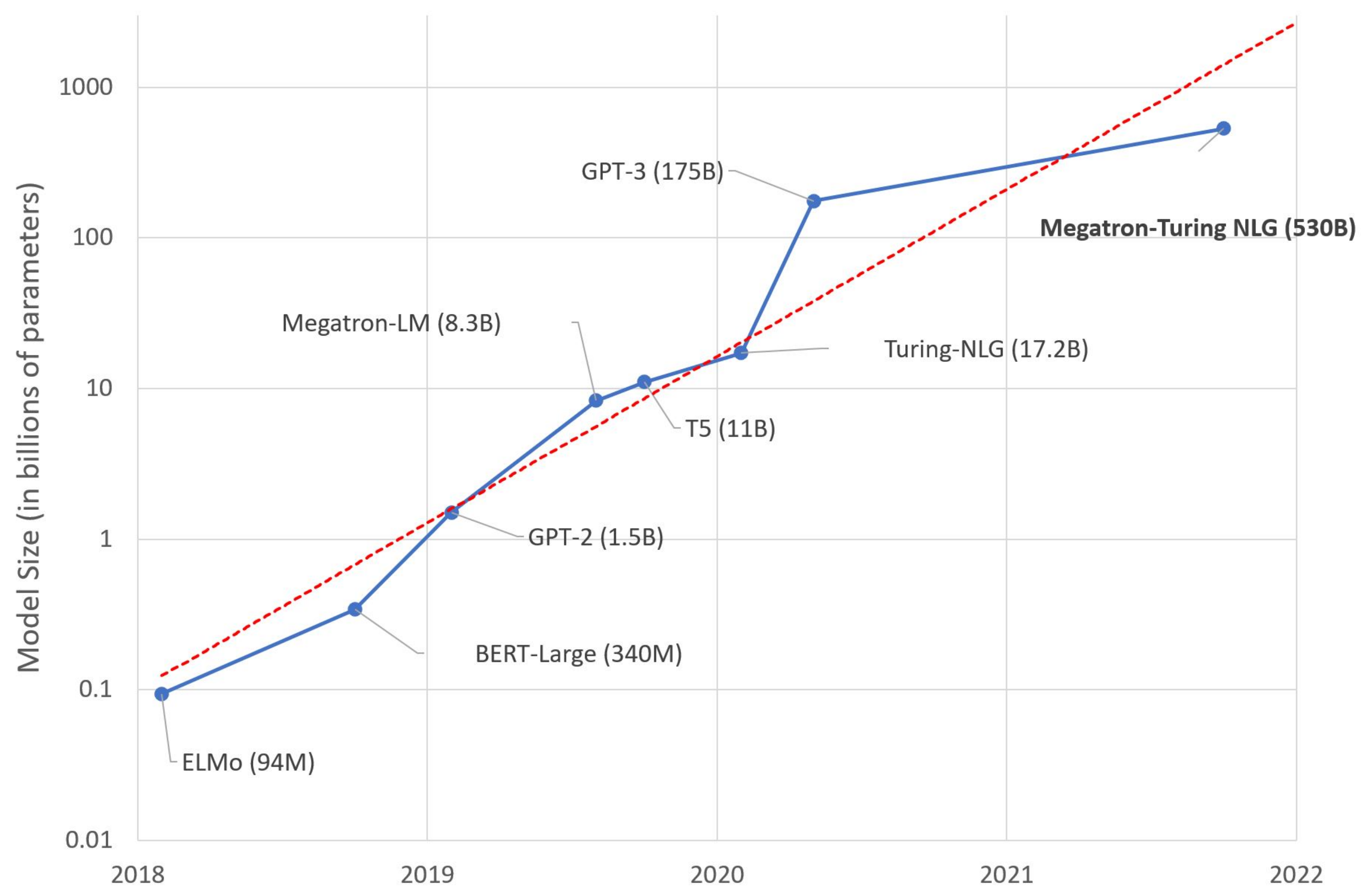
Pretrained neural language models (LMs) are prone to generating racist, sexist, or otherwise toxic language which hinders their safe deployment. We investigate the extent to which pretrained LMs can be prompted to generate toxic language, and the effectiveness of controllable text generation algorithms at preventing such toxic degeneration. We create and release REALTOXICITYPROMPTS, a dataset of 100K naturally occurring, sentence-level prompts derived from a large corpus of English web text, paired with toxicity scores from a widely-used toxicity classifier. Using REALTOXICI-

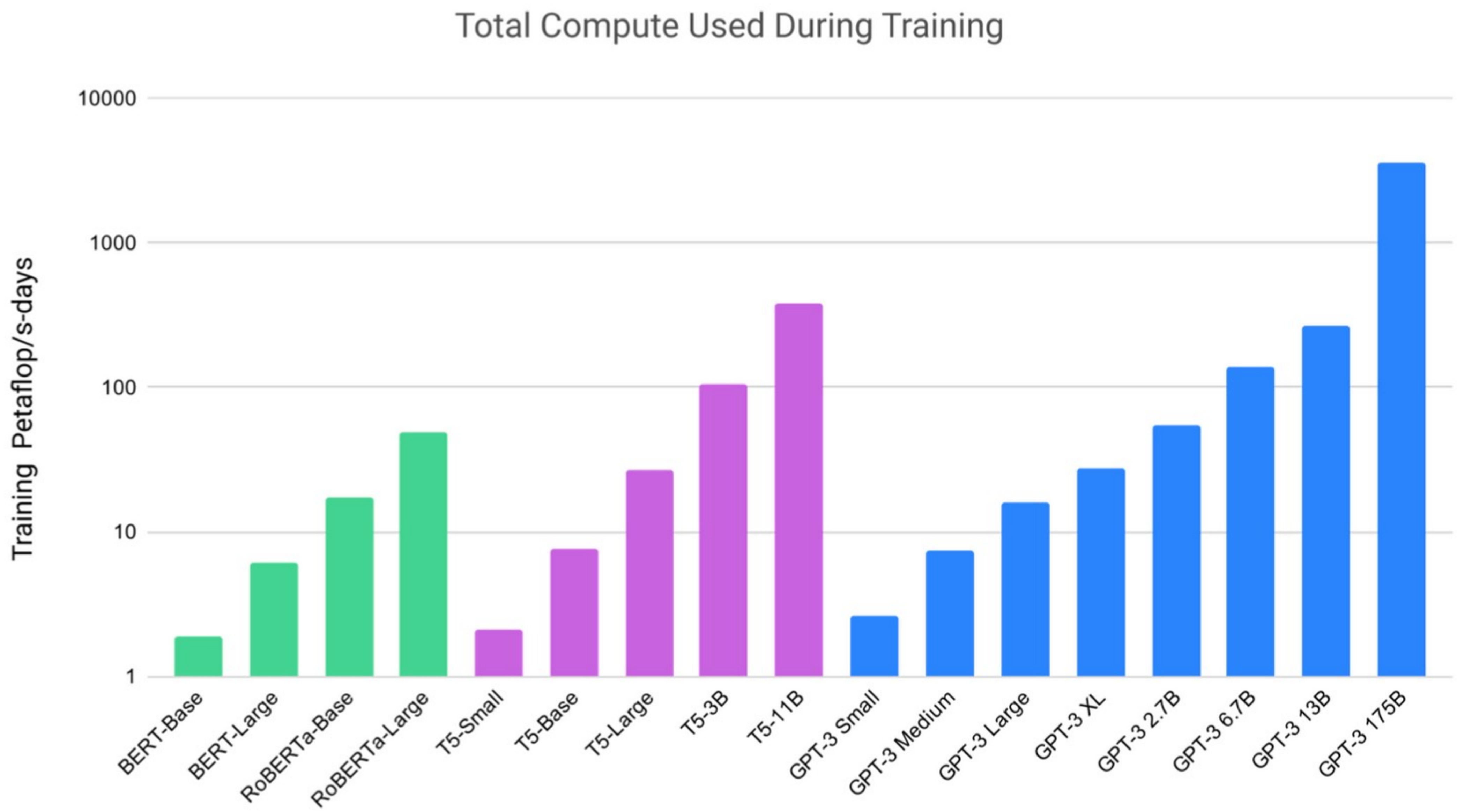


THE UGLY SIDE: COMPUTATIONAL COST

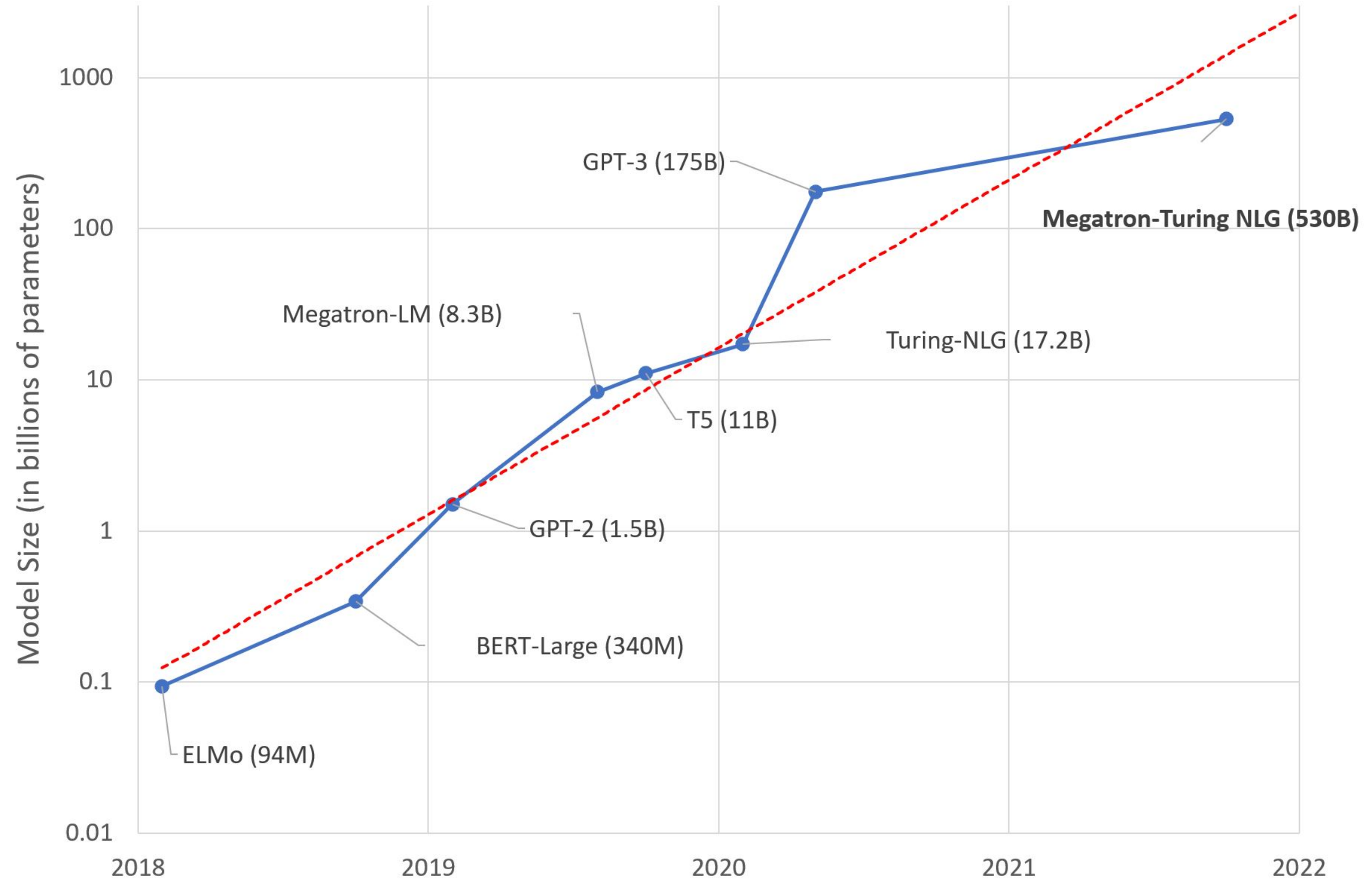


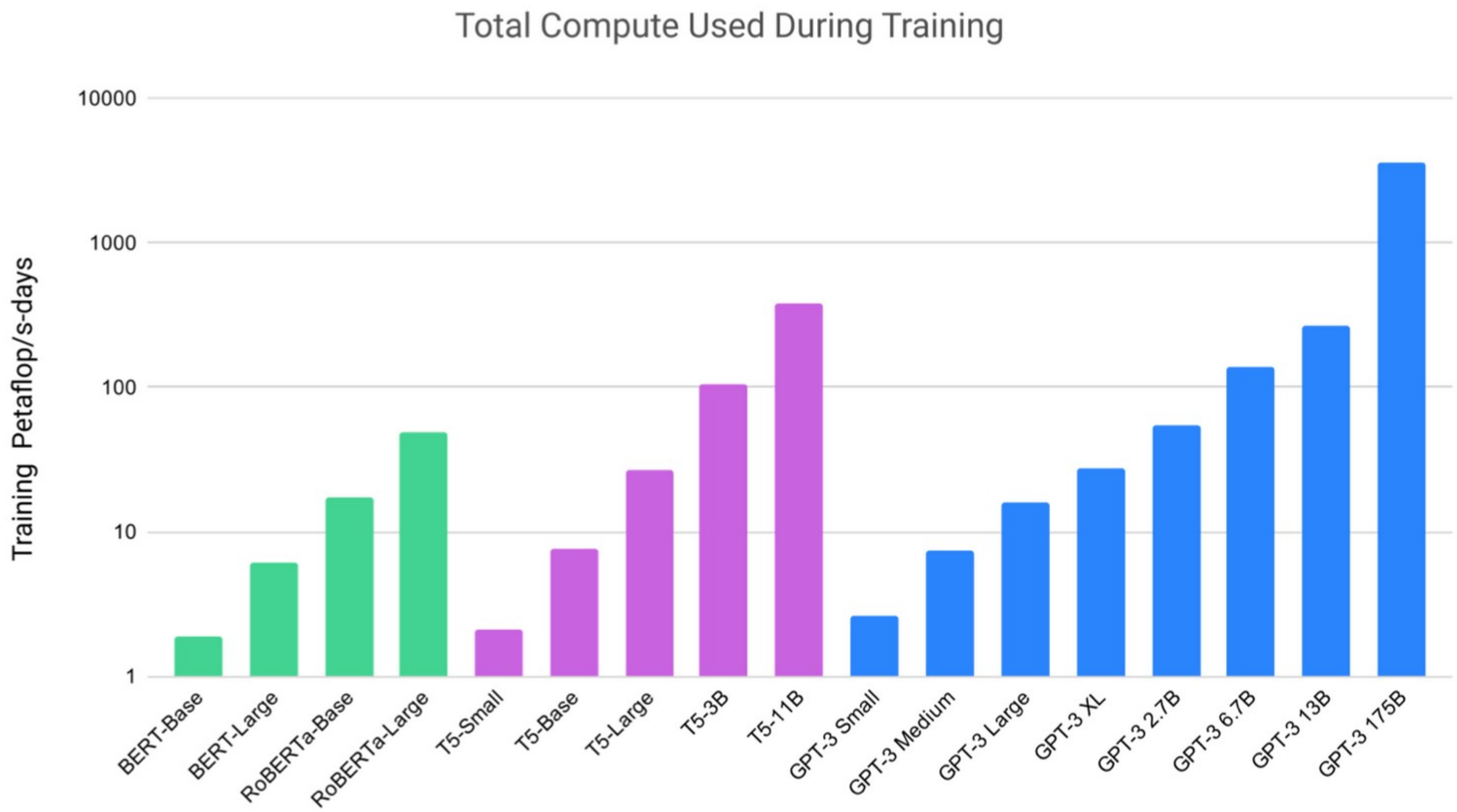
THE UGLY SIDE: COMPUTATIONAL COST





THE UGLY SIDE: COMPUTATIONAL COST





Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell **Ananya Ganesh** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, aganesh, mccallum}@cs.umass.edu

Abstract

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large networks trained on abundant data. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements depend on the availability of exceptionally large computational resources that necessitate similarly substantial energy consumption. As a result these models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

THE UGLY SIDE: AMPLIFYING DATA BIASES

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], Kai-Wei Chang[2], James Zou[2], Venkatesh Saligrama[1,2], Adam Kalai[2]
[1]Boston University, 8 Saint Mary's Street, Boston, MA
[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

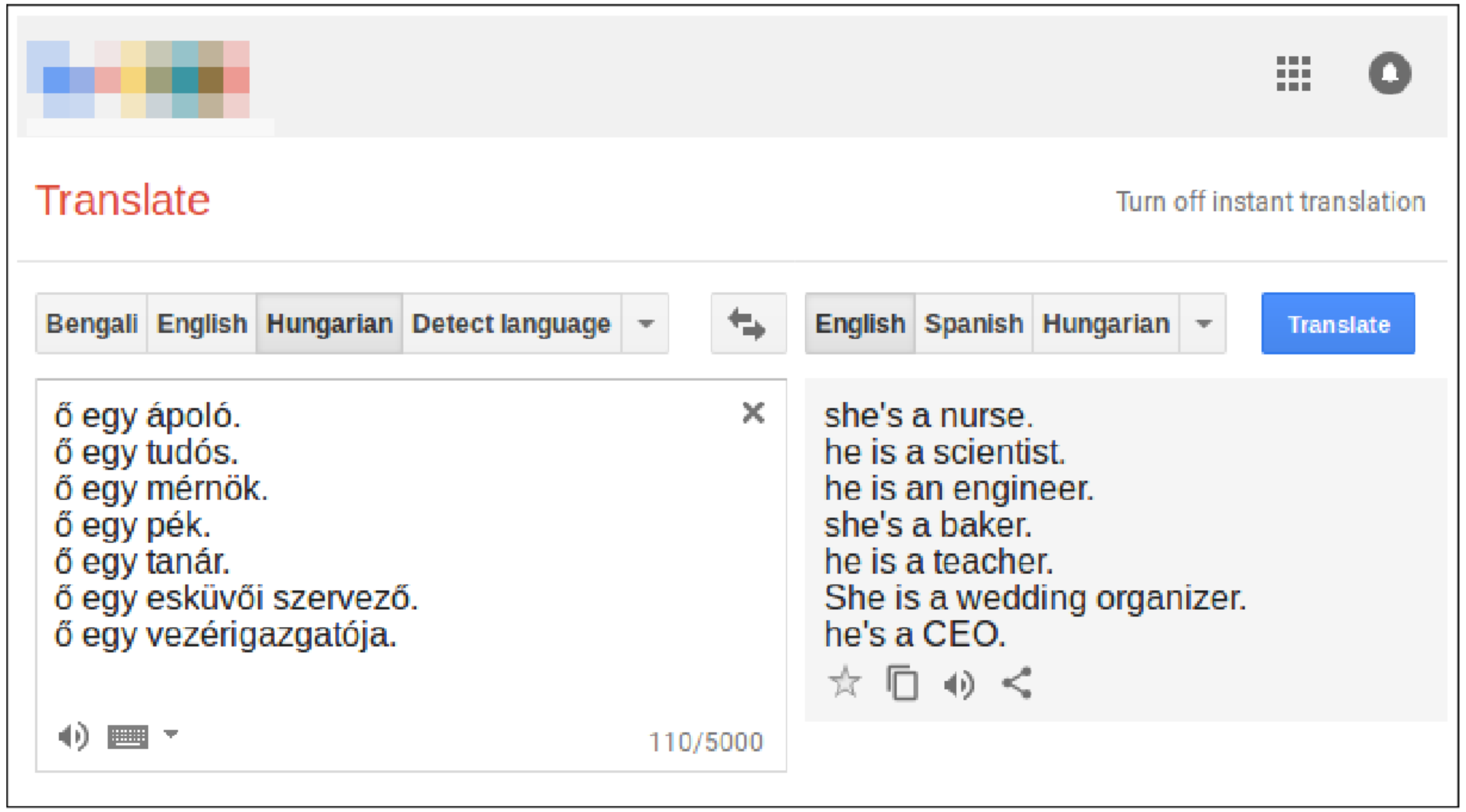
$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}.$$

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss



Source: Prates el al. 2018

KEY IDEAS WE'VE SEEN TODAY

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation

- Model sequential data with **recurrent neural networks**: challenges and solutions

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation

- Model sequential data with **recurrent neural networks**: challenges and solutions

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation
- Model sequential data with **recurrent neural networks**: challenges and solutions
- **Language models:** the backbone of most modern NLP systems

KEY IDEAS WE'VE SEEN TODAY

- **Continuous** (rather than **discrete**) representations: better for computation

- Model sequential data with **recurrent neural networks**: challenges and solutions

- **Language models:** the backbone of most modern NLP systems

KEY IDEAS WE'VE SEEN TODAY

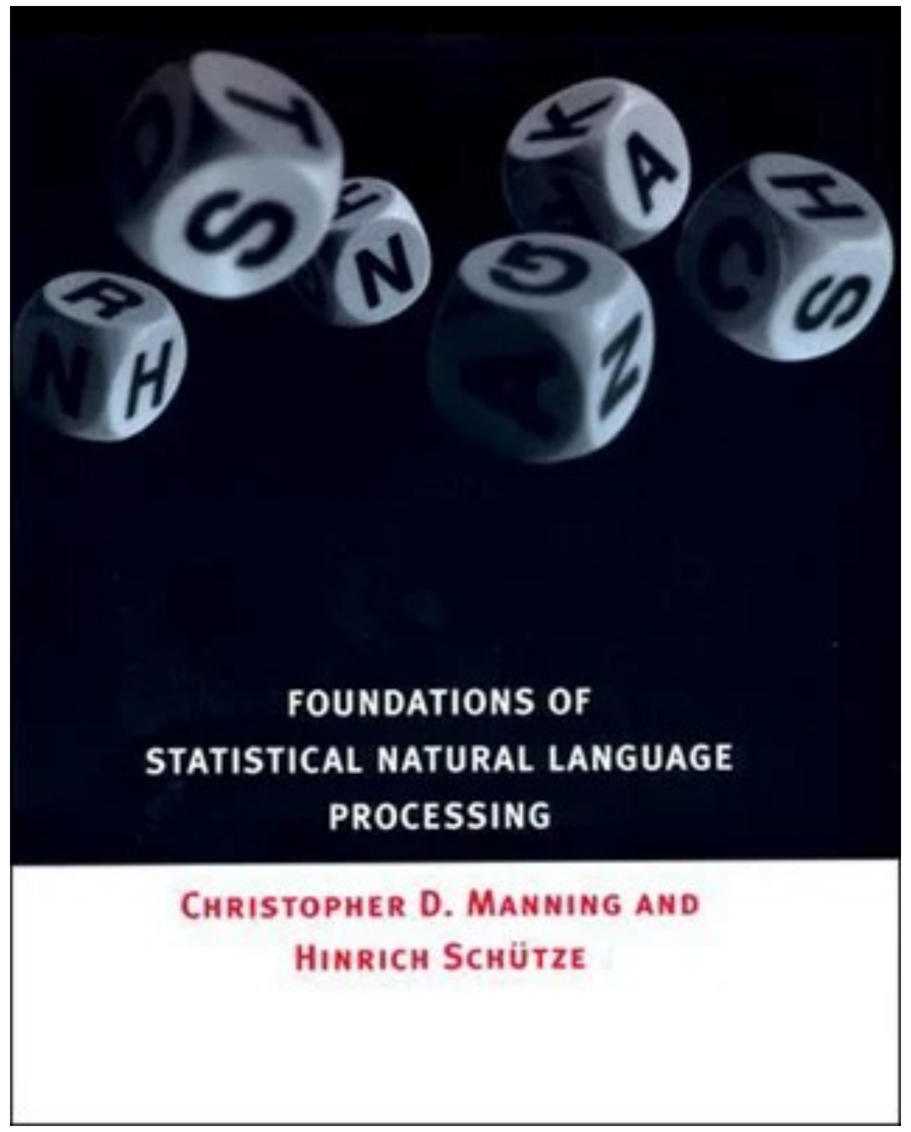
- **Continuous** (rather than **discrete**) representations: better for computation

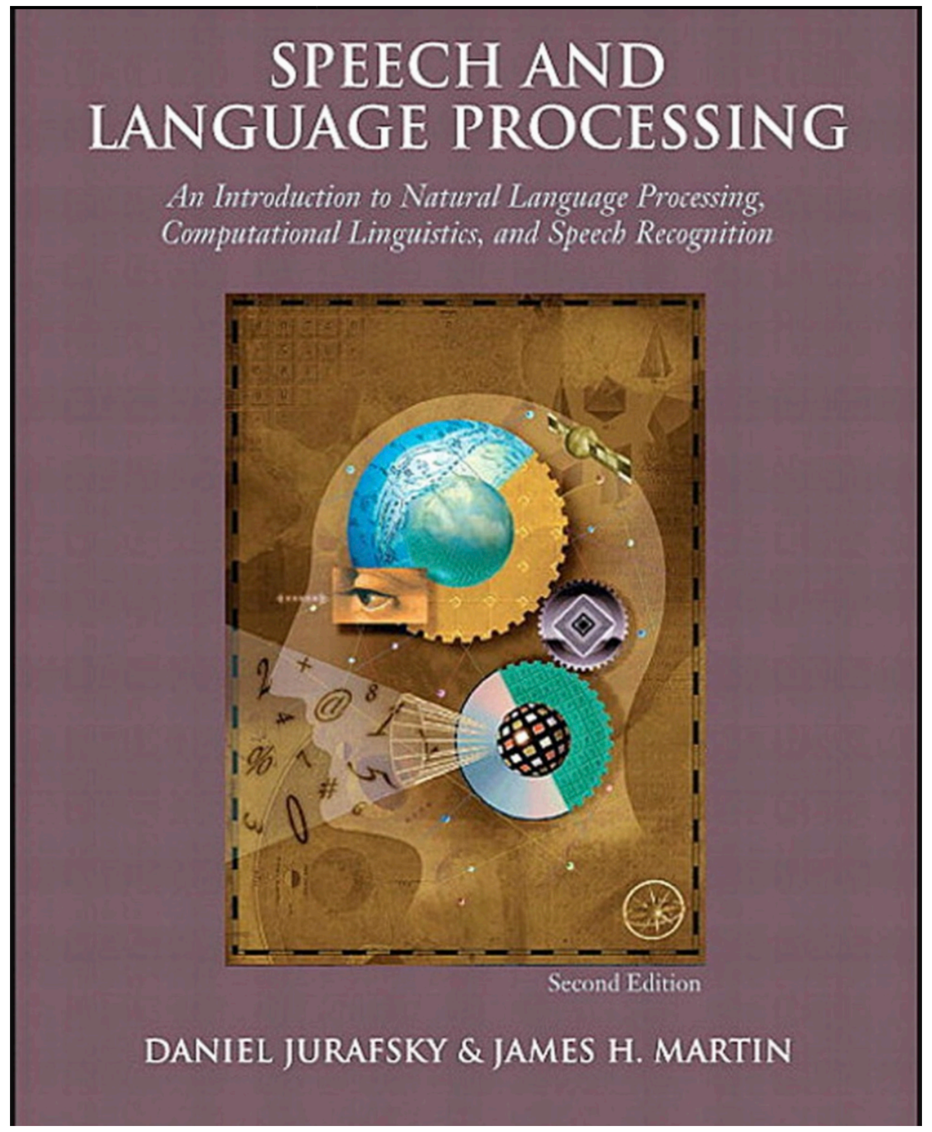
- Model sequential data with **recurrent neural networks**: challenges and solutions

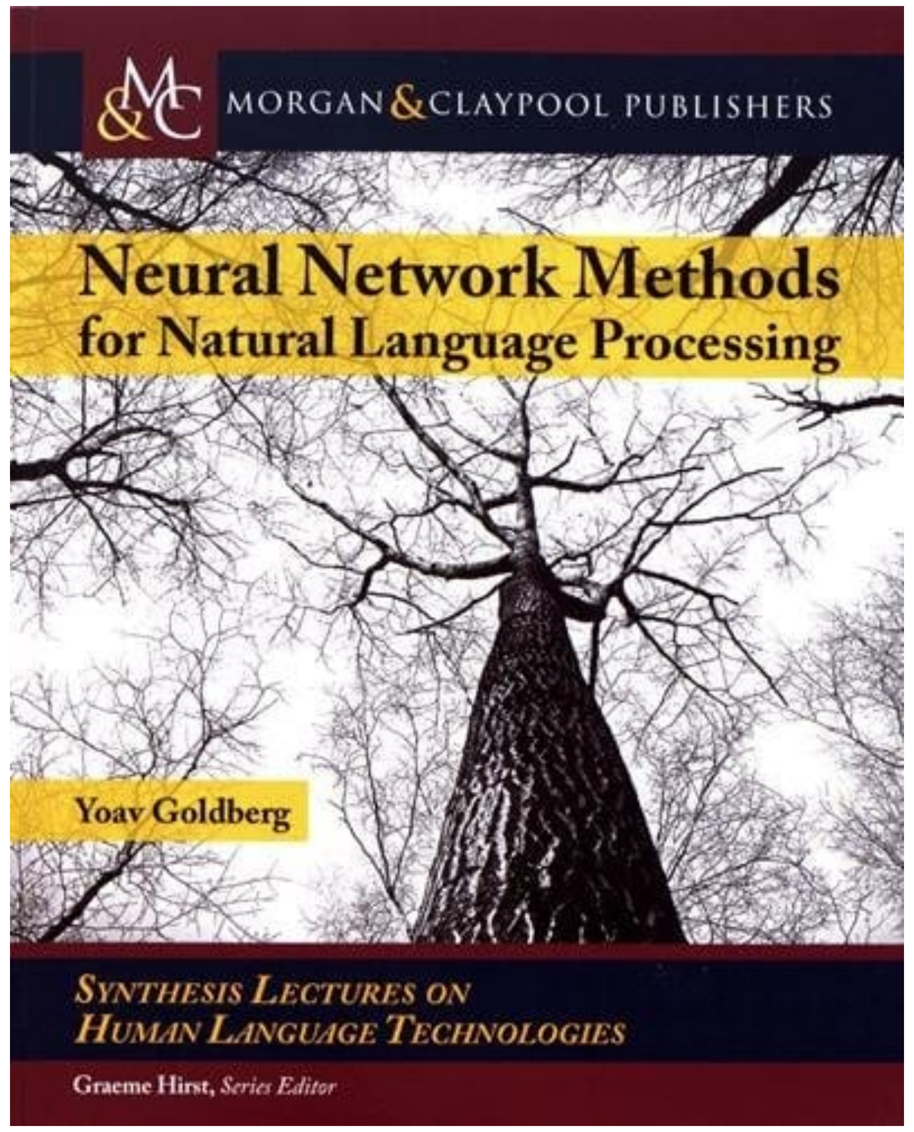
- **Language models**: the backbone of most modern NLP systems

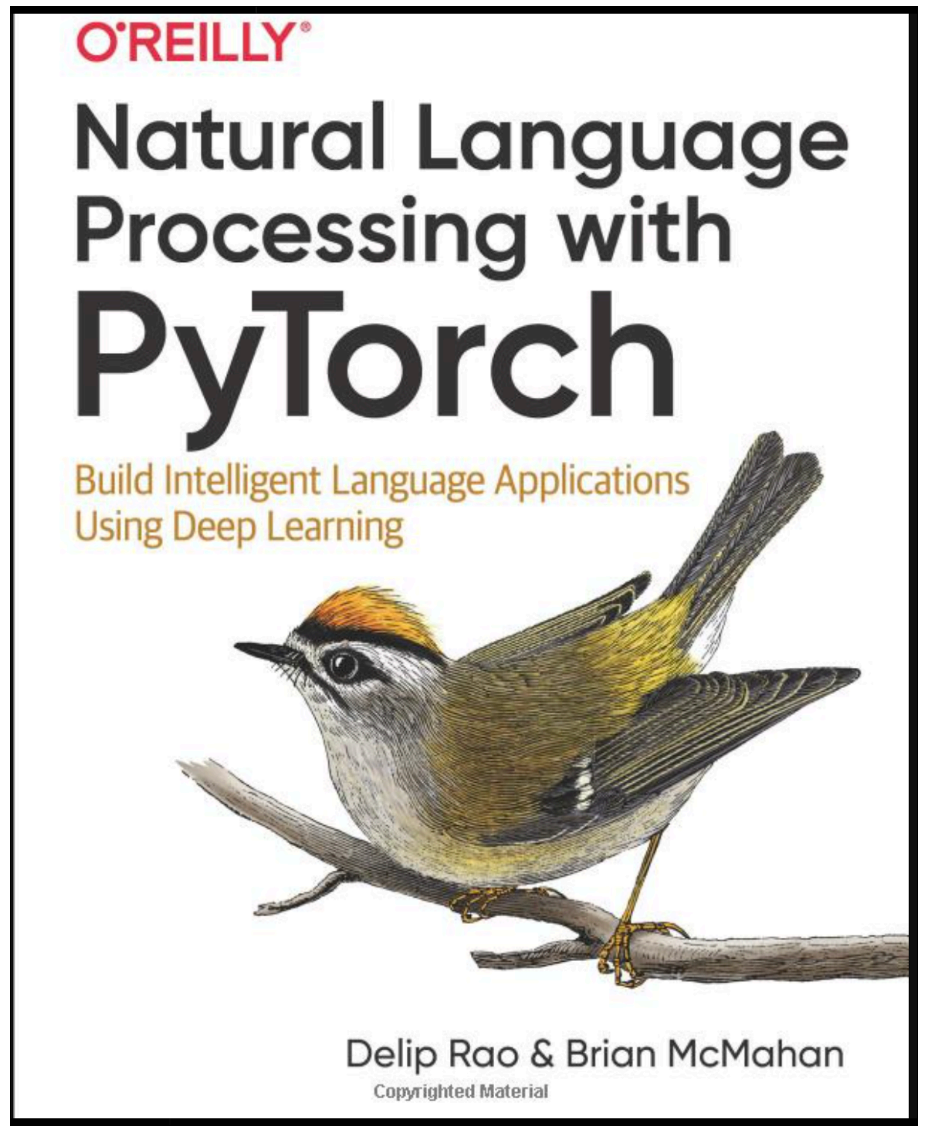
- NLP is not (just) a research field anymore, it's a commodity: **high societal impact**

RECOMMENDED READINGS

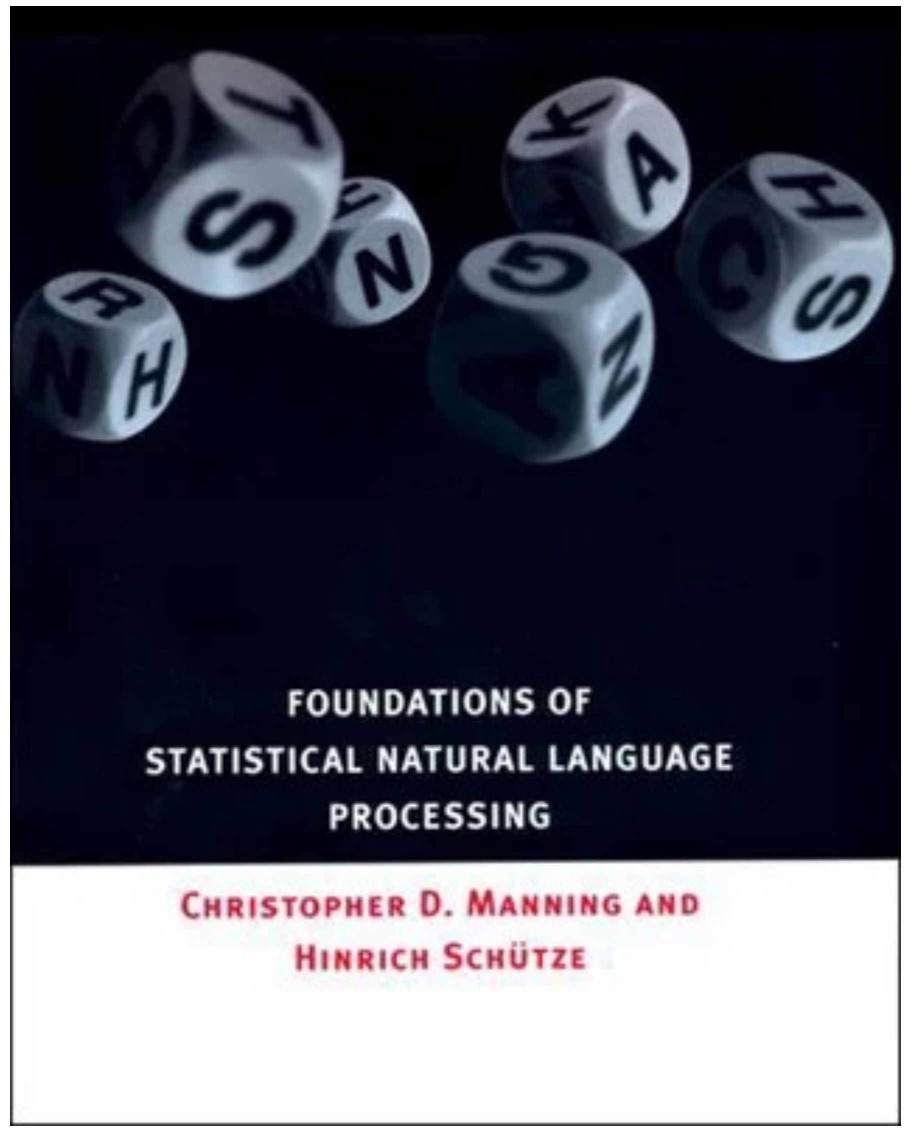


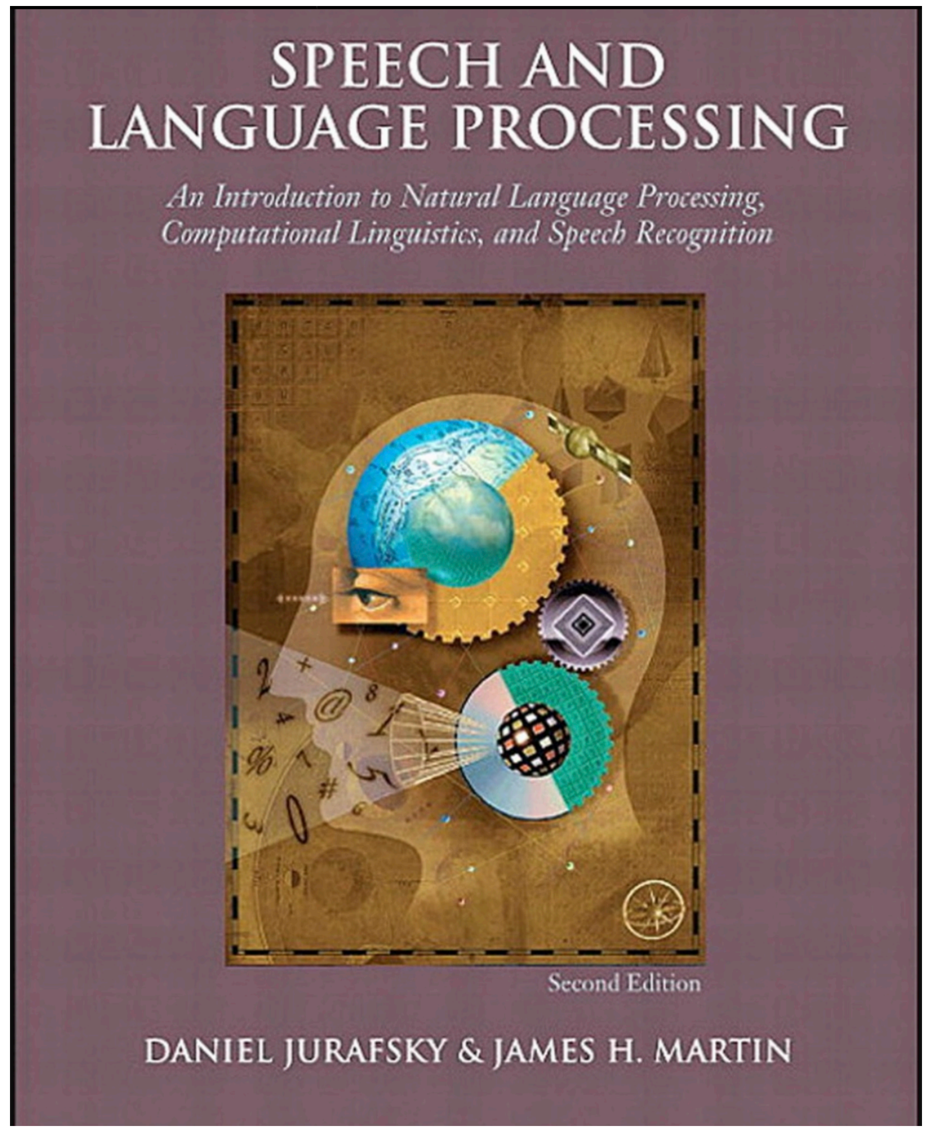


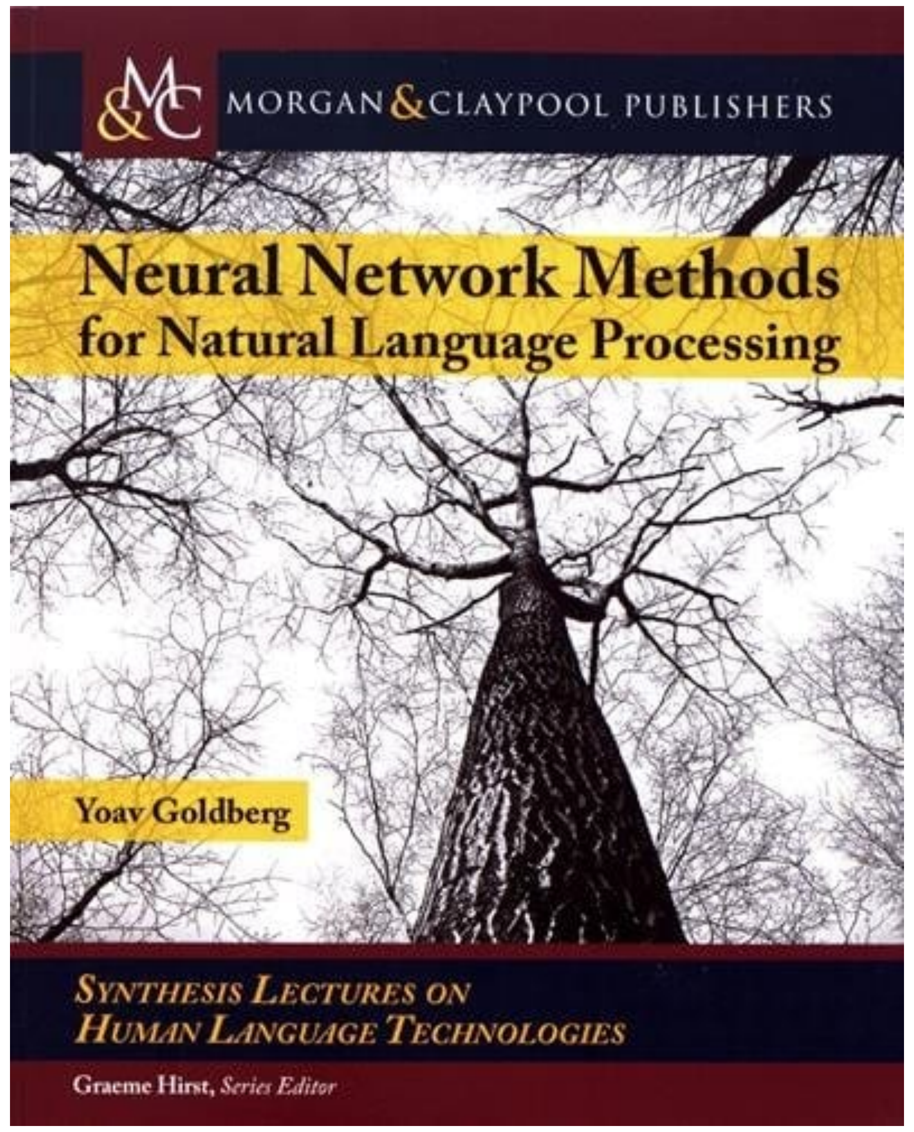


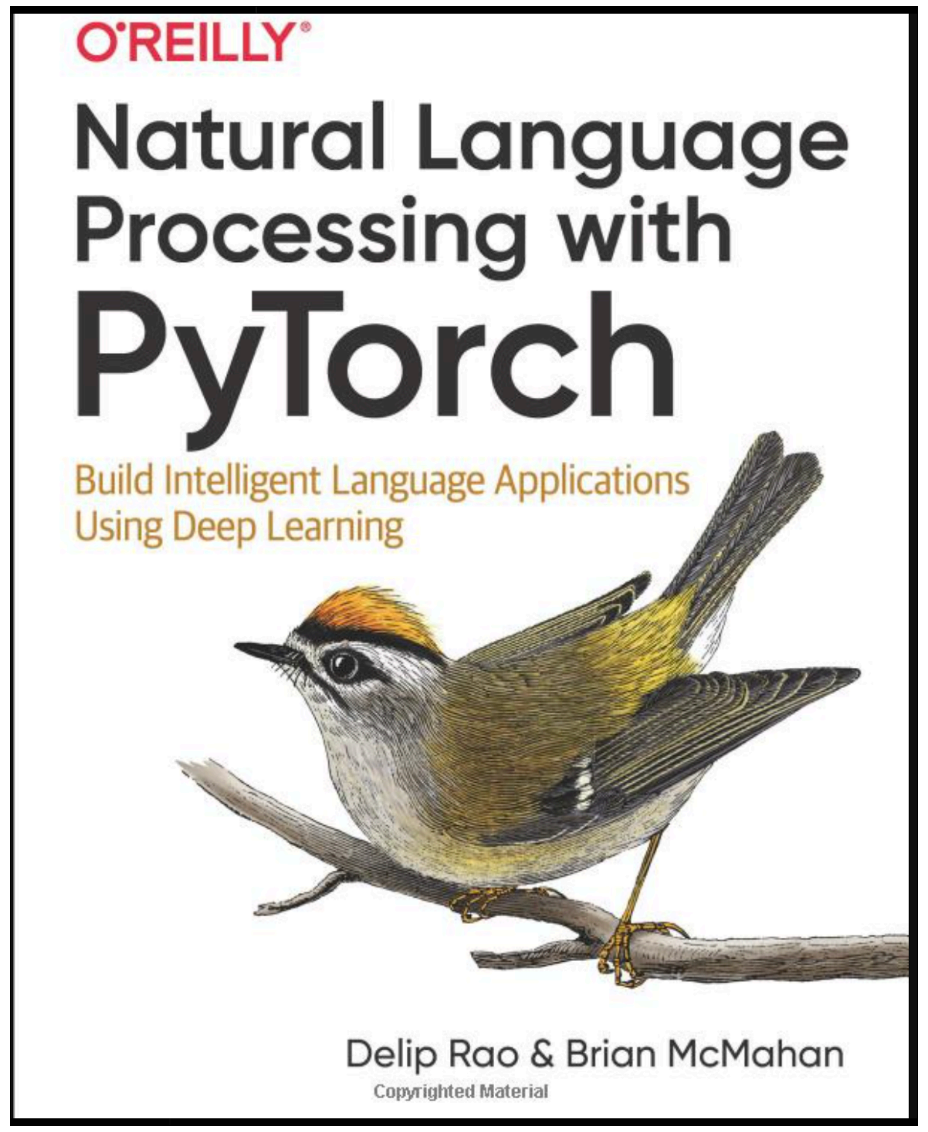


RECOMMENDED READINGS



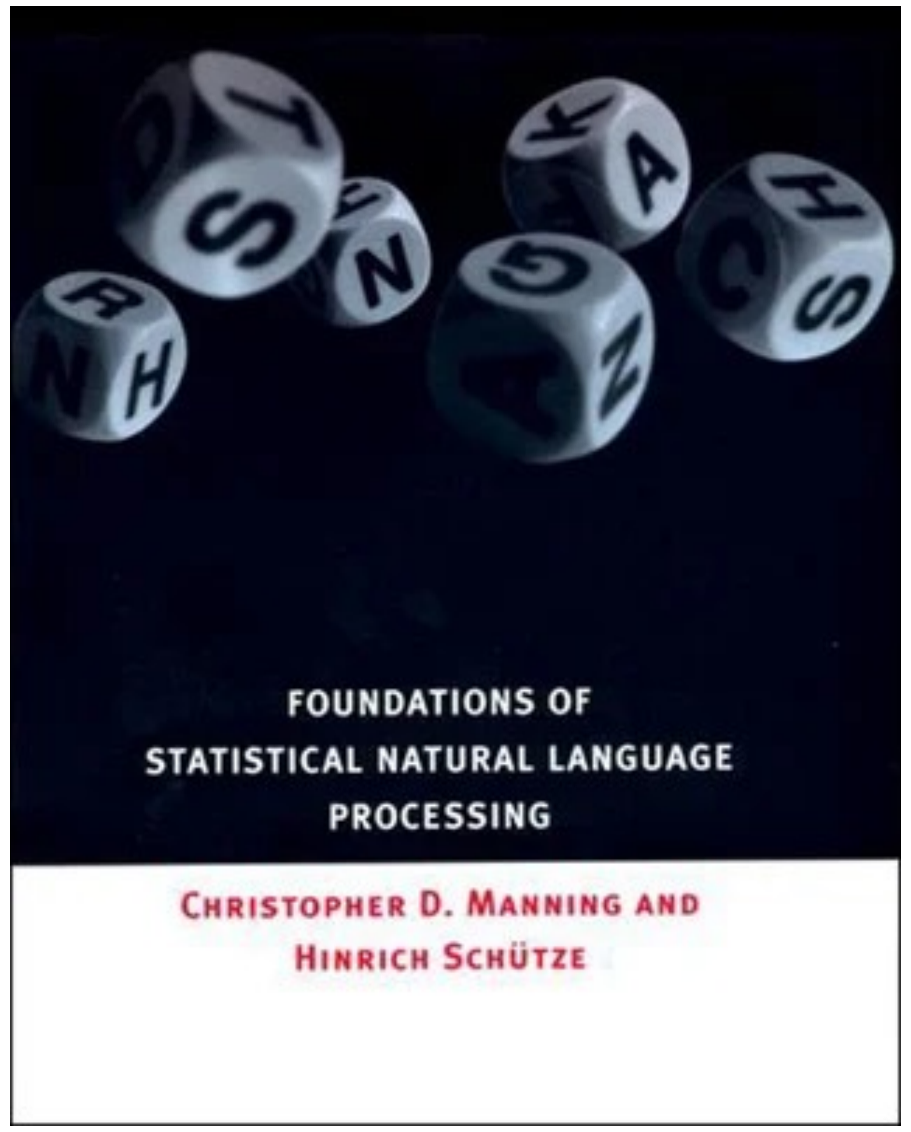


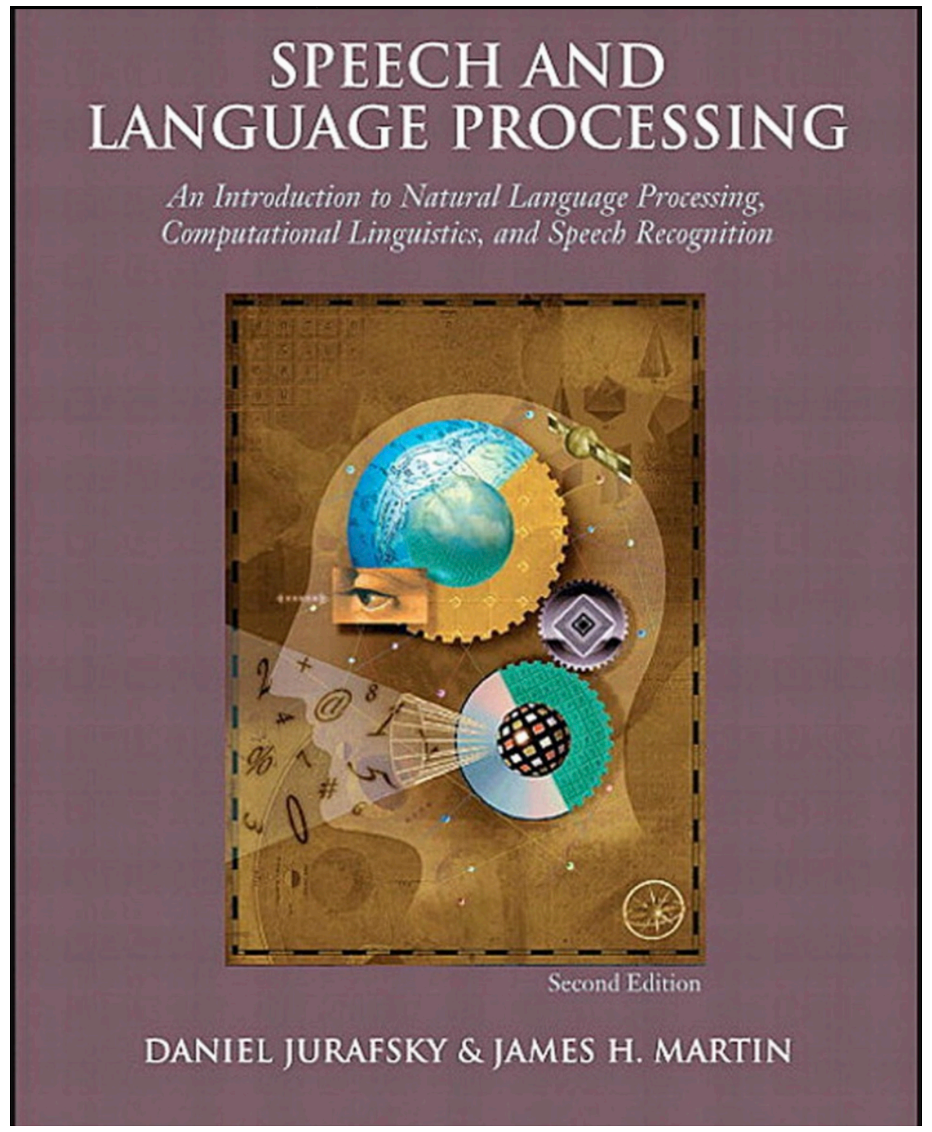


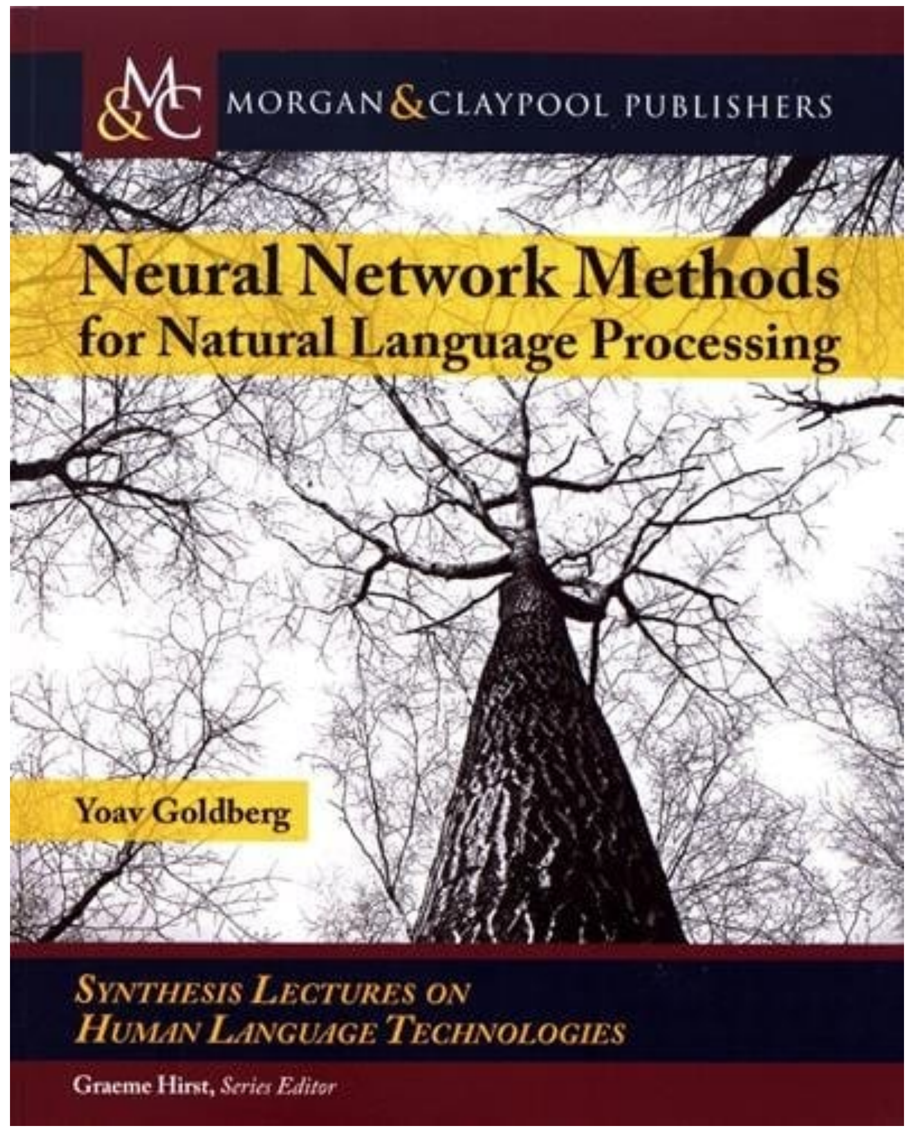


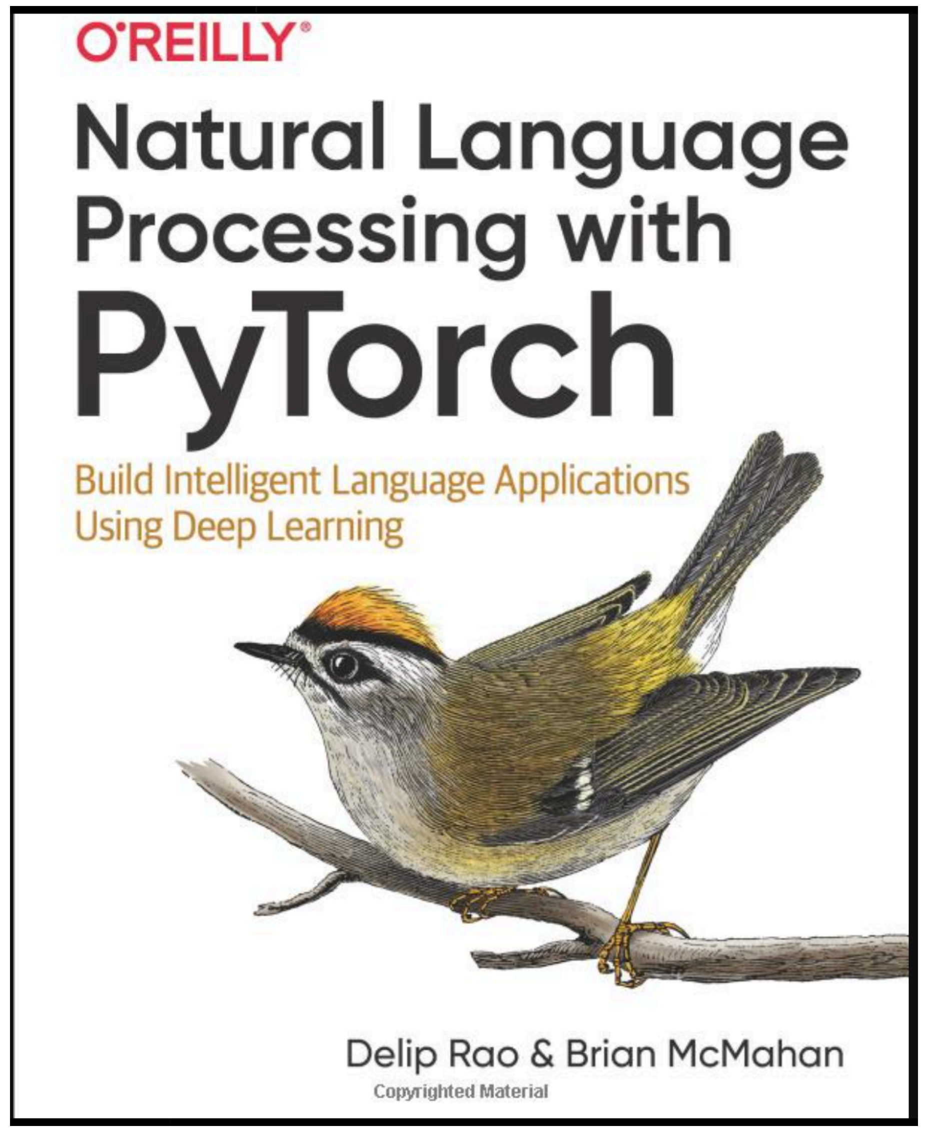
Classic text-books; great reference for foundations and pre-neural NLP

RECOMMENDED READINGS





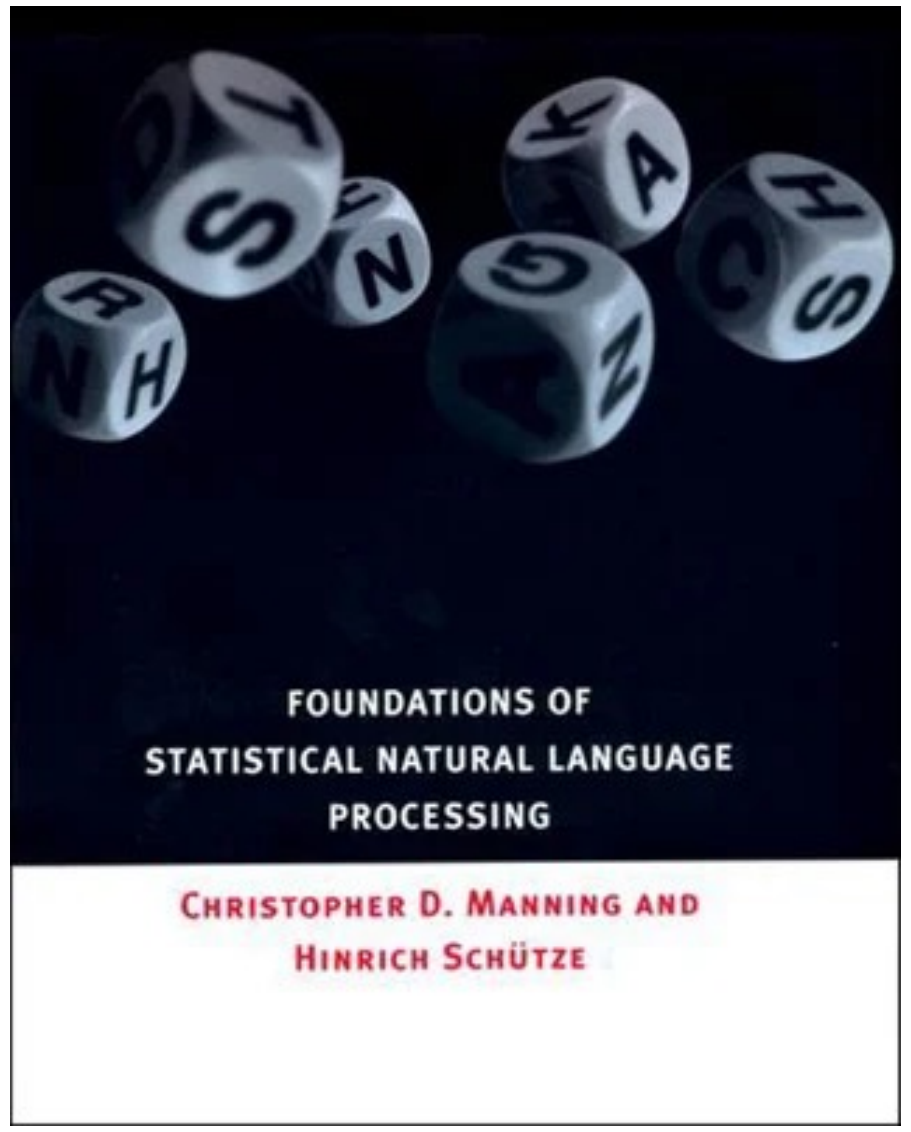


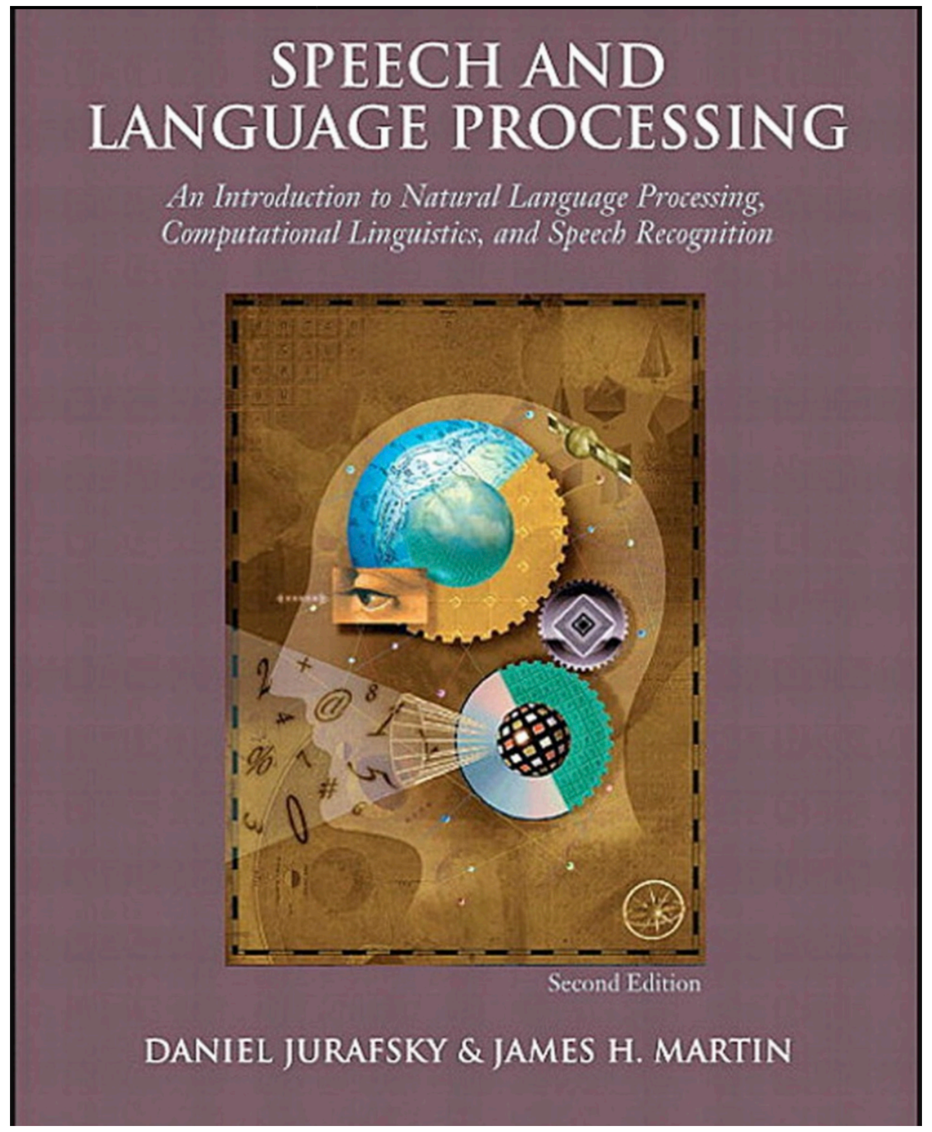


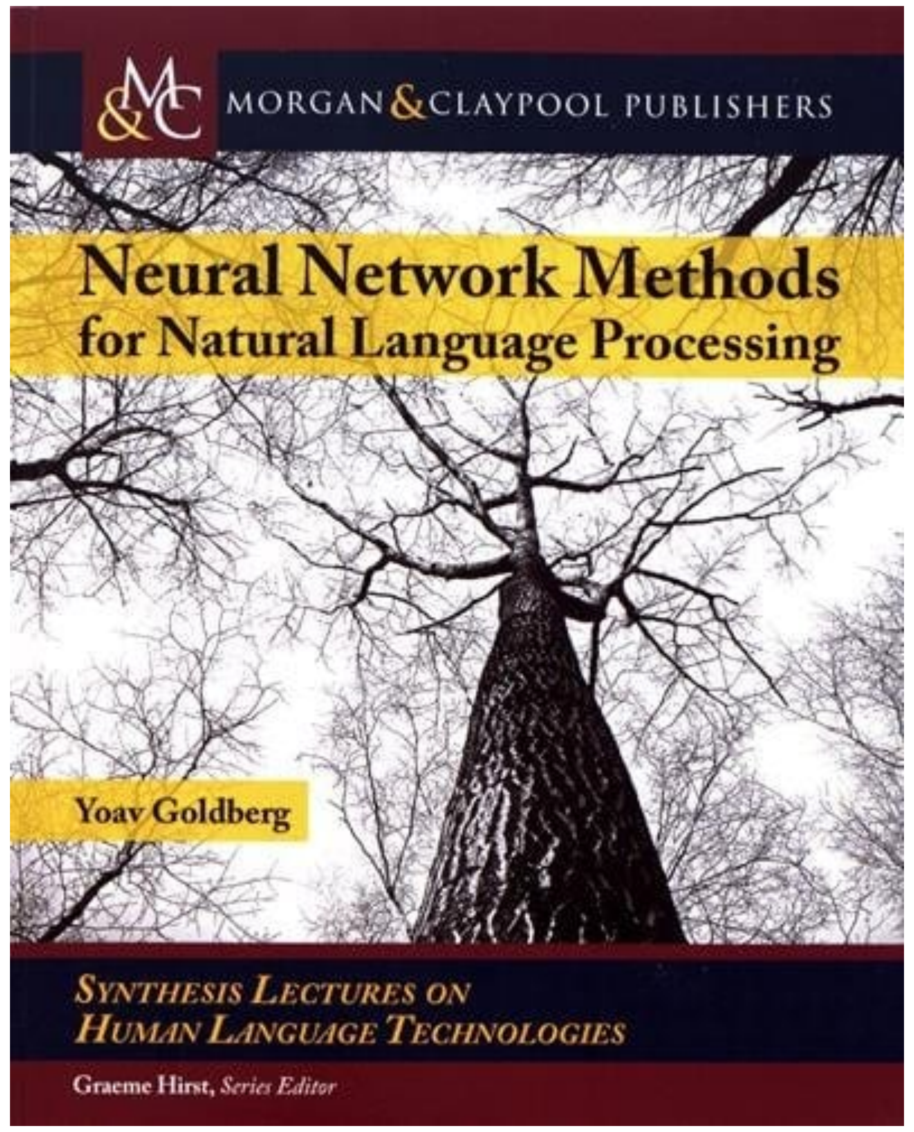
Classic text-books; great reference for foundations and pre-neural NLP

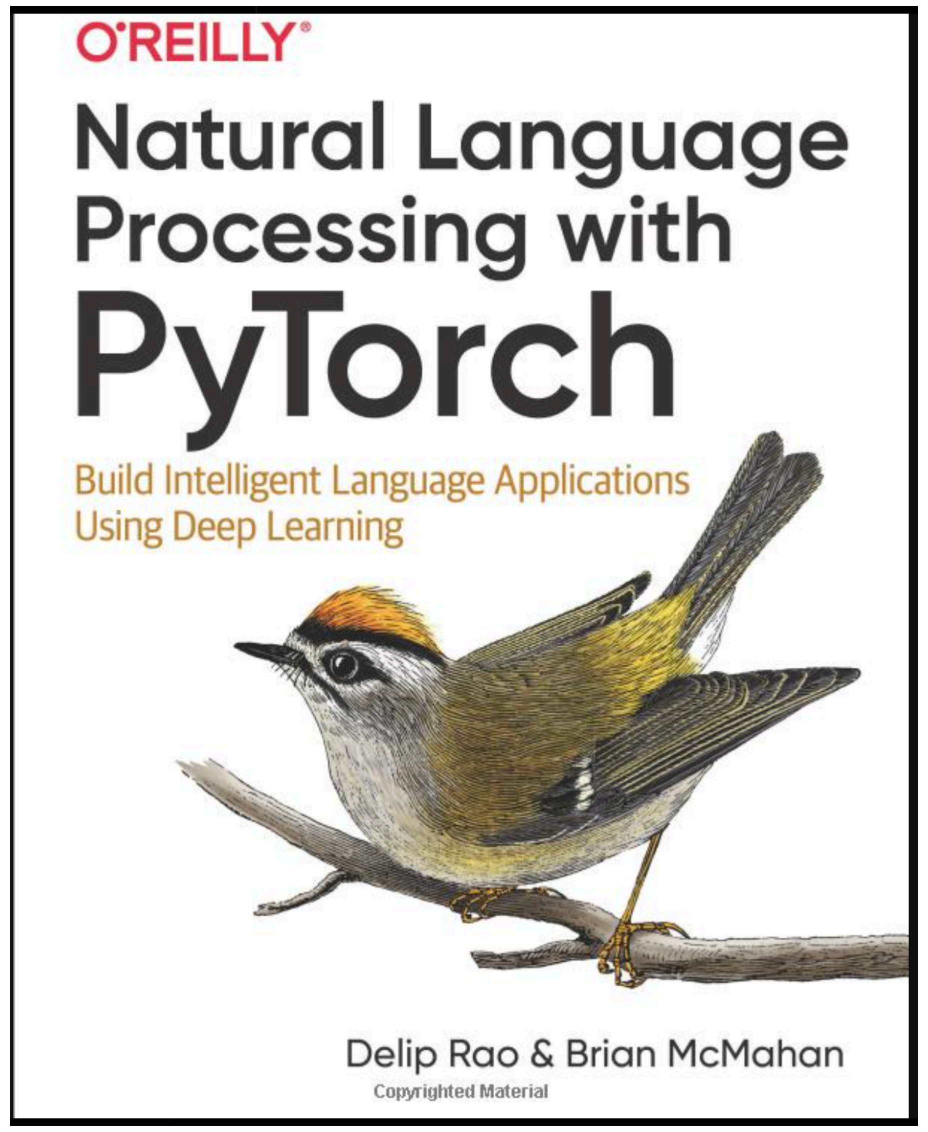
self-contained intro to neural NLP

RECOMMENDED READINGS









Classic text-books; great reference for foundations and pre-neural NLP

self-contained intro to neural NLP

Hands-on!

BONUS:

FURTHER TOPICS

"EVERY TIME I FIRE A LINGUIST, THE PERFORMANCE OF THE SPEECH RECOGNIZER GOES UP"

Fred Jelinek,

NLP + ASR pioneer

FURTHER TOPICS:

INTERPRETABILITY IN NLP

INTERPRETABILITY IN NLP

Modern NLP models have [mi|bi|tri]-llions of parameters – essentially black boxes!

How can we interpret their predictions?

Via Attention?

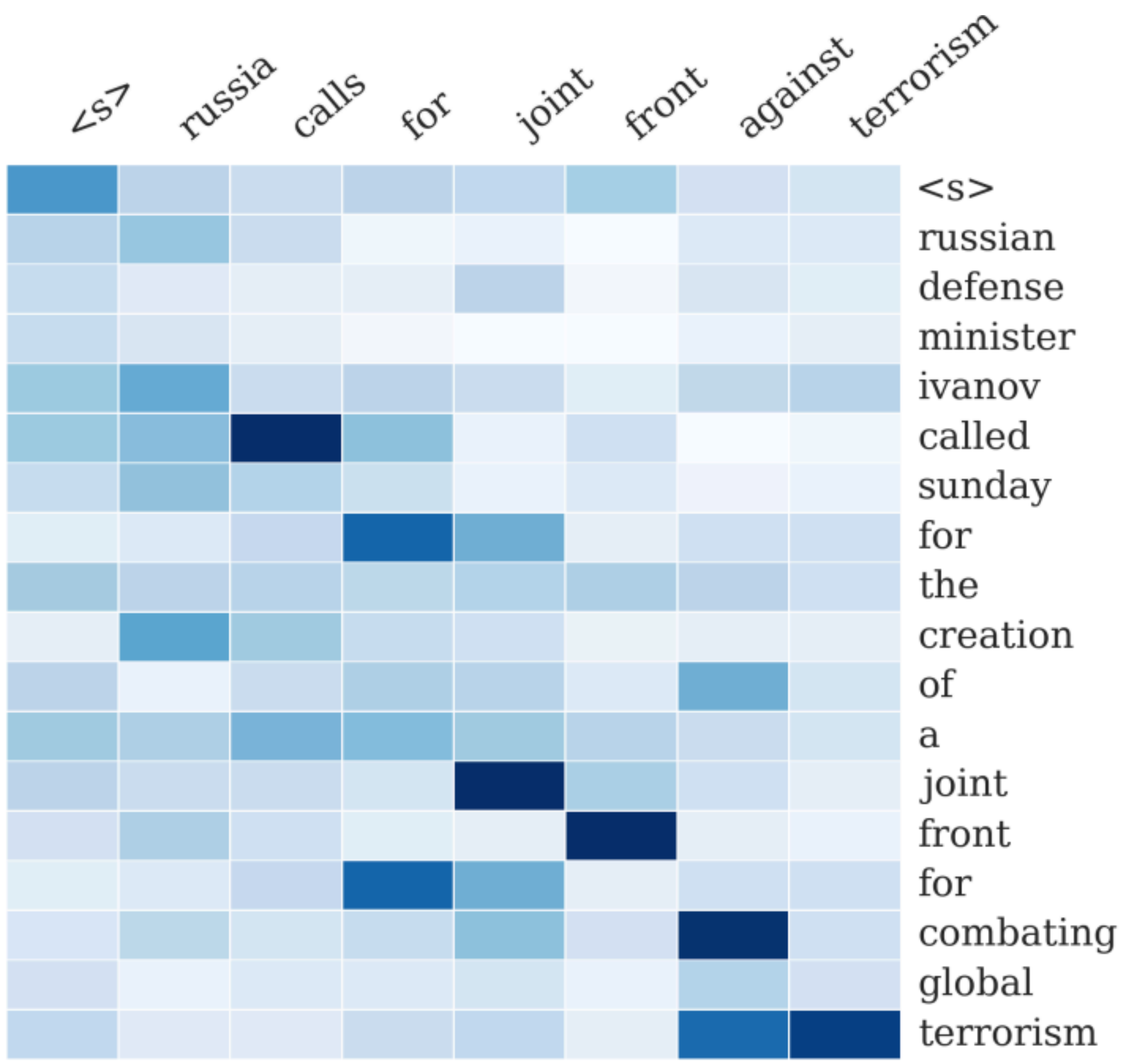
Attention is dense! Not very interpretable, especially for long inputs/outputs

INTERPRETABILITY IN NLP

Modern NLP models have [mi|bi|tri]-llions of parameters – essentially black boxes!

How can we interpret their predictions?

Via Attention?

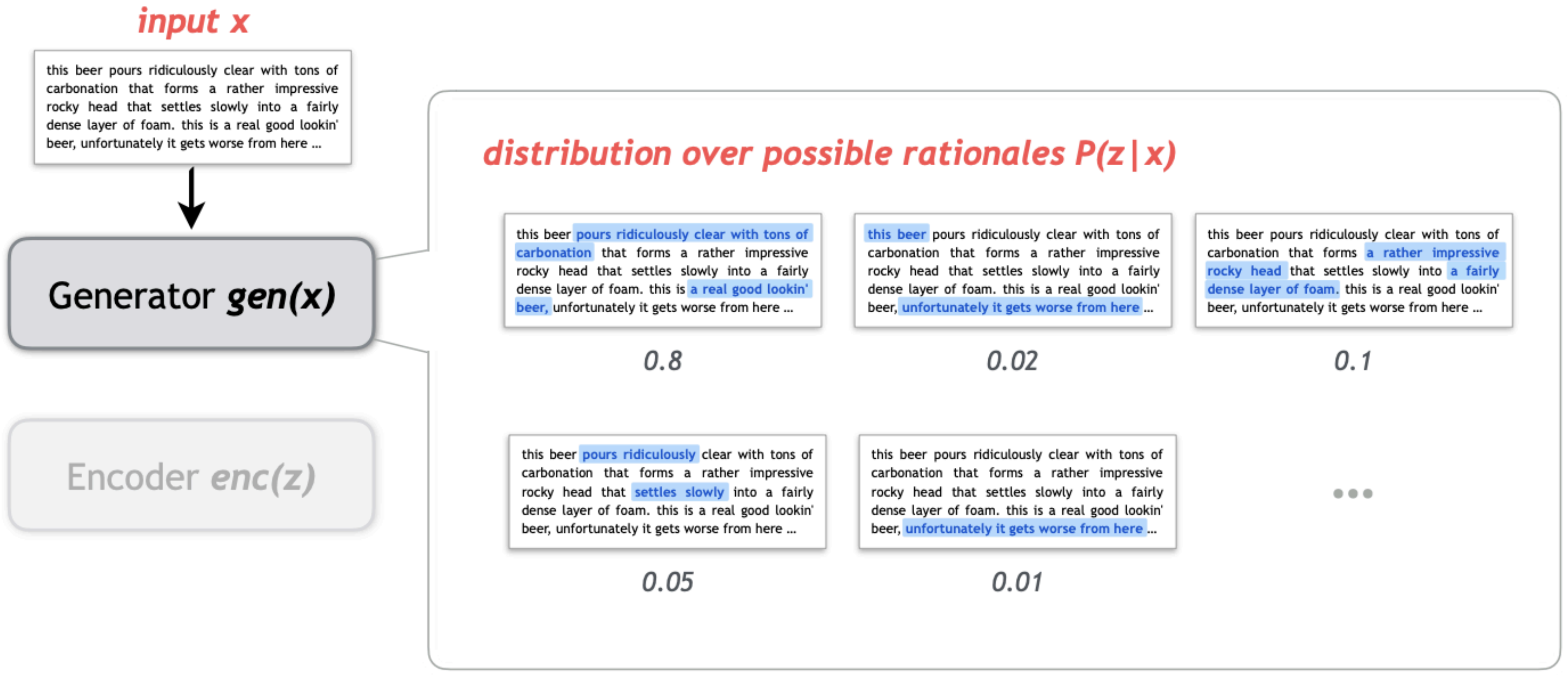


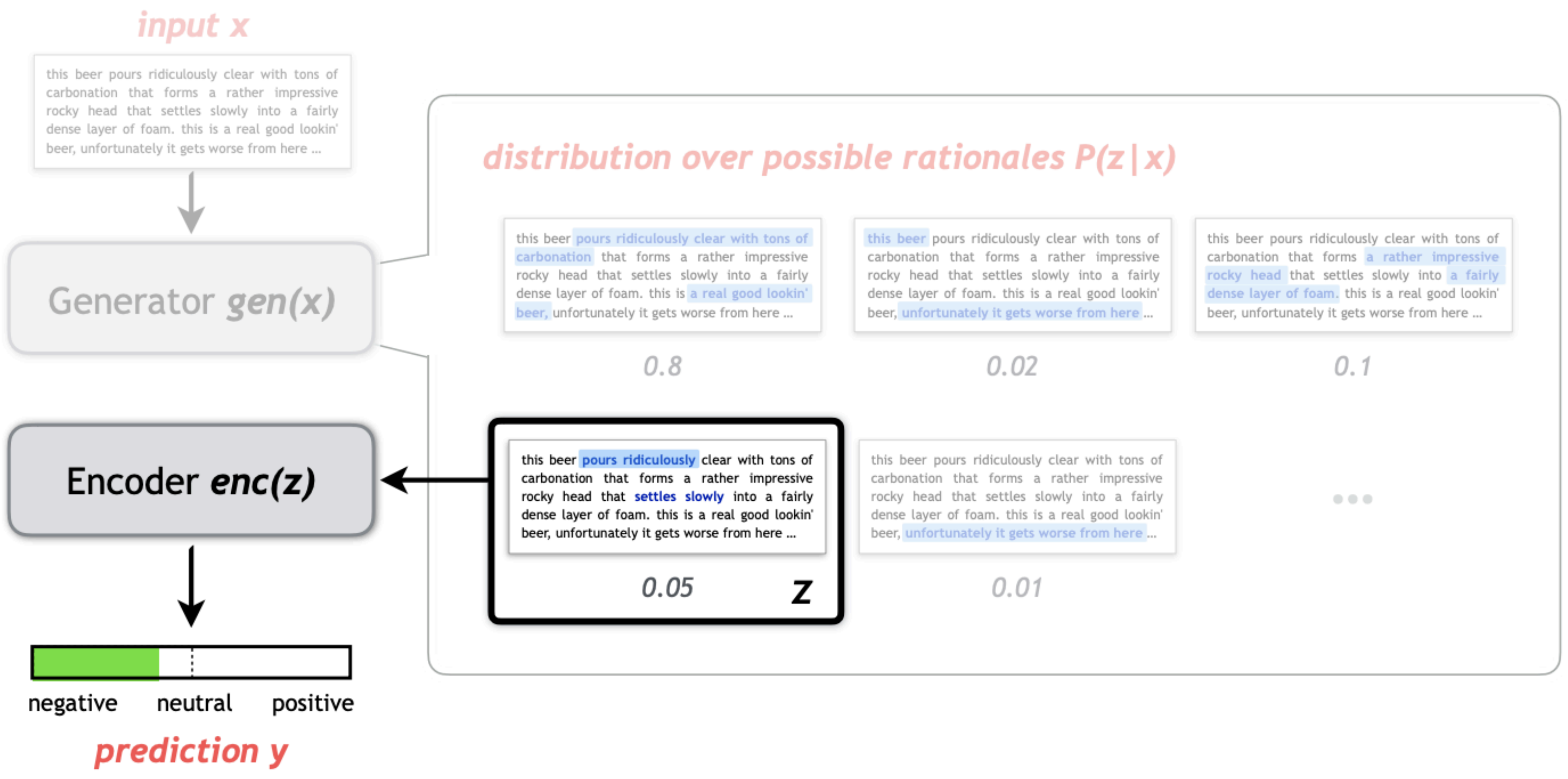
[Rush et al., 2015]

Attention is dense! Not very interpretable, especially for long inputs/outputs

RATIONALIZING NEURAL PREDICTIONS
[Lei et al. 2016]

Force model to use a small subset of the original input - interpretation as cooperative game





generator specifies the distribution of rationales

encoder makes prediction given rationale

RATIONALIZING NEURAL PREDICTIONS
[Lei et al. 2016]

Force model to use a small subset of the original input - interpretation as cooperative game

Task: predict ratings and rationales for each aspect

this beer pours ridiculously clear with tons of carbonation that forms a rather impressive rocky head that settles slowly into a fairly dense layer of foam. this is a real good lookin' beer, unfortunately it gets worse from here … first, the aroma is kind of bubblegum-like and grainy. next, the taste is sweet and grainy with an unpleasant bitterness in the finish. … … overall, the fat weasel is good for a fairly cheap buzz, but only if you like your beer grainy and bitter .

Ratings

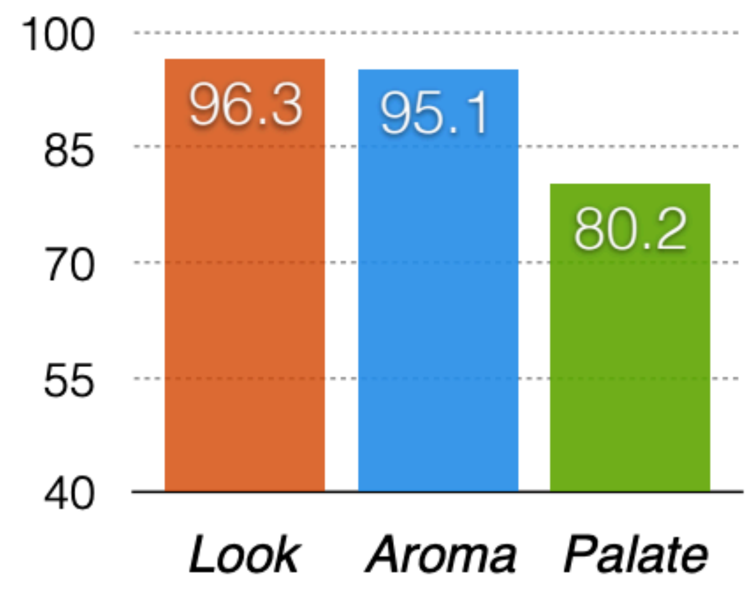
Look: 5 stars

Aroma: 2 stars

Examples and precisions of rationales

a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a generous head that sustained life throughout . nothing out of the ordinary here , but a good brew still . body was kind of heavy , but not thick . the hop smell was excellent and enticing . very drinkable

poured into a snifter . produces a small coffee head that reduces quickly . black as night . pretty typical imp . roasted malts hit on the nose . a little sweet chocolate follows . big toasty character on the taste . in between i 'm getting plenty of dark chocolate and some bitter espresso . it finishes with hop bitterness . nice smooth mouthfeel with perfect carbonation for the style . overall a nice stout i would love to have again , maybe with some age on it .



Evaluation: Parsing Pathology Report

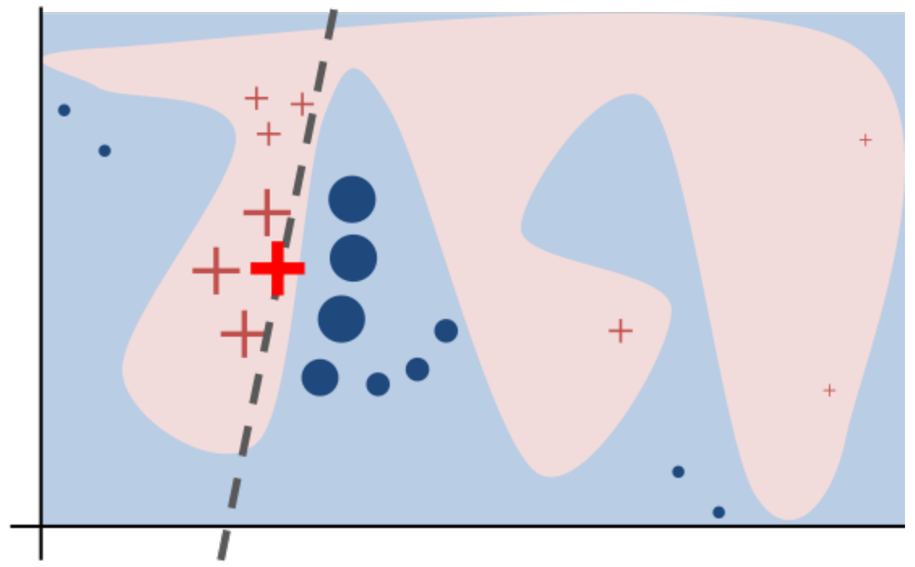
Category:		*F-score:*
IDC	*Accession Number* <unk> *Report Status* Final *Type* Surgical Pathology … *Pathology Report:* LEFT BREAST ULTRASOUND GUIDED CORE NEEDLE BIOPSIES … INVASIVE DUCTAL CARCINOMA poorly differentiated modified Bloom Richardson grade III III measuring at least 0 7cm in this limited specimen Central hyalinization is present within the tumor mass but no necrosis is noted No lymphovascular invasion is identified No in situ carcinoma is present Special studies were performed at an outside institution with the following results not reviewed ESTROGEN RECEPTOR NEGATIVE PROGESTERONE RECEPTOR NEGATIVE …	98%

INTERPRETABILITY IN NLP

- What if the model is already trained? And we have no access to its parameters etc…
- Idea (Ribeiro et al. 2016): fit a simple interpretable model around a given query, using perturbations of the input
- But this assumes input is continuous, output is a single value. Can we extend this to text data?

INTERPRETABILITY IN NLP

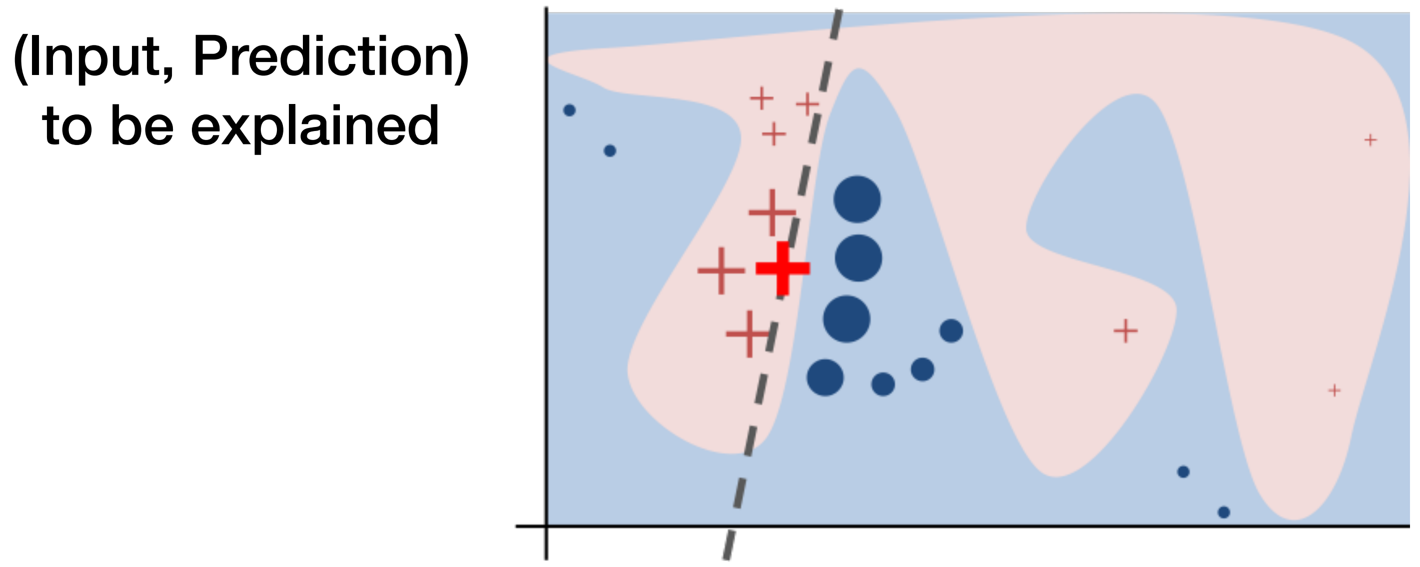
- What if the model is already trained? And we have no access to its parameters etc…
- Idea (Ribeiro et al. 2016): fit a simple interpretable model around a given query, using perturbations of the input



- But this assumes input is continuous, output is a single value. Can we extend this to text data?

INTERPRETABILITY IN NLP

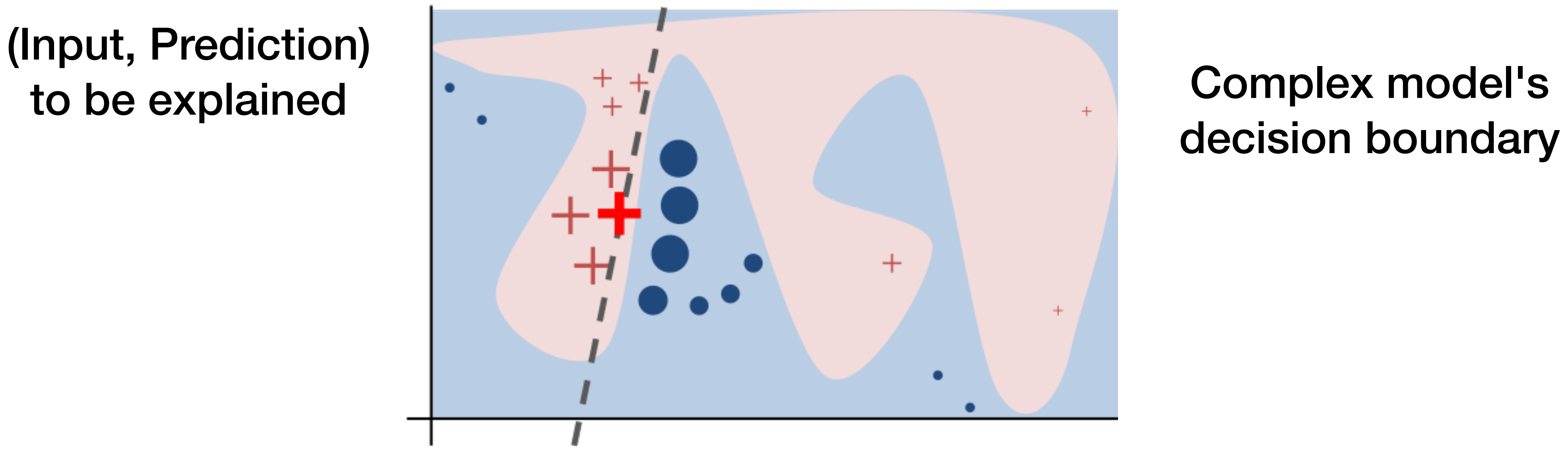
- What if the model is already trained? And we have no access to its parameters etc…
- Idea (Ribeiro et al. 2016): fit a simple interpretable model around a given query, using perturbations of the input



- But this assumes input is continuous, output is a single value. Can we extend this to text data?

INTERPRETABILITY IN NLP

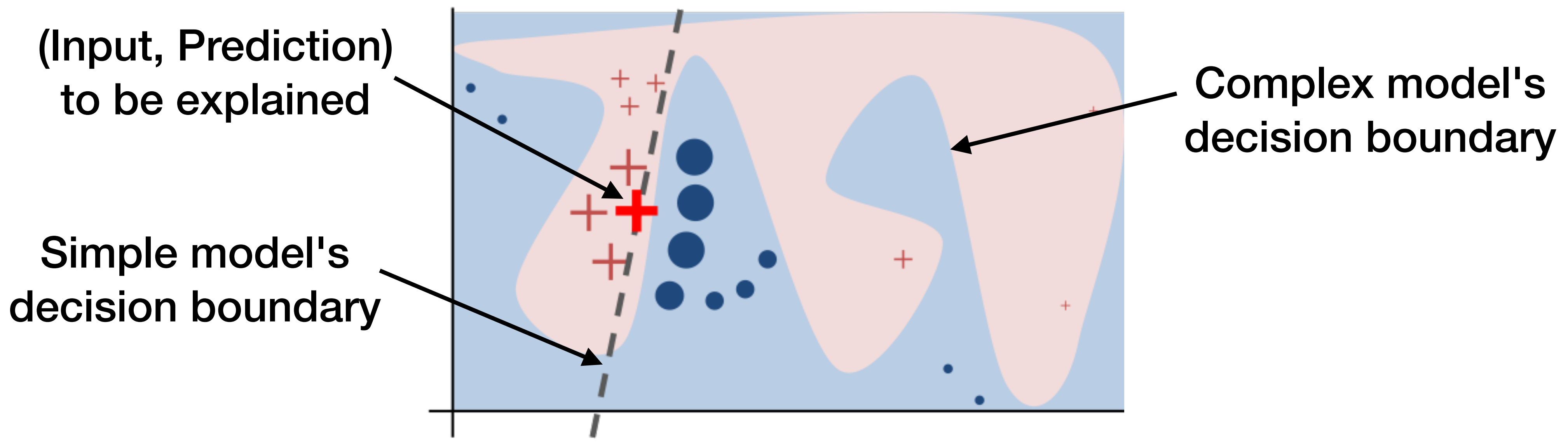
- What if the model is already trained? And we have no access to its parameters etc…
- Idea (Ribeiro et al. 2016): fit a simple interpretable model around a given query, using perturbations of the input



- But this assumes input is continuous, output is a single value. Can we extend this to text data?

INTERPRETABILITY IN NLP

- What if the model is already trained? And we have no access to its parameters etc…
- Idea (Ribeiro et al. 2016): fit a simple interpretable model around a given query, using perturbations of the input



- But this assumes input is continuous, output is a single value. Can we extend this to text data?

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

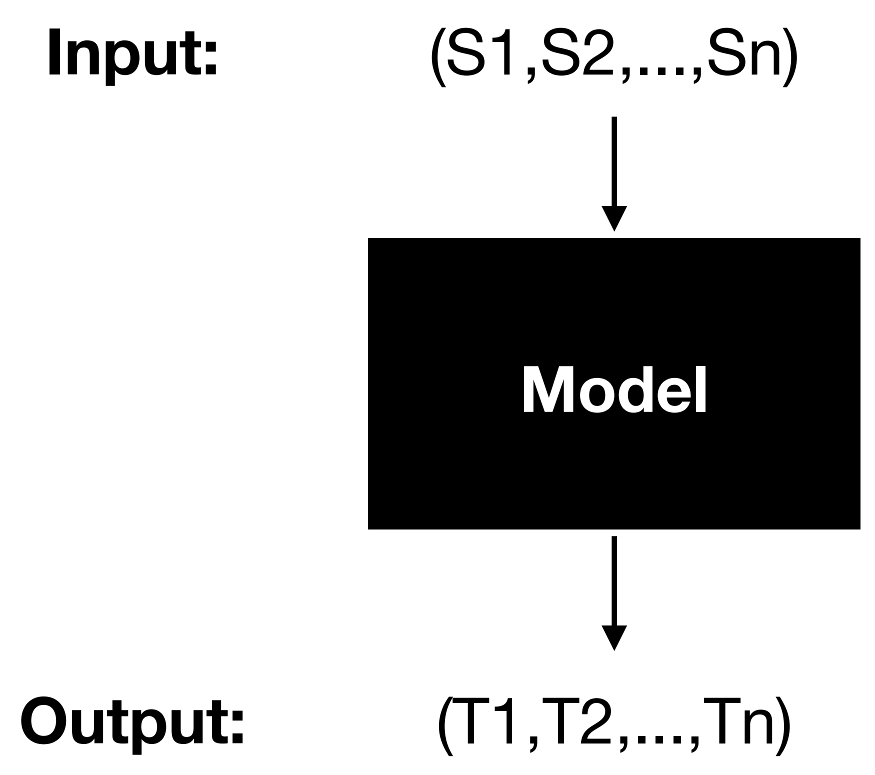
- Weighted bipartite graph summarizes local behavior of the model.

- Explanation:

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

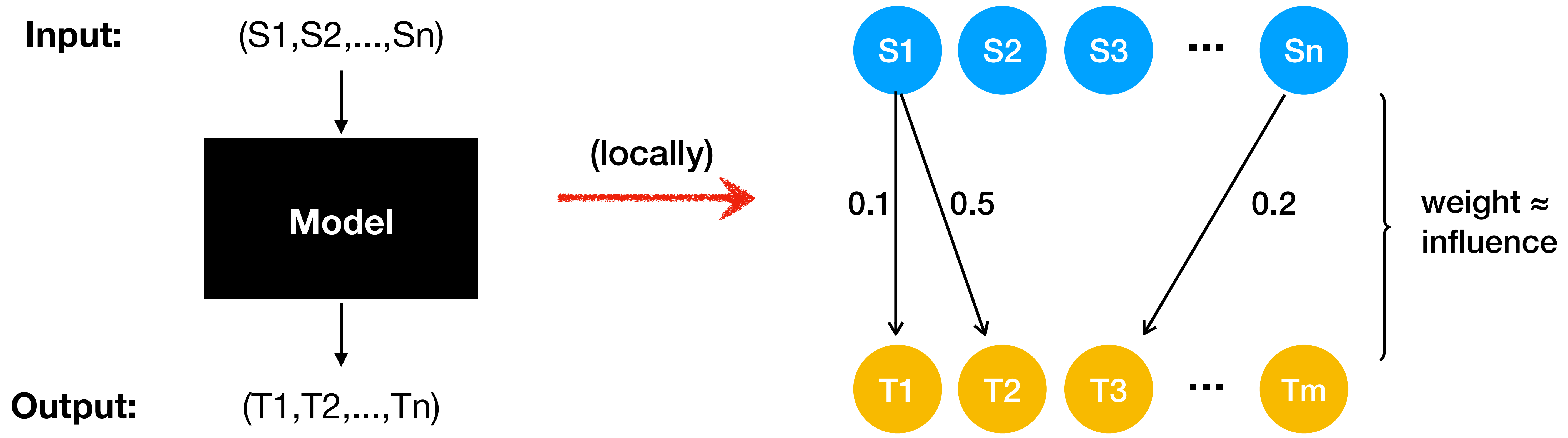
- Weighted bipartite graph summarizes local behavior of the model.



- Explanation:

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

- Weighted bipartite graph summarizes local behavior of the model.

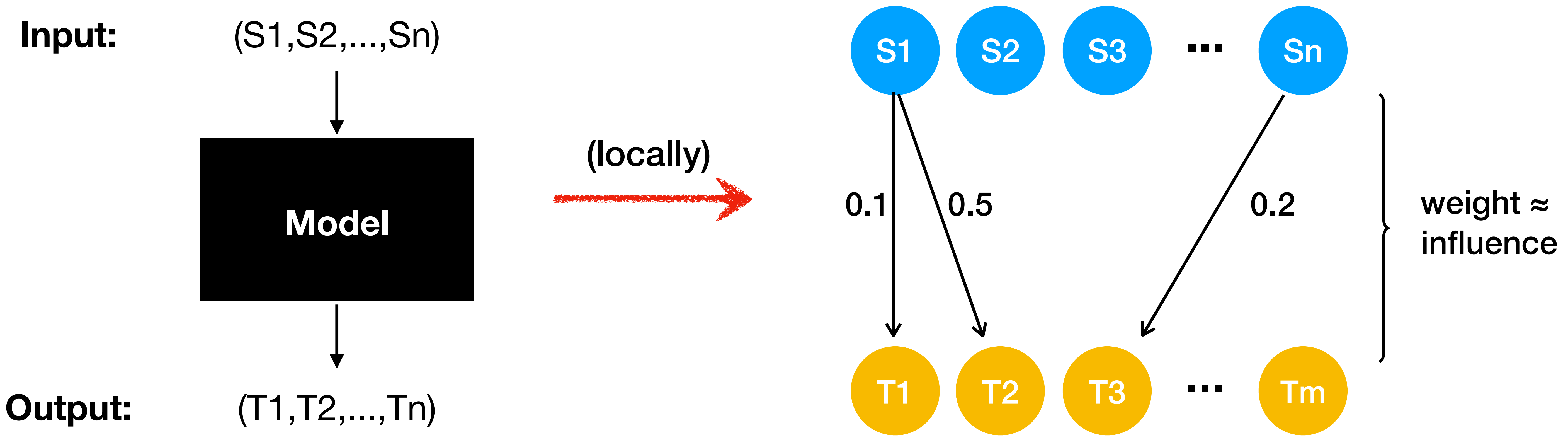


- Explanation:

$$E_{x\to y} = \{G^1, \ldots, G^k\}$$

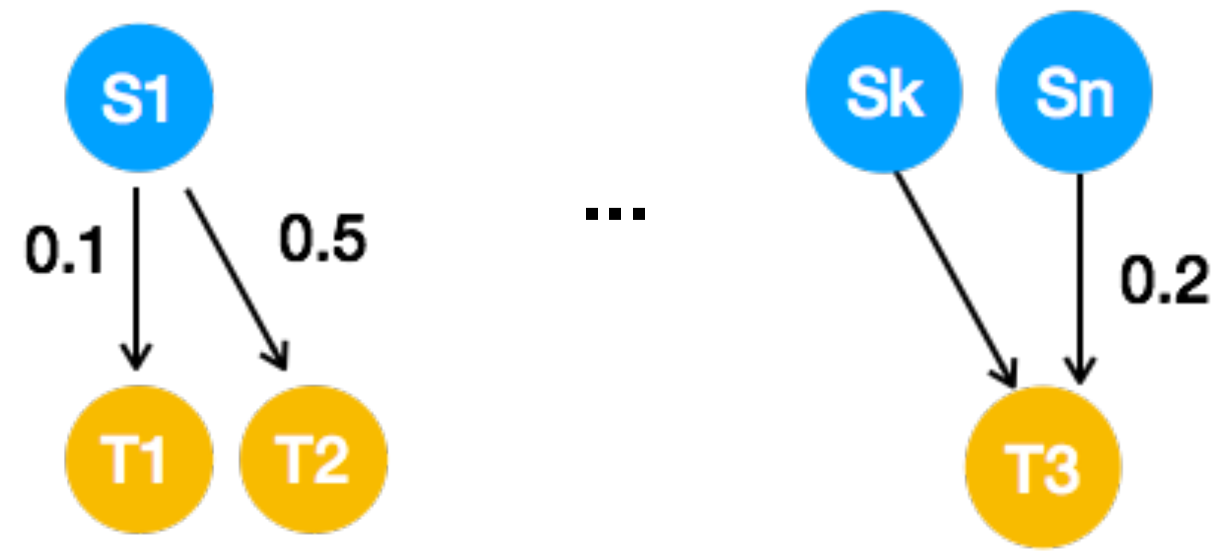
INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

- Weighted bipartite graph summarizes local behavior of the model.



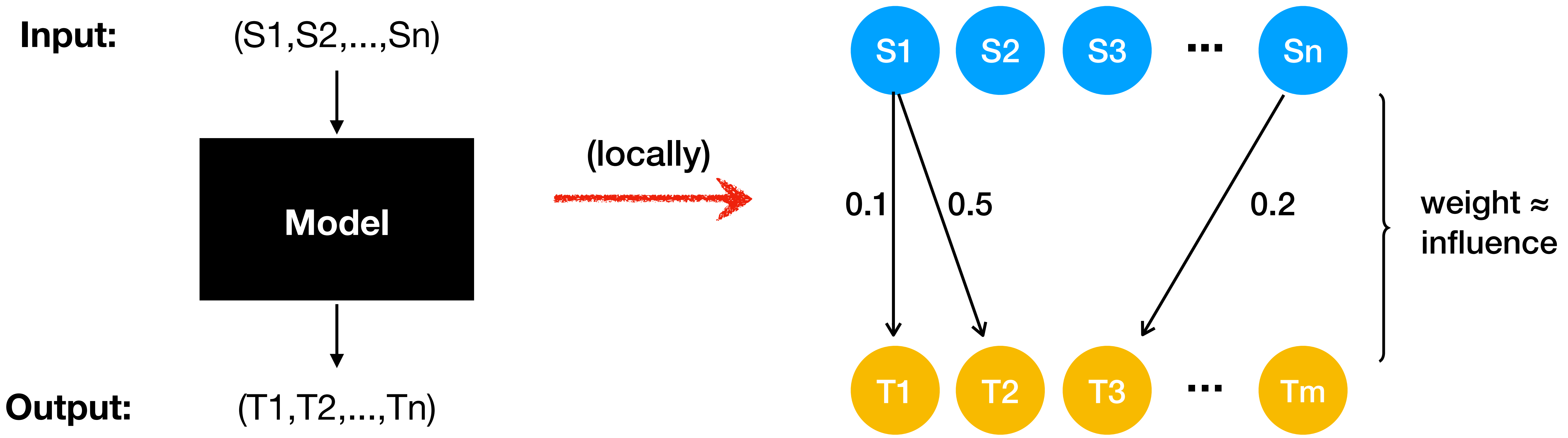
- Explanation:

$$E_{x \to y} = \{G^1, \ldots, G^k\}$$



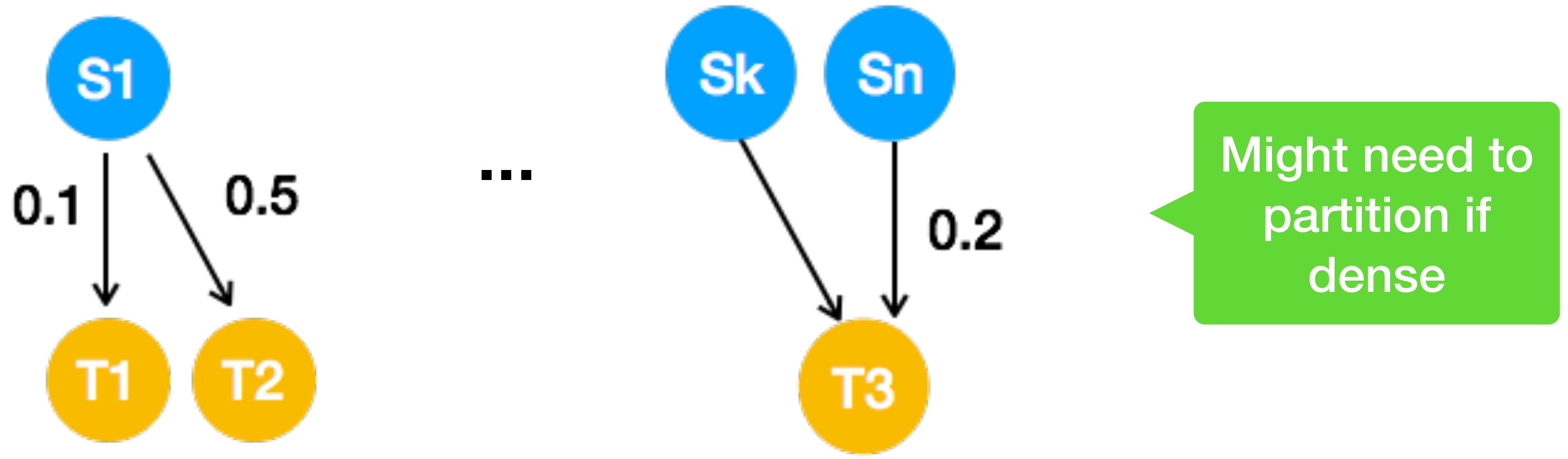
INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

- Weighted bipartite graph summarizes local behavior of the model.



- Explanation:

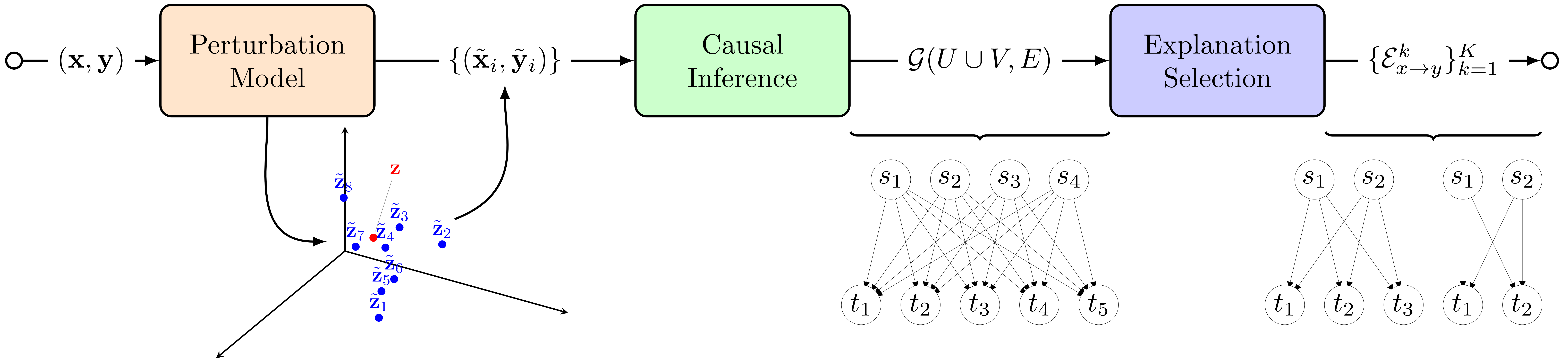
$$E_{x \to y} = \{G^1, \dots, G^k\}$$



INTERPRETABILITY VIA LOCAL APPROXIMATION

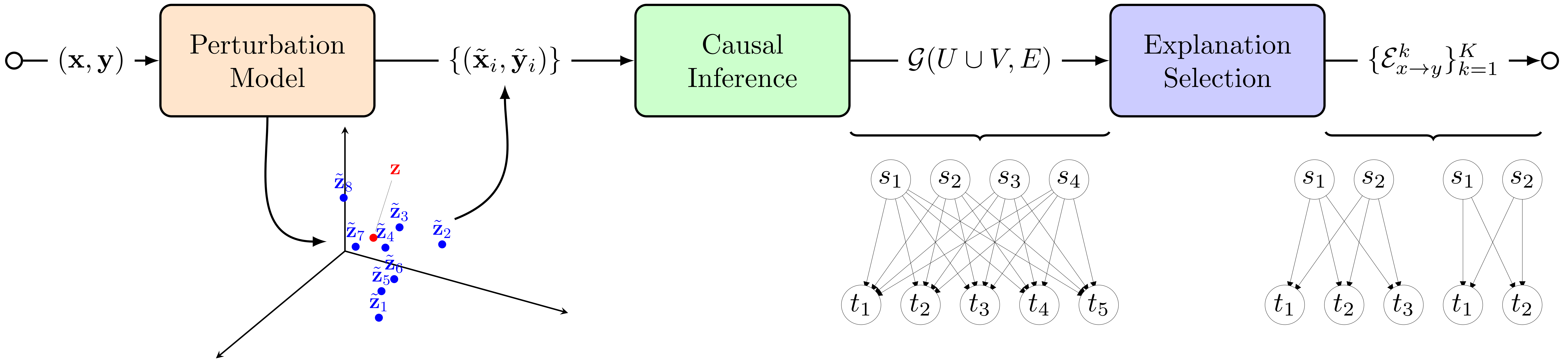
[AM & Jaakkola, 2018]

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]



INTERPRETABILITY VIA LOCAL APPROXIMATION

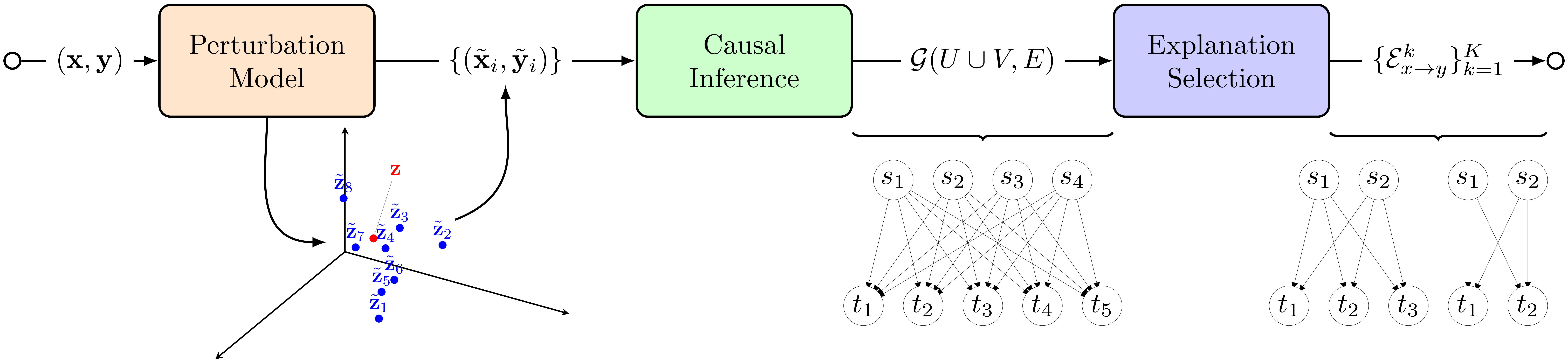
[AM & Jaakkola, 2018]



1. Encode input to vector representation z

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

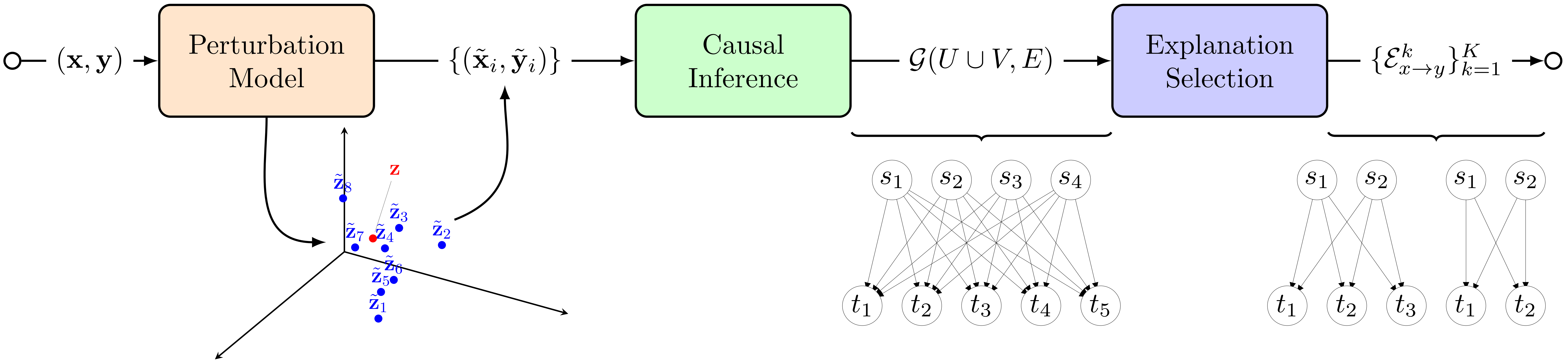


1. Encode input to vector representation z

2. Generate samples around z

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

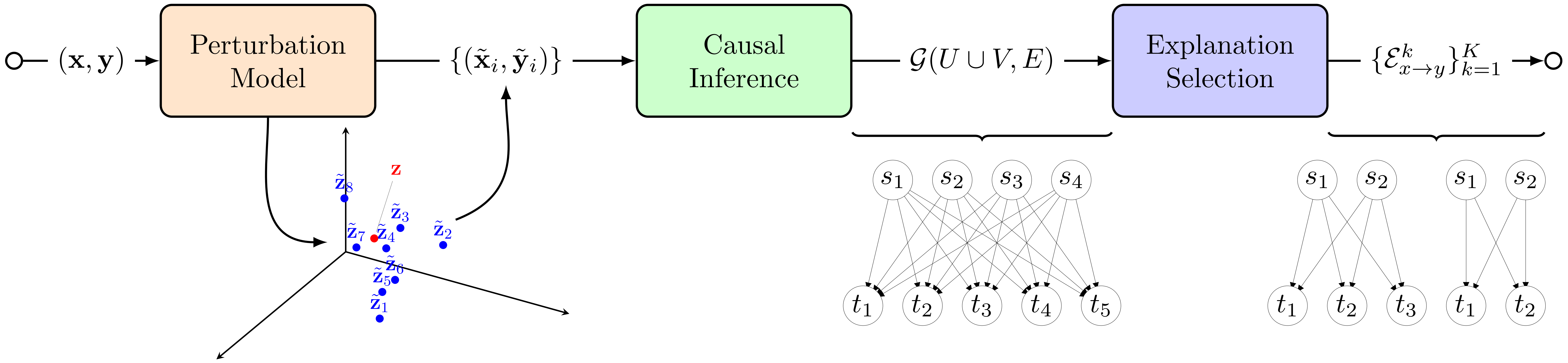


1. Encode input to vector representation z

2. Generate samples around z

3. Decode samples into sequences

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]



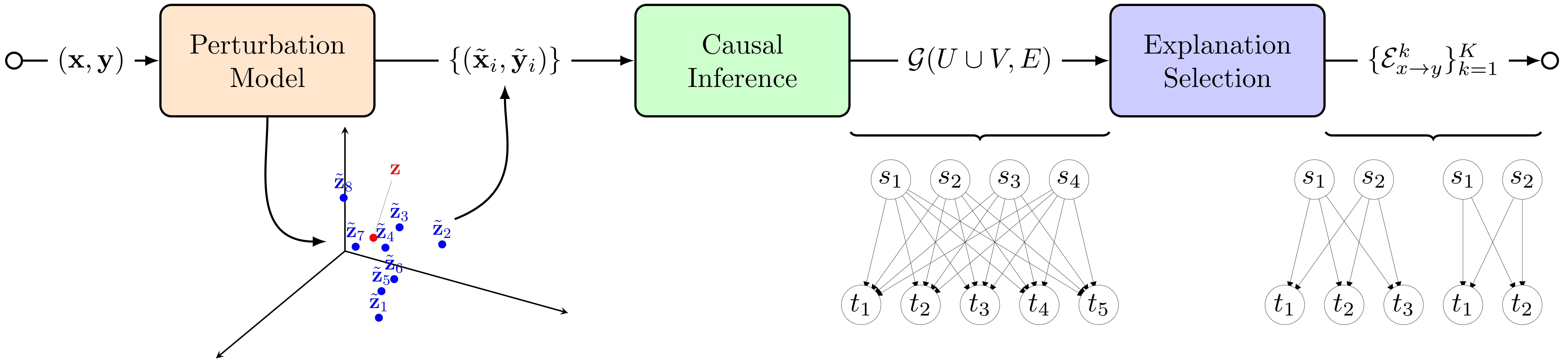
1. Encode input to vector representation z

2. Generate samples around z

3. Decode samples into sequences

4. Map perturbed sequences using decoder

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]



1. Encode input to vector representation z

2. Generate samples around z

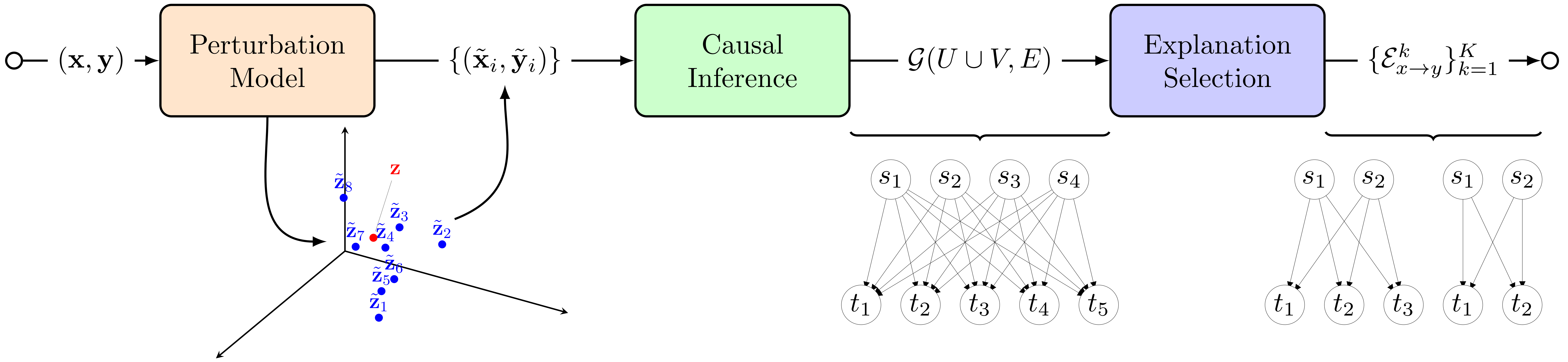
3. Decode samples into sequences

4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]



1. Encode input to vector representation z

2. Generate samples around z

3. Decode samples into sequences

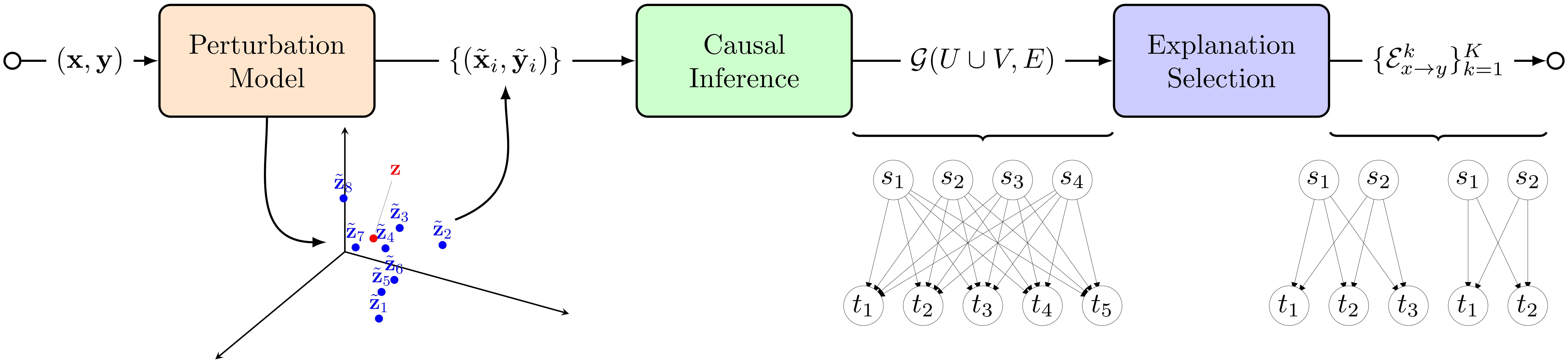
4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens

- Simplest approach: logistic regression

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

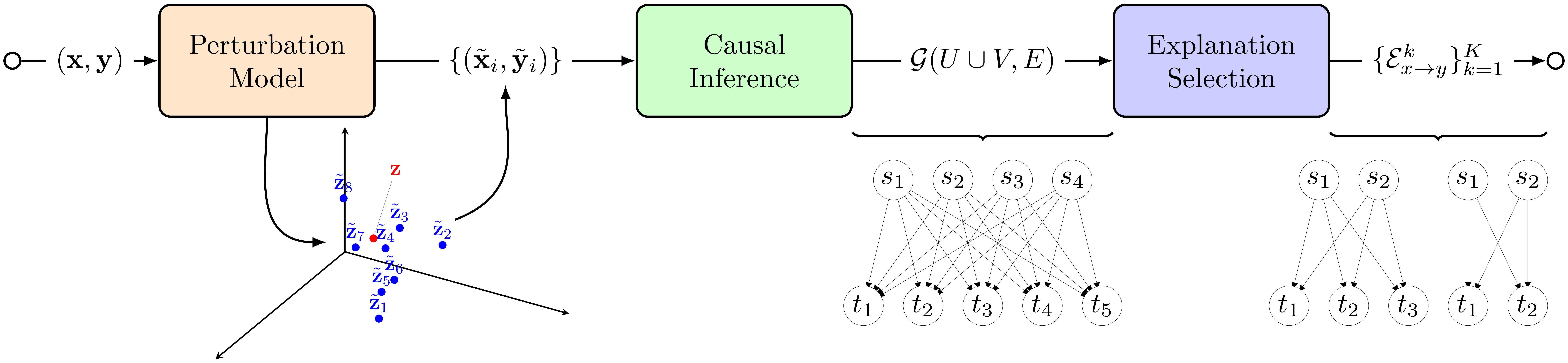


1. Encode input to vector representation z
2. Generate samples around z
3. Decode samples into sequences
4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens
- Simplest approach: logistic regression
- Account for uncertainty: Bayesian LR

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]



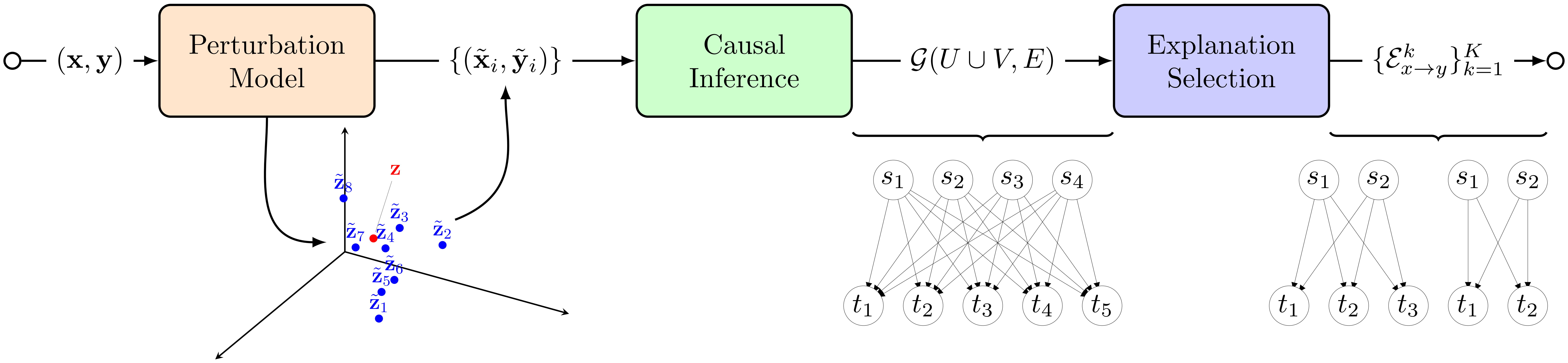
1. Encode input to vector representation z
2. Generate samples around z
3. Decode samples into sequences
4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens
- Simplest approach: logistic regression
- Account for uncertainty: Bayesian LR

- For large inputs/outputs, dense graph might not be interpretable

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

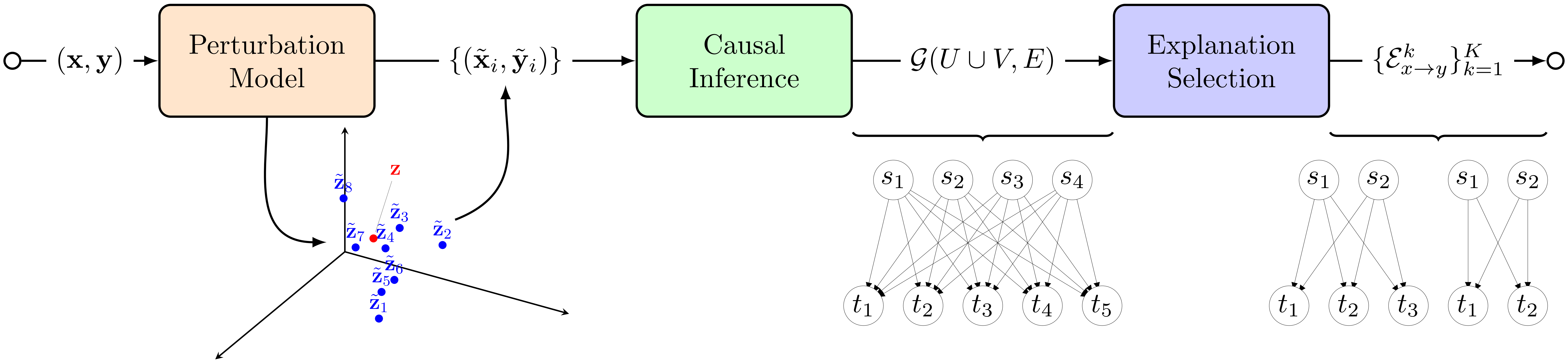


1. Encode input to vector representation z
2. Generate samples around z
3. Decode samples into sequences
4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens
- Simplest approach: logistic regression
- Account for uncertainty: Bayesian LR

- For large inputs/outputs, dense graph might not be interpretable
- Cast as k-cut graph partitioning

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]



1. Encode input to vector representation z
2. Generate samples around z
3. Decode samples into sequences
4. Map perturbed sequences using decoder

- Using perturbations, infer dependencies between original input/output tokens
- Simplest approach: logistic regression
- Account for uncertainty: Bayesian LR

- For large inputs/outputs, dense graph might not be interpretable
- Cast as k-cut graph partitioning
- Graph partitioning with uncertainty [Fan et al. 2012]

INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

Application: explaining biases in machine translation systems

Model: Azure MT service (via API), English to French

Inputs: Sentences containing bias-prone words

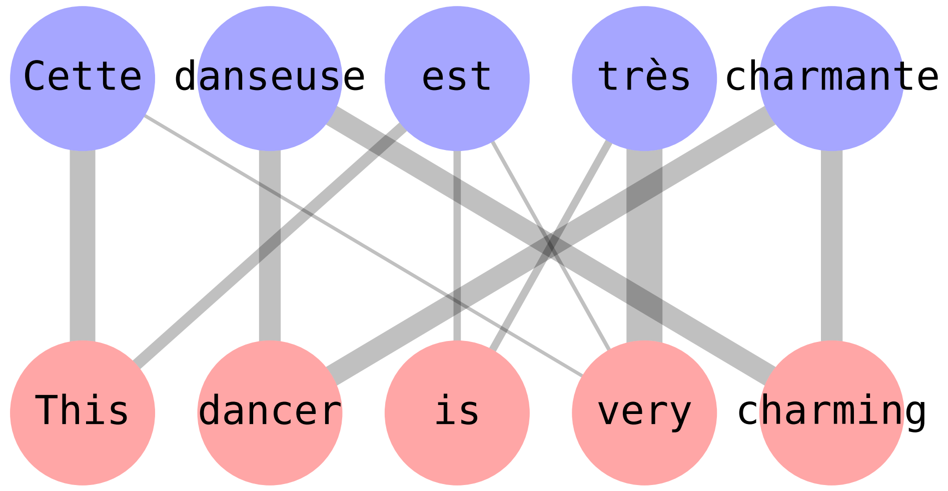
Findings: Model exhibits strong unexplained grammatical gender preferences.

- Chooses masculine in sentences containing doctor, professor, smart, talented
- Chooses feminine in sentences containing dancer, nurse, charming, compassionate

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

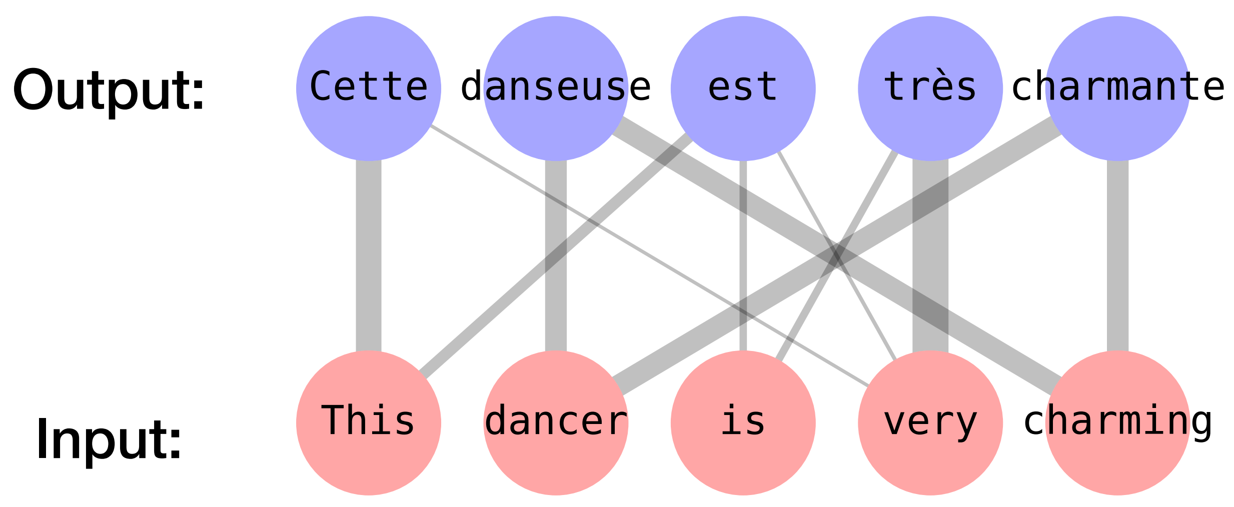
Application: explaining biases in MT systems



INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

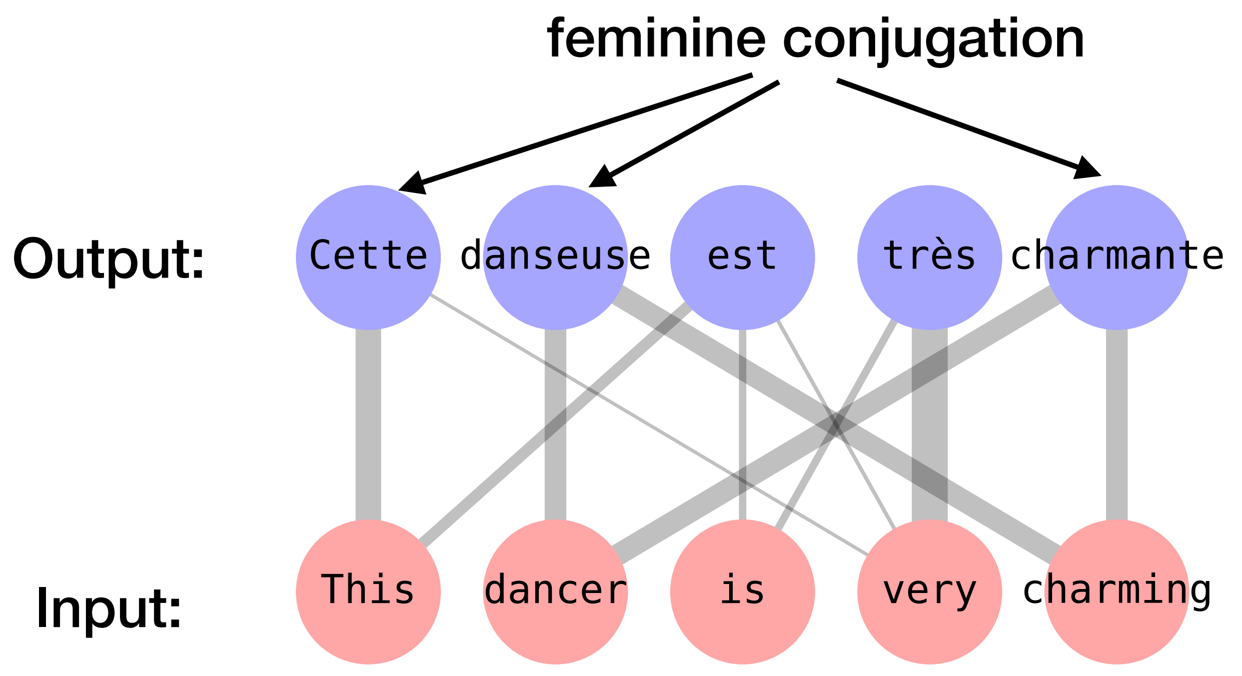
Application: explaining biases in MT systems



INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

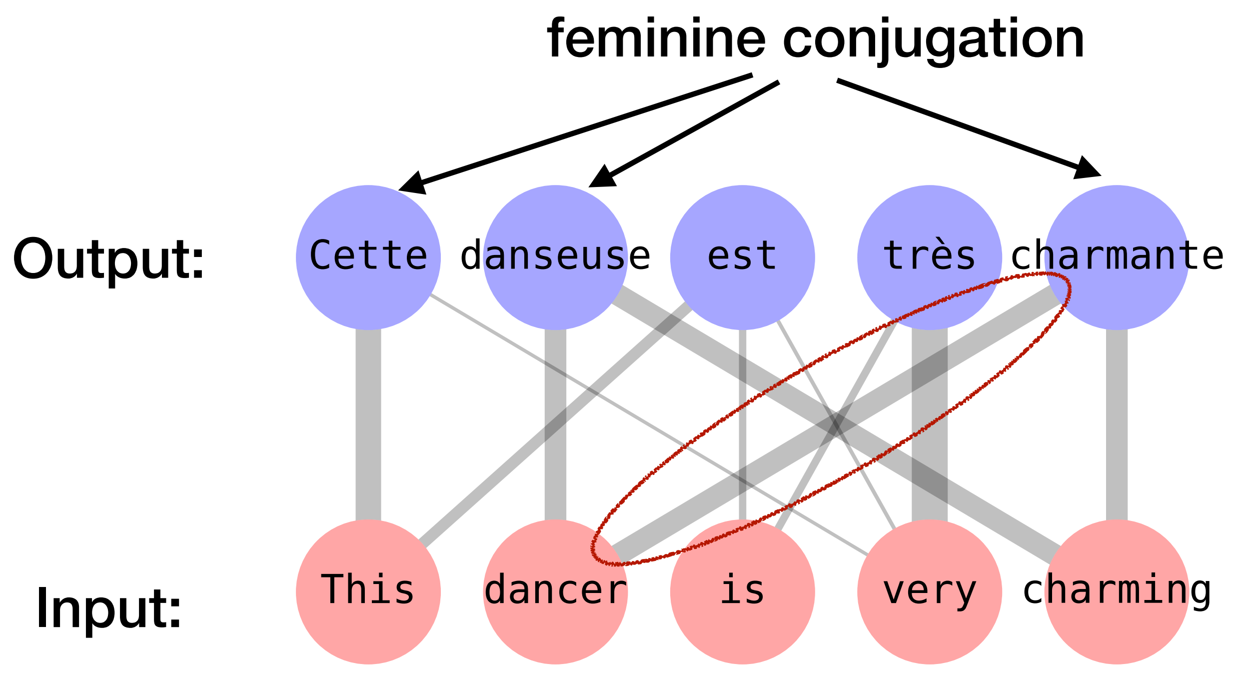
Application: explaining biases in MT systems



INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

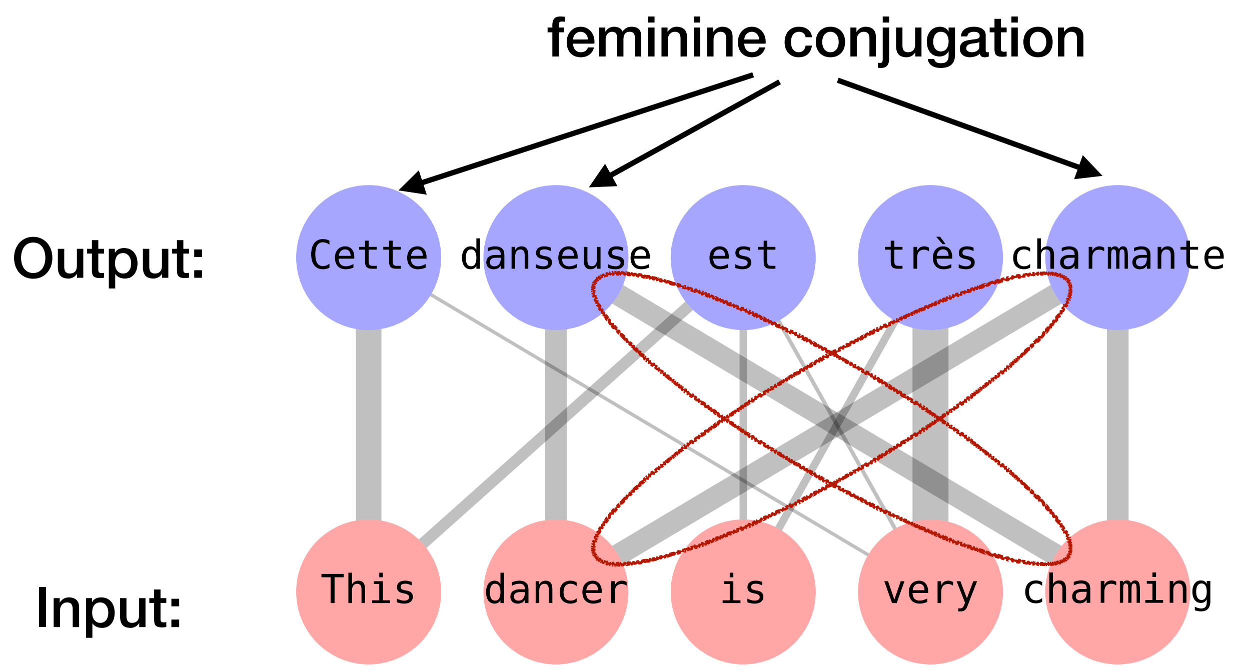
Application: explaining biases in MT systems



INTERPRETABILITY VIA LOCAL APPROXIMATION

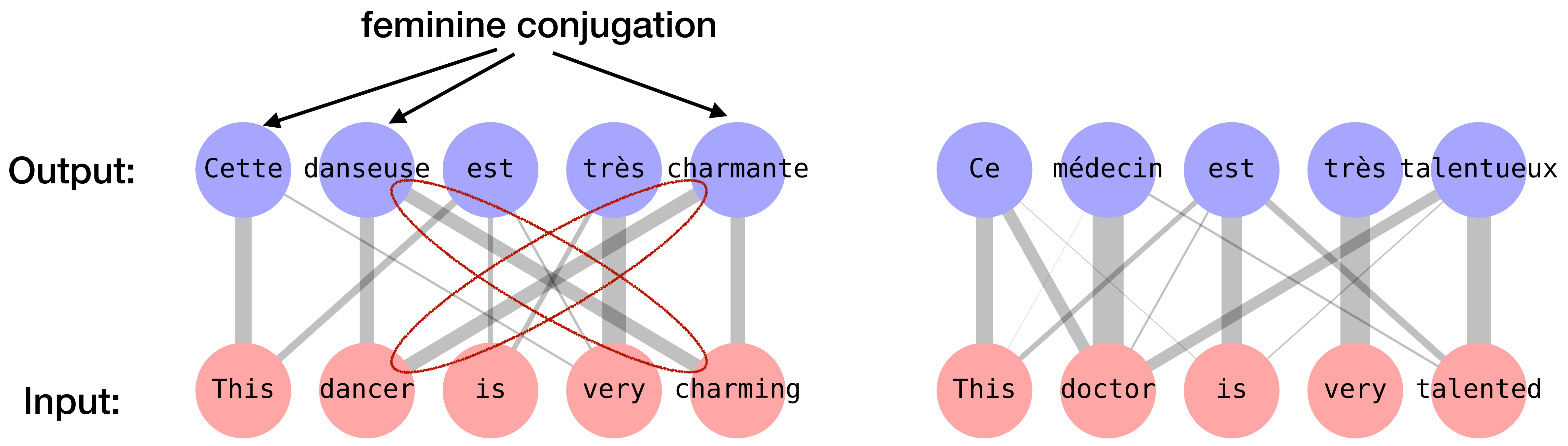
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



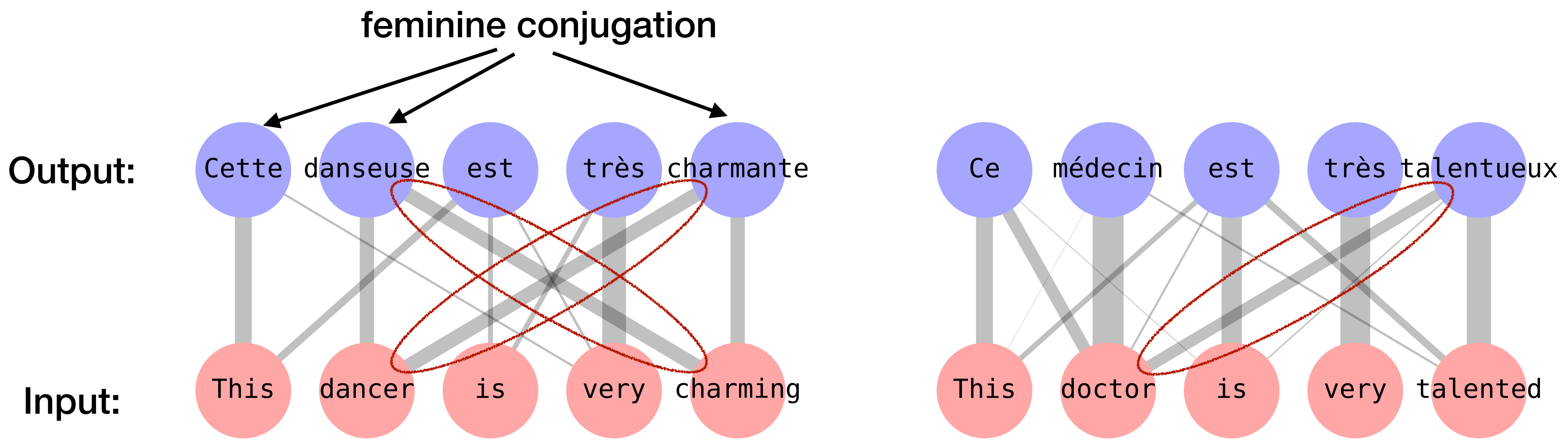
INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



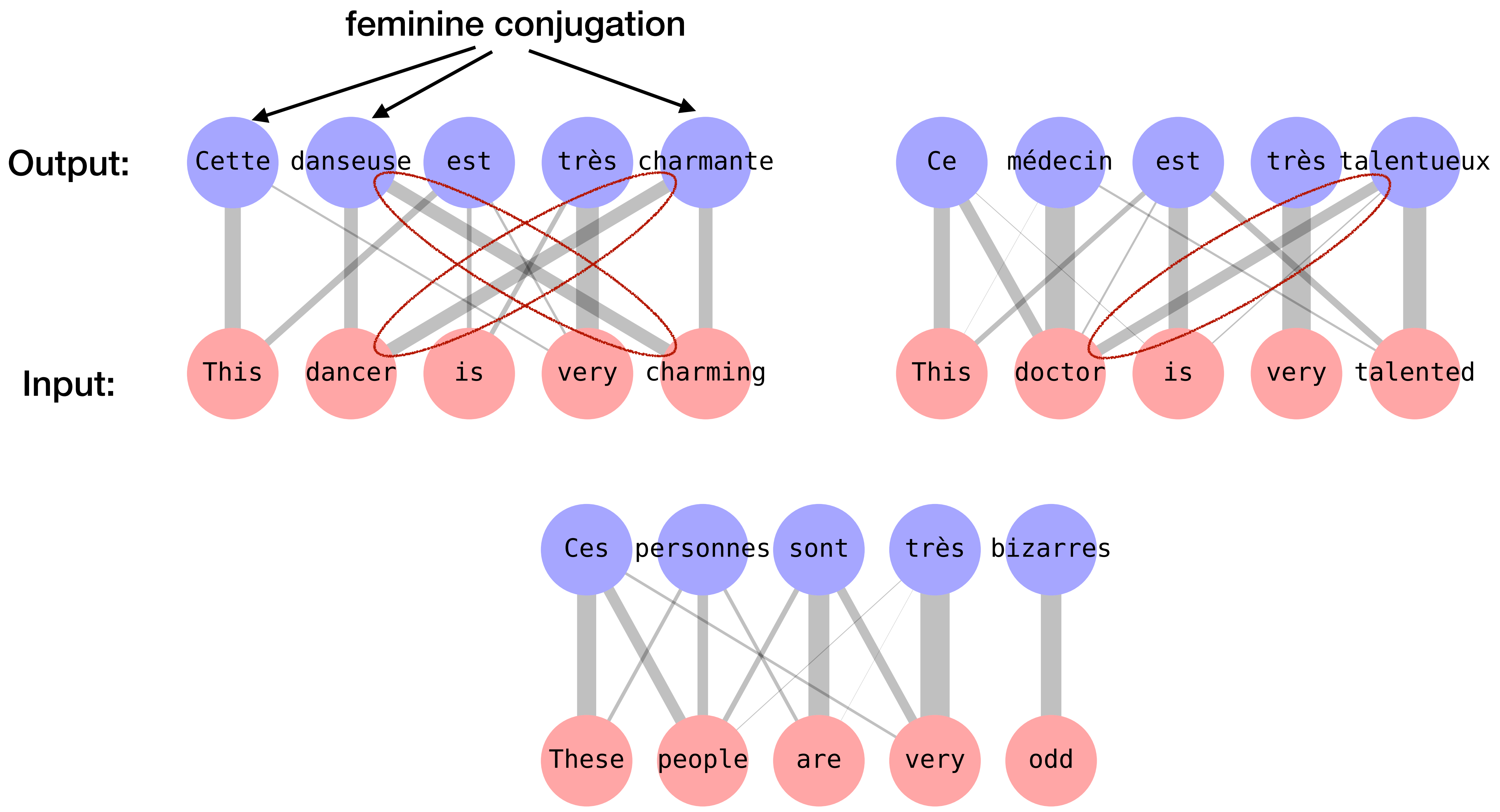
INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



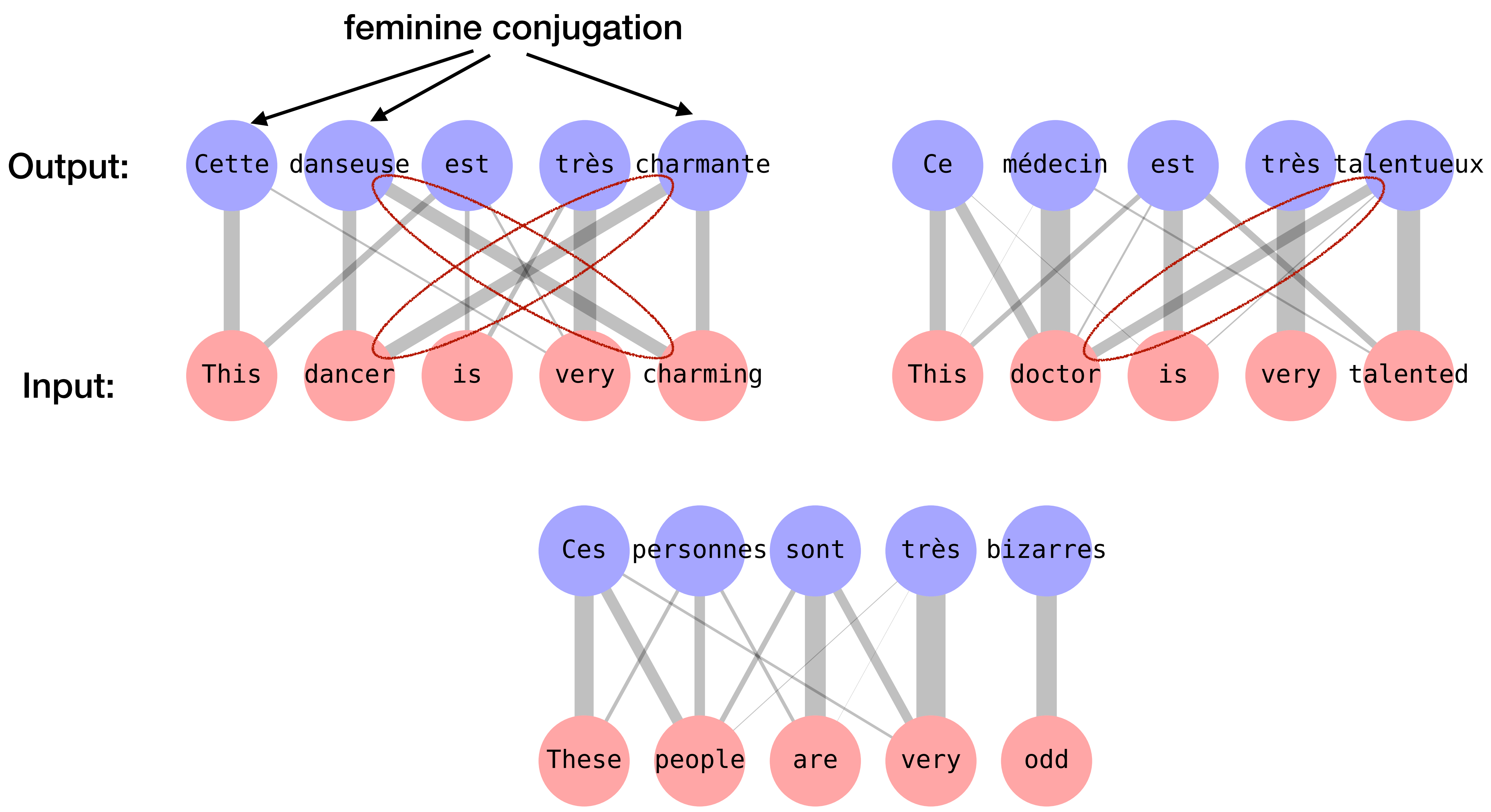
INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



INTERPRETABILITY VIA LOCAL APPROXIMATION
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



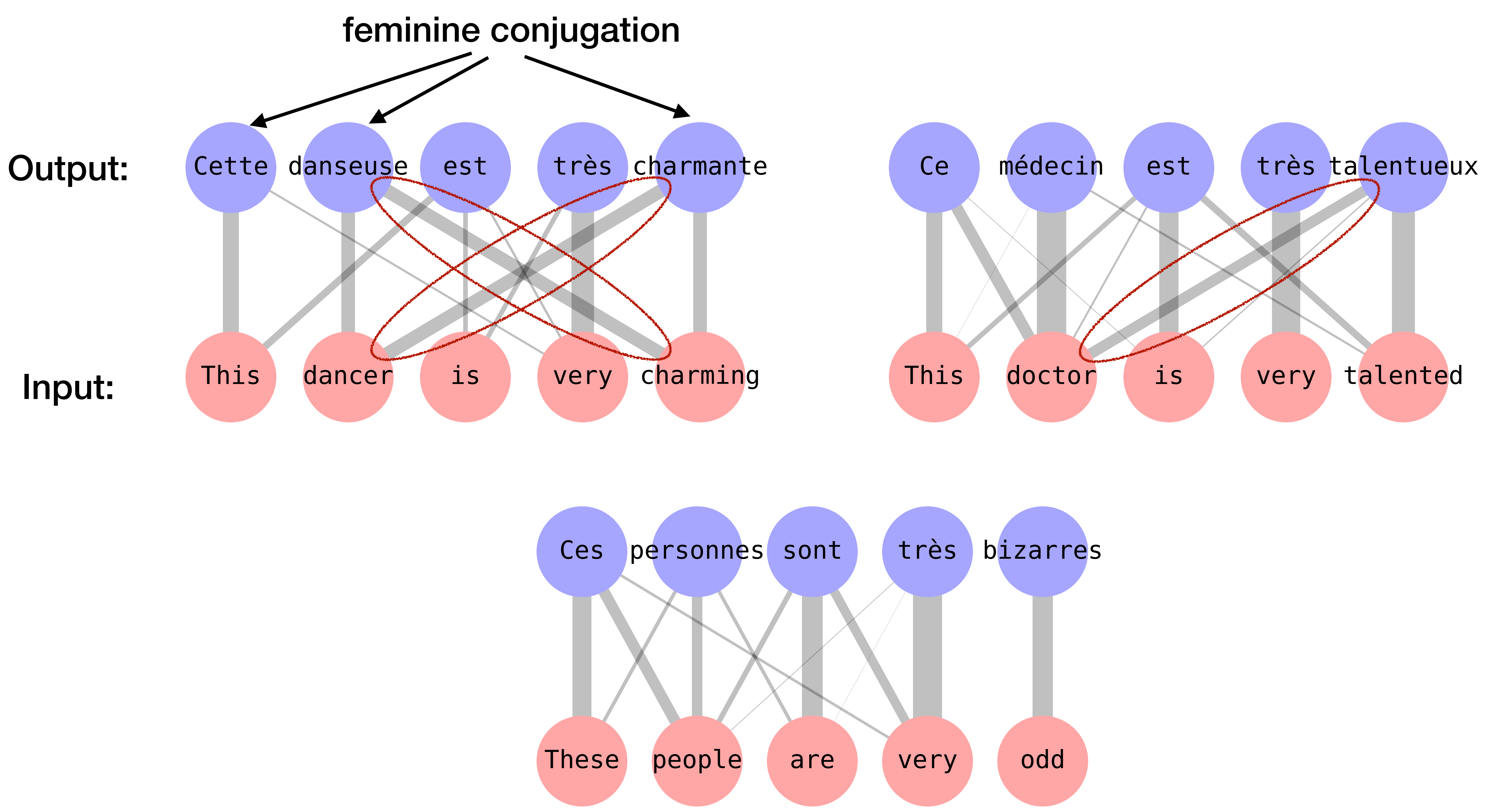
Application: flaw detection in dialogue systems

Input	Prediction
What do you mean it doesn't matter?	*I don't know*
Perhaps have we met before?	*I don't think so*
Can I get you two a cocktail?	*No, thanks.*

INTERPRETABILITY VIA LOCAL APPROXIMATION

[AM & Jaakkola, 2018]

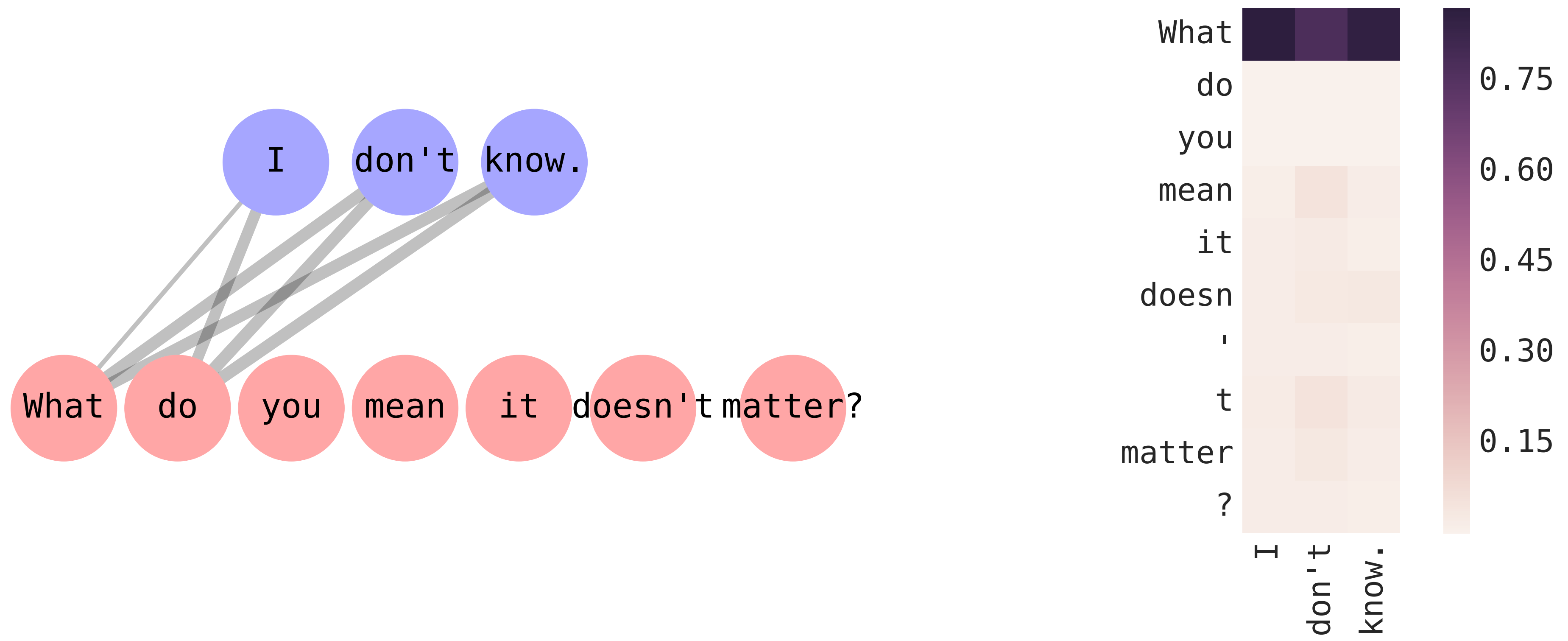
Application: explaining biases in MT systems



Application: flaw detection in dialogue systems

seq2seq with attention

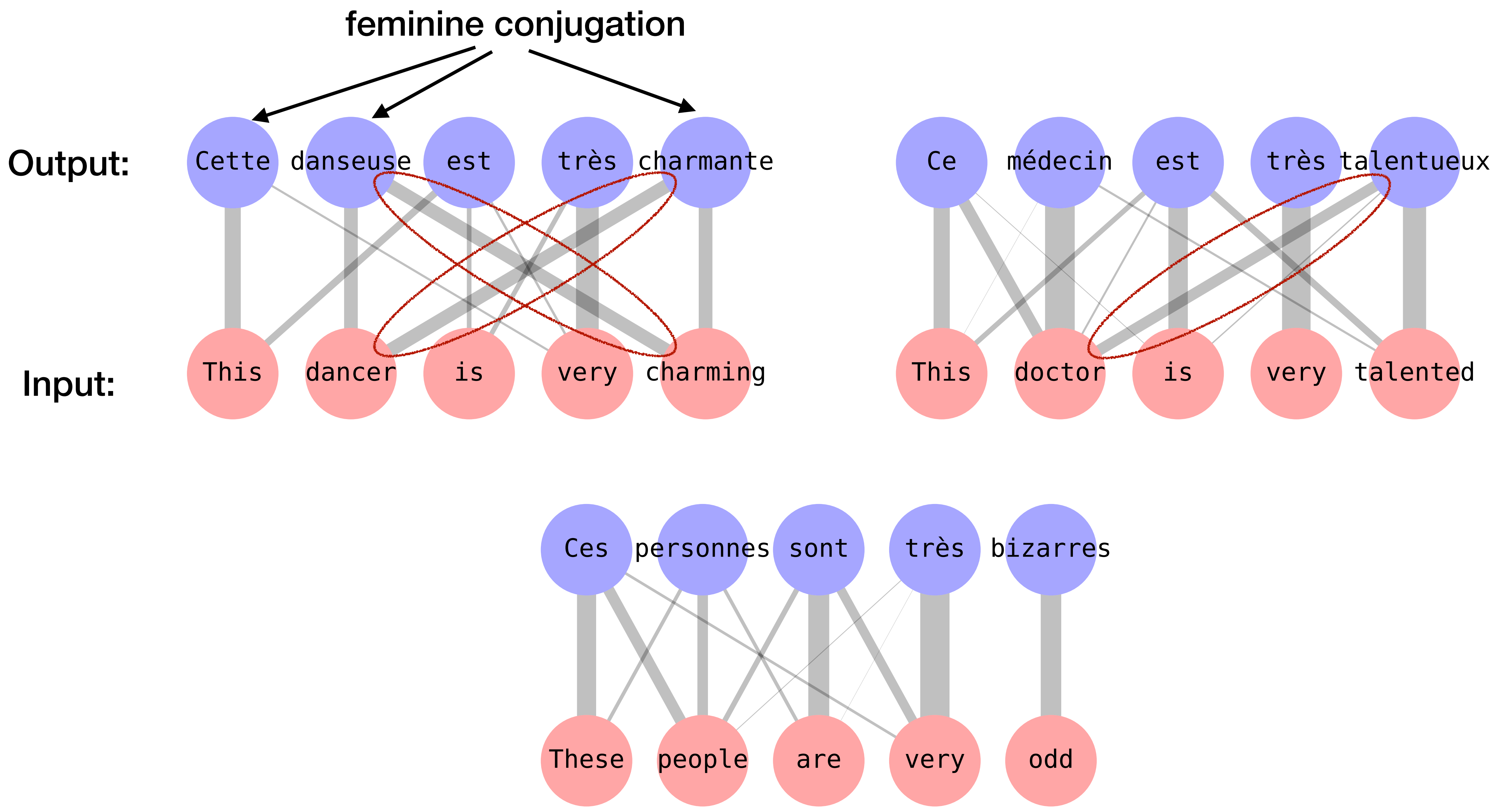
Input	Prediction
What do you mean it doesn't matter?	*I don't know*
Perhaps have we met before?	*I don't think so*
Can I get you two a cocktail?	*No, thanks.*



INTERPRETABILITY VIA LOCAL APPROXIMATION

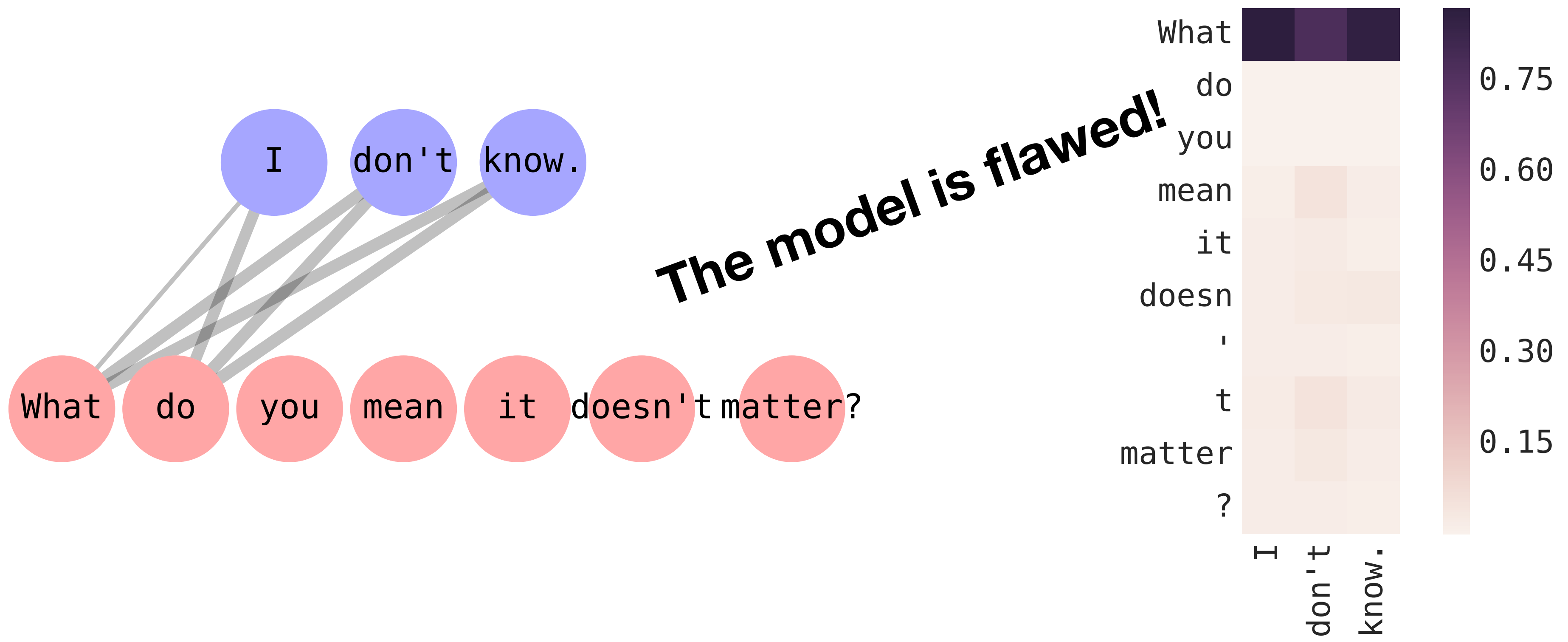
[AM & Jaakkola, 2018]

Application: explaining biases in MT systems



Application: flaw detection in dialogue systems

Input	Prediction
What do you mean it doesn't matter?	*I don't know*
Perhaps have we met before?	*I don't think so*
Can I get you two a cocktail?	*No, thanks.*

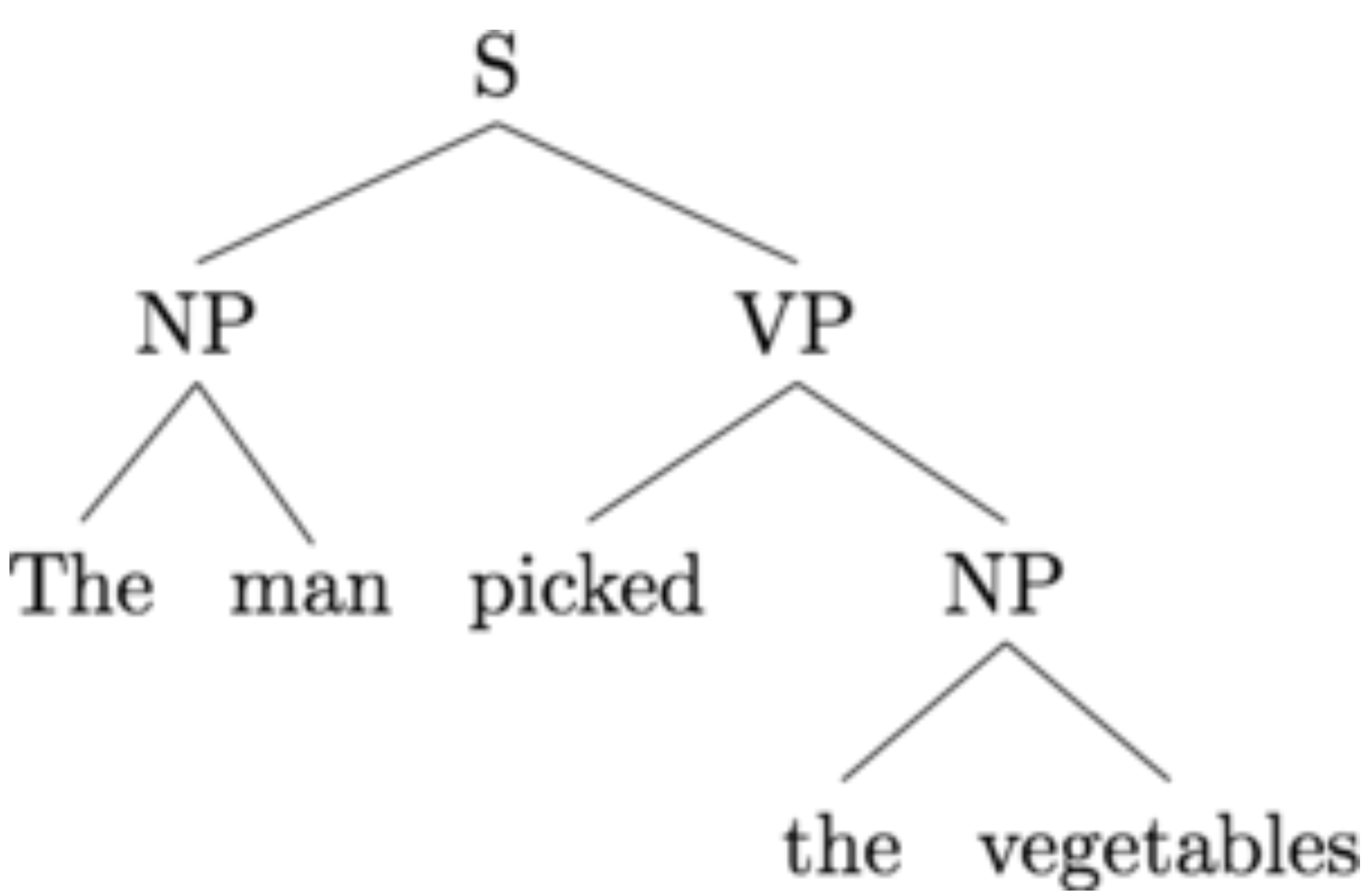


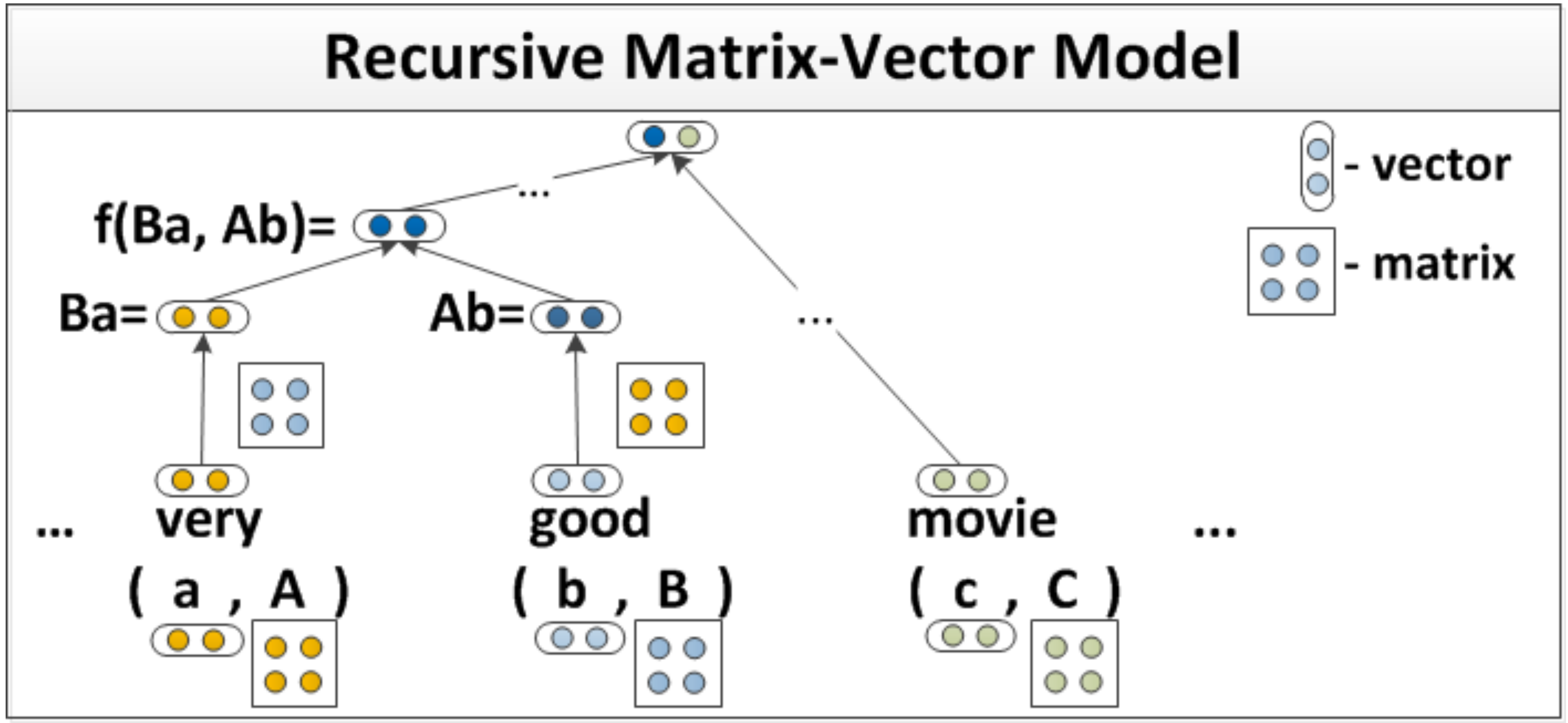
FURTHER TOPICS:

STRUCTURED NLP MODELS

WHAT ABOUT STRUCTURE?

- Language is non-linear. It has structure and compositionality [e.g. Chomsky]

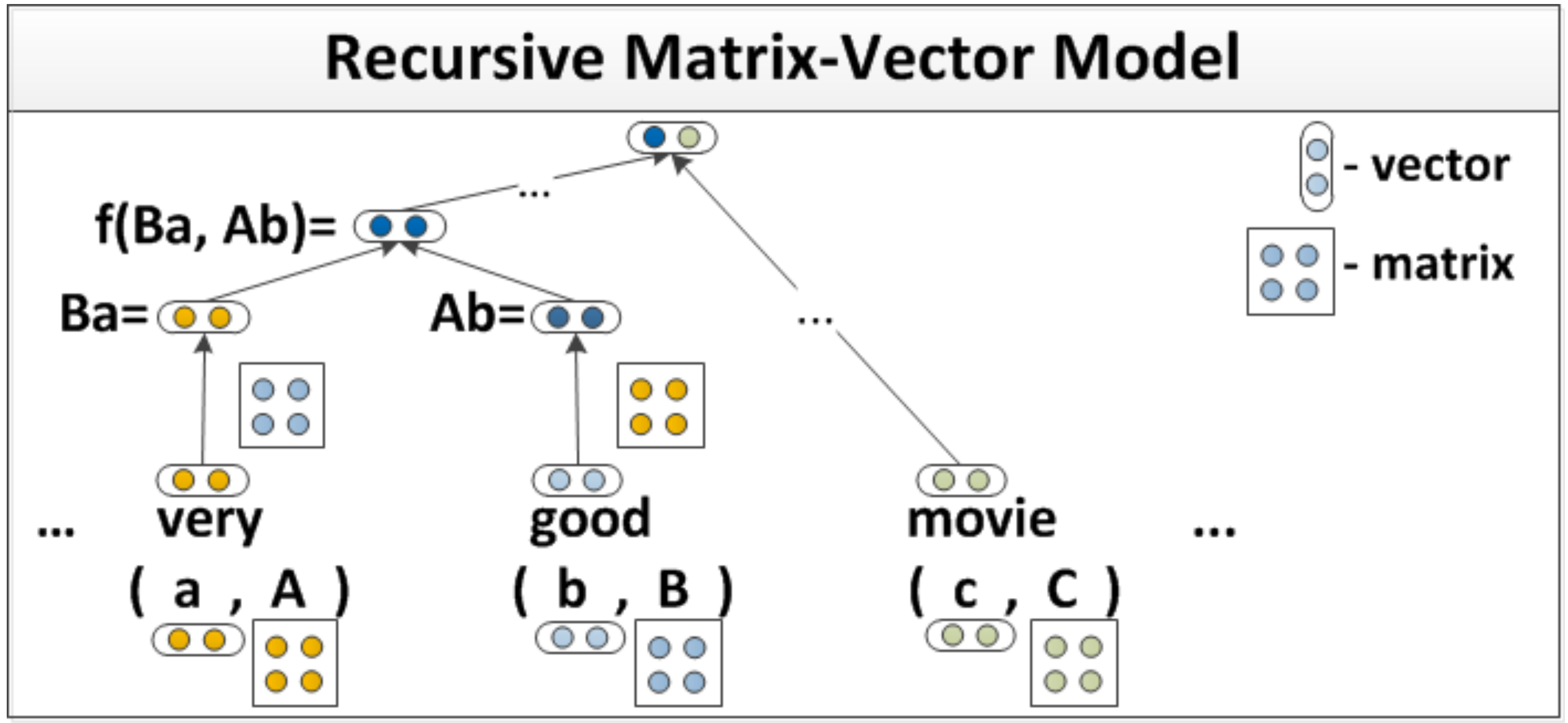




Source: socher.org

RECURSIVE NEURAL NETS

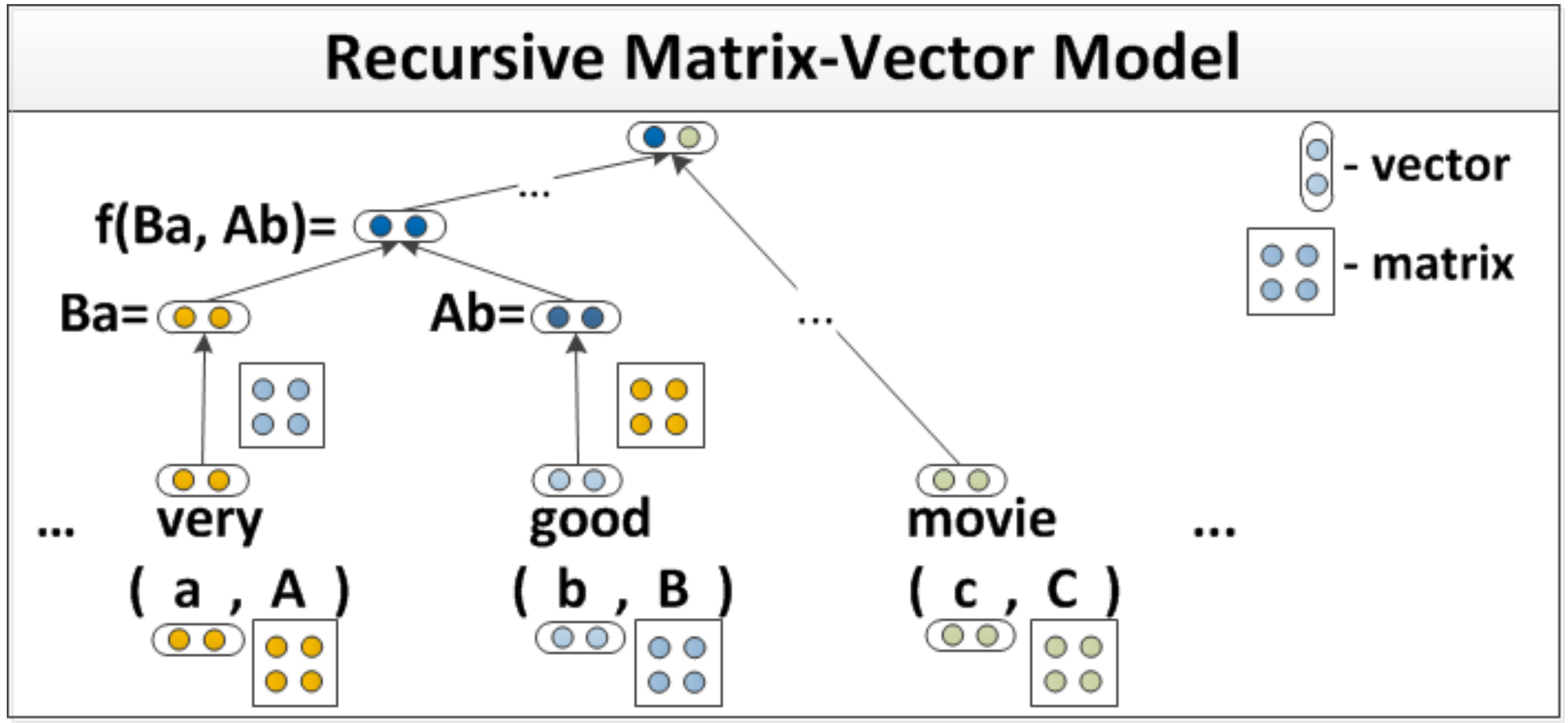
[Socher et al., 2011]

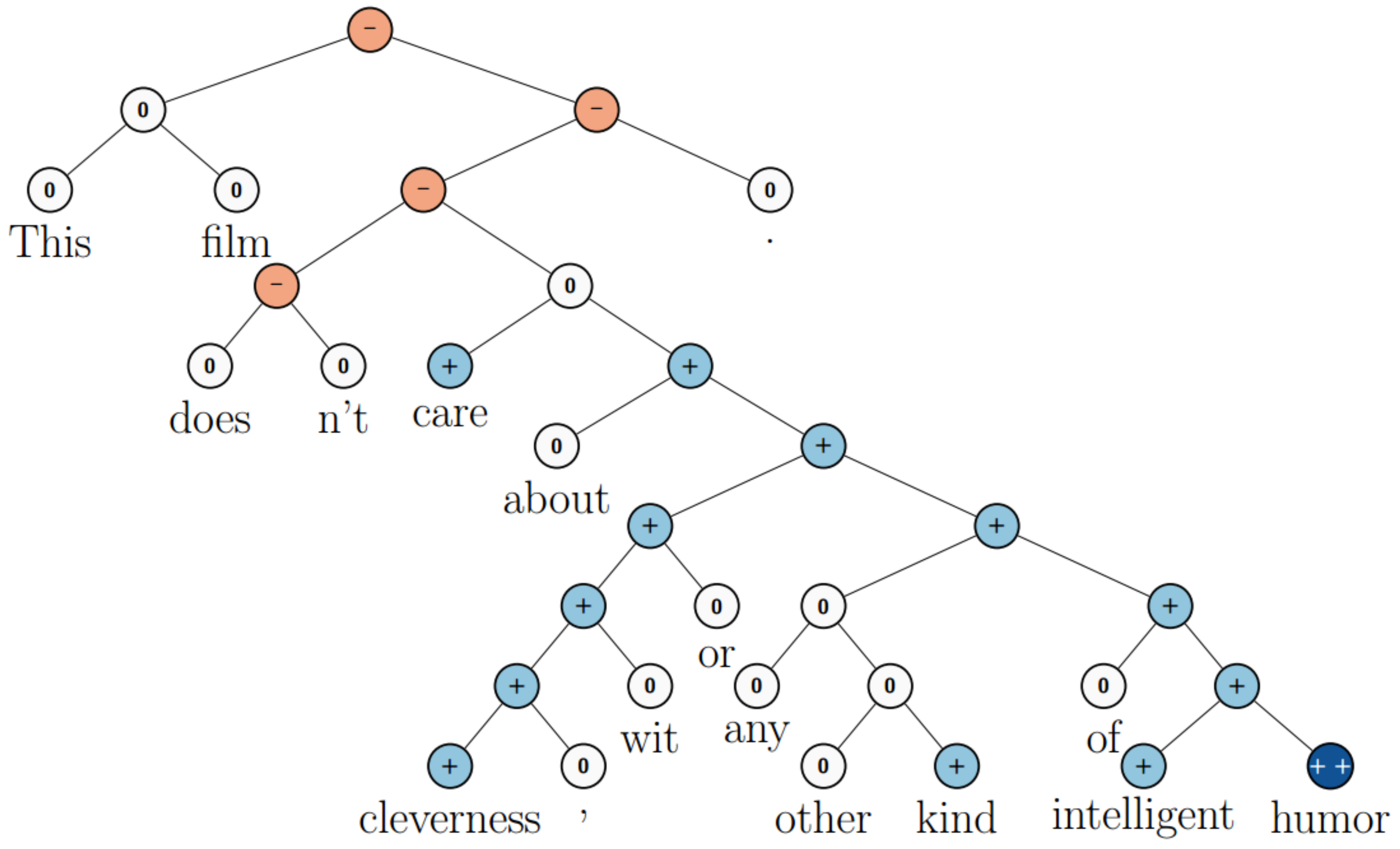


Source: socher.org

RECURSIVE NEURAL NETS

[Socher et al., 2011]



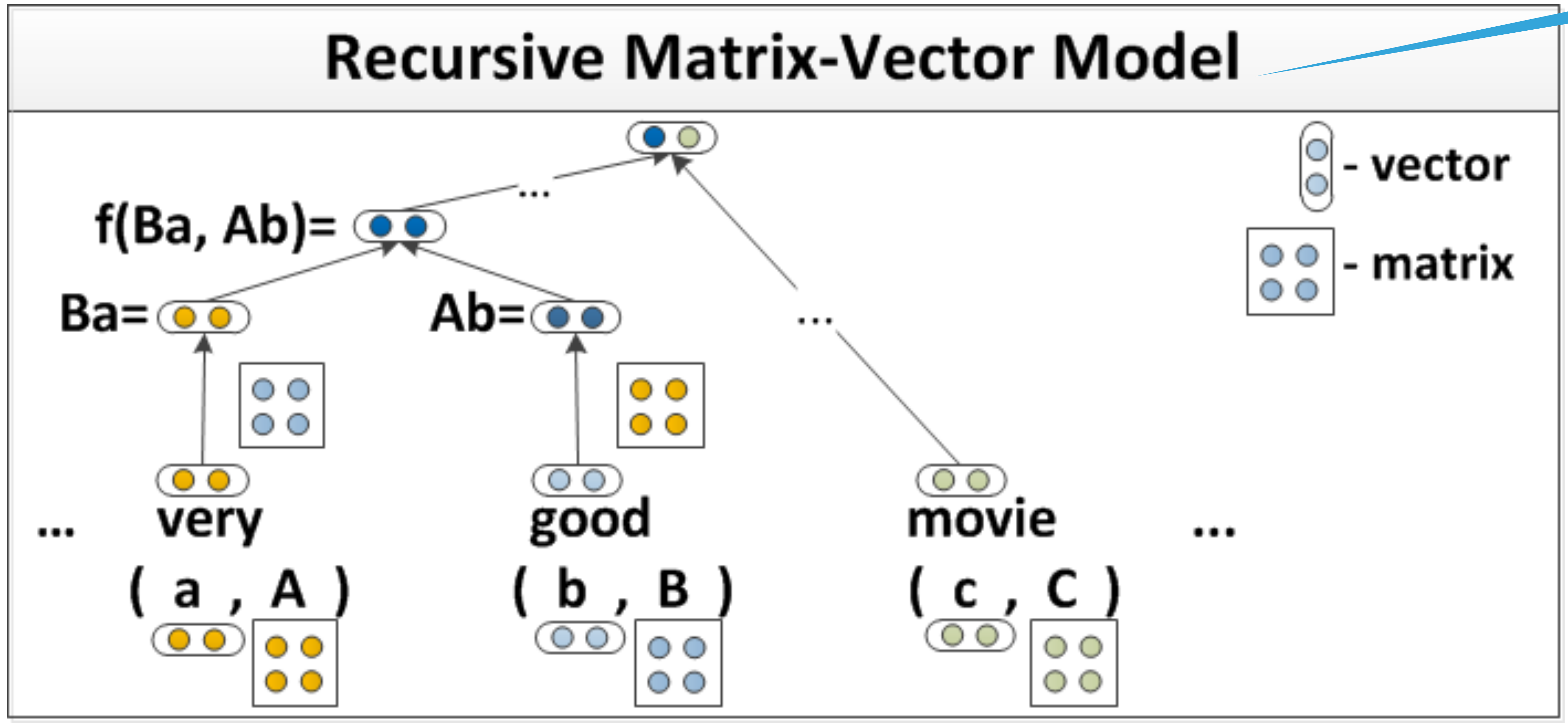


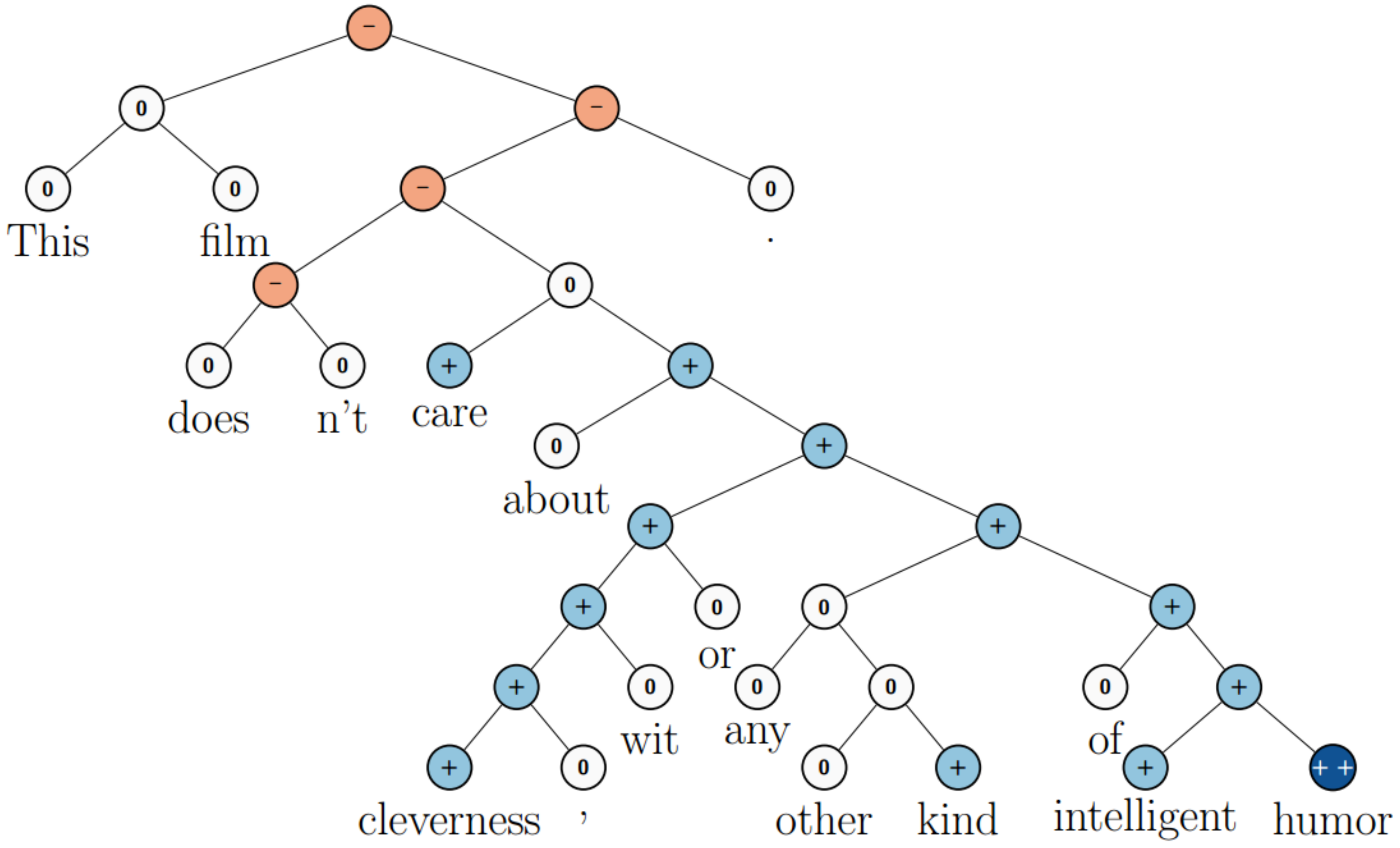
Source: socher.org

RECURSIVE NEURAL NETS

[Socher et al., 2011]

ALLOWS ENCODING OF STRUCTURE OBJECTS. WHAT ABOUT DECODING?





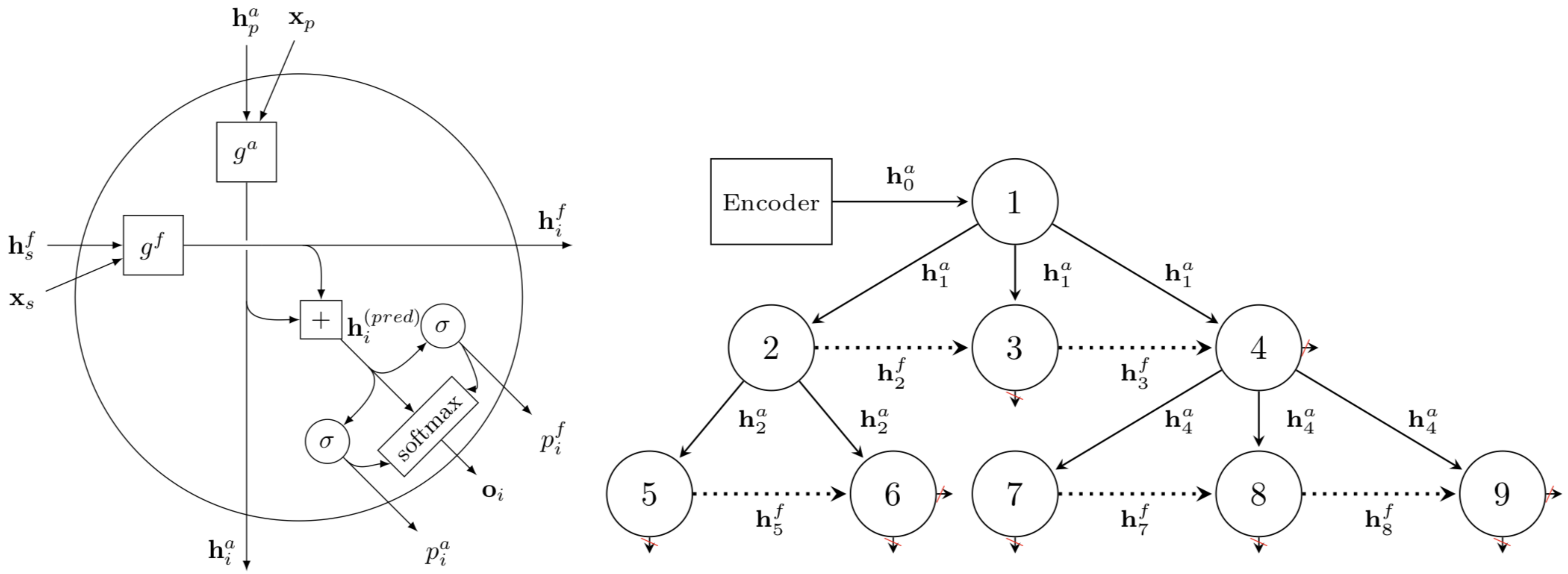
Source: socher.org

TREE TO TREE: STRUCTURED ENCODING AND DECODING

[Dong & Lapata, 2016; AM & Jaakkola, 2017]

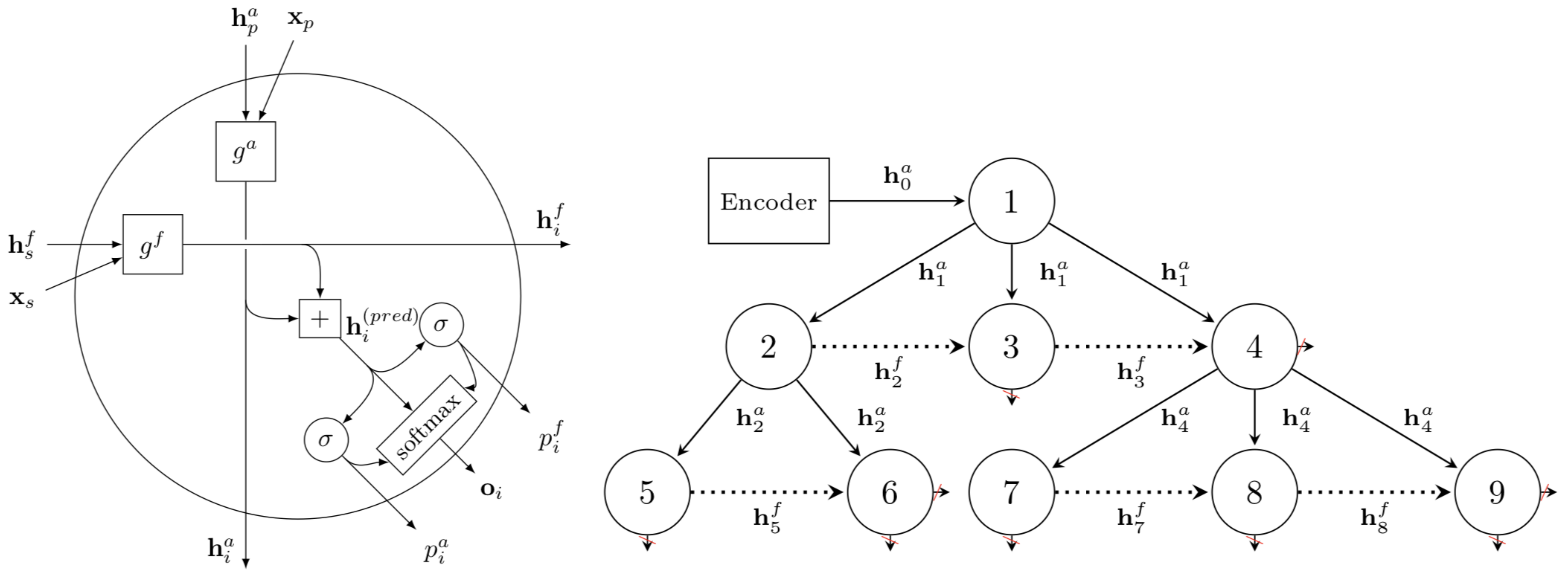
TREE TO TREE: STRUCTURED ENCODING AND DECODING

[Dong & Lapata, 2016; AM & Jaakkola, 2017]



TREE TO TREE: STRUCTURED ENCODING AND DECODING

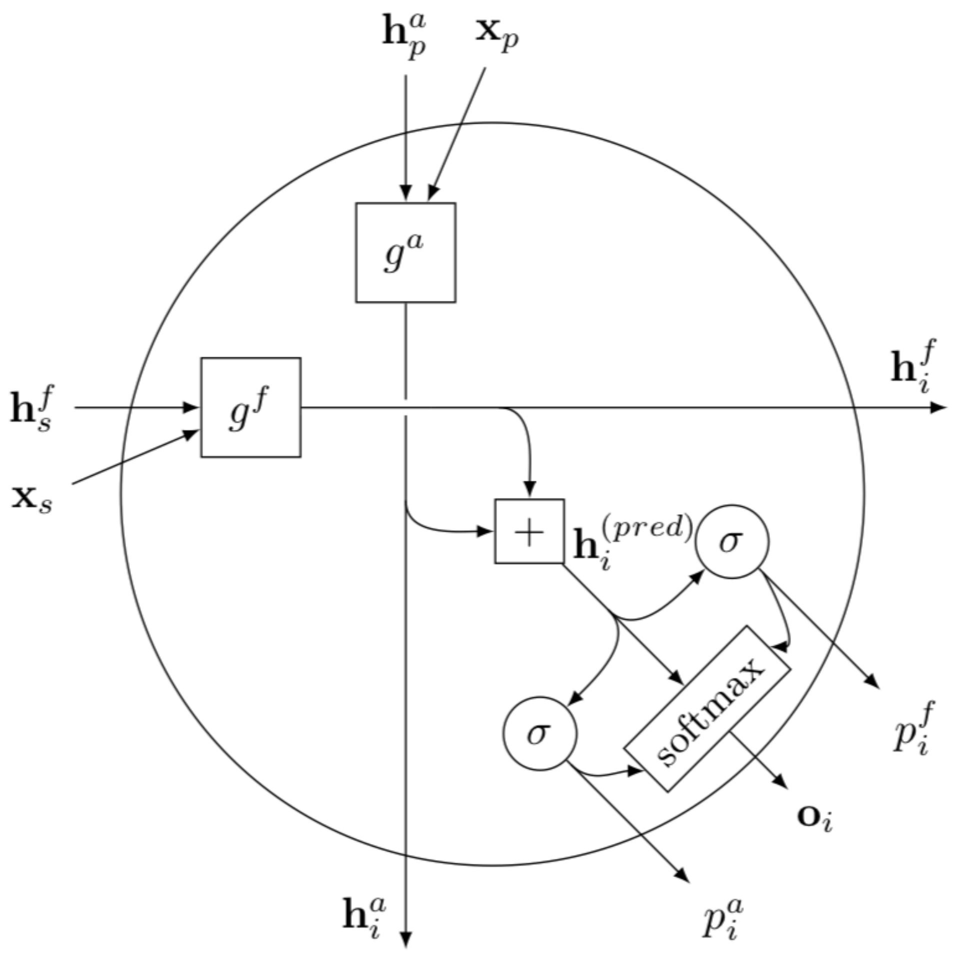
[Dong & Lapata, 2016; AM & Jaakkola, 2017]

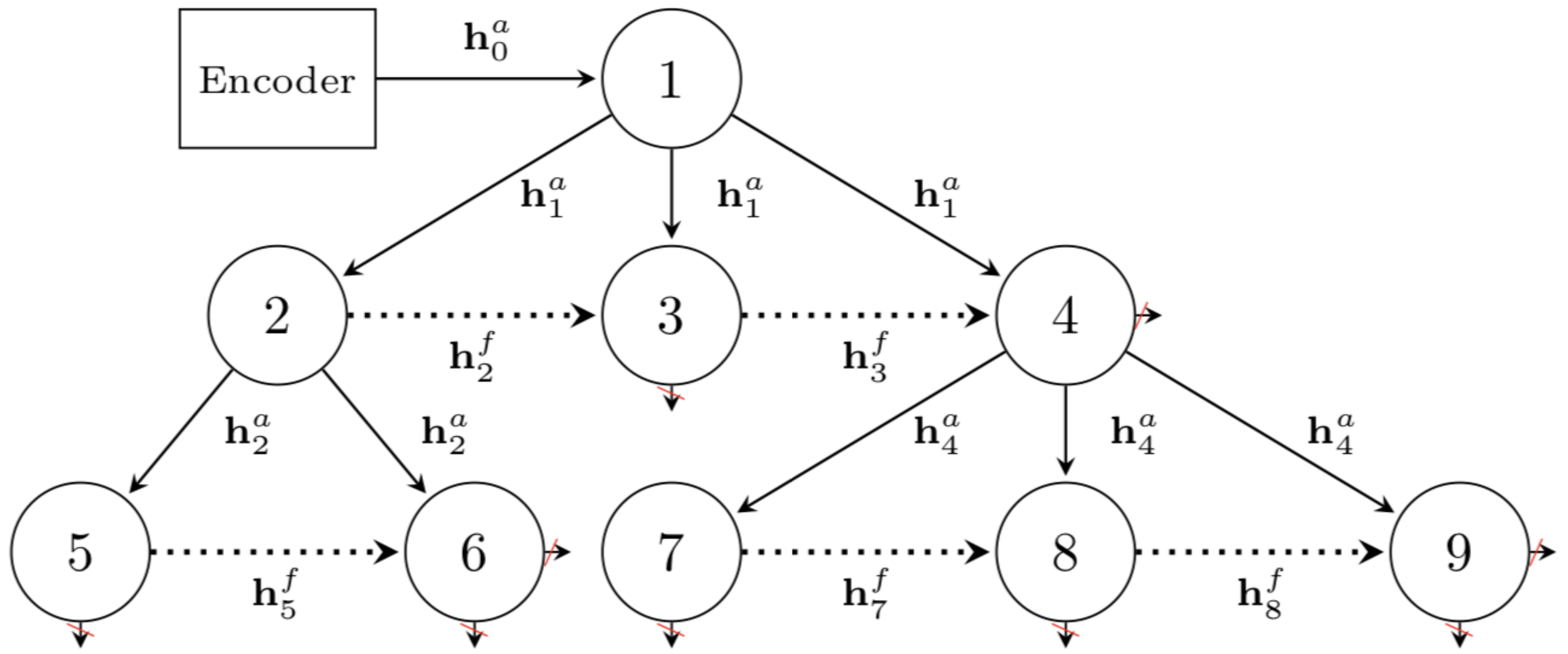


APPLICATION: GENERATING EXECUTABLE PROGRAMS FROM NATURAL LANGUAGE DESCRIPTIONS

TREE TO TREE: STRUCTURED ENCODING AND DECODING

[Dong & Lapata, 2016; AM & Jaakkola, 2017]





APPLICATION: GENERATING EXECUTABLE PROGRAMS FROM NATURAL LANGUAGE DESCRIPTIONS